RESEARCH ARTICLE | NOVEMBER 09 2021

Knowledge-based learning of nonlinear dynamics and chaos

Tom Z. Jiahao ✉ ; M. Ani Hsieh ; Eric Forgoston

Chaos 31, 111101 (2021)
https://doi.org/10.1063/5.0065617

Knowledge-based learning of nonlinear dynamics and chaos

Cite as: Chaos **31**, 111101 (2021); doi: 10.1063/5.0065617
Submitted: 3 August 2021 · Accepted: 20 October 2021 ·
Published Online: 9 November 2021

Tom Z. Jiahao,[1,a)] M. Ani Hsieh,[2] and Eric Forgoston[3]

AFFILIATIONS

[1]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA
[2]Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA
[3]Department of Applied Mathematics and Statistics, Montclair State University, Montclair, New Jersey 07043, USA

[a)]**Author to whom correspondence should be addressed:** zjh@seas.upenn.edu

ABSTRACT

Extracting predictive models from nonlinear systems is a central task in scientific machine learning. One key problem is the reconciliation between modern data-driven approaches and first principles. Despite rapid advances in machine learning techniques, embedding domain knowledge into data-driven models remains a challenge. In this work, we present a universal learning framework for extracting predictive models from nonlinear systems based on observations. Our framework can readily incorporate first principle knowledge because it naturally models nonlinear systems as continuous-time systems. This both improves the extracted models' extrapolation power and reduces the amount of data needed for training. In addition, our framework has the advantages of robustness to observational noise and applicability to irregularly sampled data. We demonstrate the effectiveness of our scheme by learning predictive models for a wide variety of systems including a stiff Van der Pol oscillator, the Lorenz system, and the Kuramoto–Sivashinsky equation. For the Lorenz system, different types of domain knowledge are incorporated to demonstrate the strength of knowledge embedding in data-driven system identification.

We consider the general problem of data-driven modeling of dynamical systems from past observations. We seek to address two important questions: (i) how to use machine learning to model general classes of dynamical systems, especially those with nonlinear and chaotic dynamics, and (ii) how to reconcile data-driven models and first principles knowledge. We propose Knowledge-Based Neural Ordinary Differential Equations (K-NODE), which, unlike other machine learning techniques, is marked by its flexibility for first principles knowledge incorporation and applicability to a wide variety of dynamical systems. Furthermore, our framework is robust to observational noise and irregularly sampled data. Given the abundance of simulated and real data available to scientists and engineers, our framework can be used to extract meaningful correlations and identify system relationships. The proposed framework enables the acquisition of new physical insights and improves the understanding of a wide variety of complex nonlinear systems in diverse scientific disciplines.

I. INTRODUCTION

Recent advances in machine learning and data analytics have largely been fueled by the vast amount of data and have resulted in significant advances in various scientific disciplines.[1,2] Deep learning tools such as recurrent and convolutional neural networks have enabled the identification of coherent data patterns commonly imperceptible to humans. However, most of these data-driven techniques perform poorly when they are trained with insufficient data or used to extrapolate beyond the sampled data. These deficiencies are largely due to the inability to incorporate first principles domain knowledge. By combining first principles with deep learning models, knowledge-based learning can potentially address these challenges by leveraging the extrapolation power of first principles knowledge. However, most existing deep learning models are incompatible with this knowledge.

Since the emergence of calculus, differential equations have been successfully used to model real world phenomena from planetary motions, fluid processes, to biological systems. Differential

equations are compact representations of vector fields on which the evolution of dynamical systems can be realized. Fundamental to the idea of differential equations is the assumption that time and space are continuous. Our vast pool of interpretable first principles knowledge are built upon these underlying assumptions. In contrast, many data-driven approaches for modeling dynamics do not assume continuity for systems and are fundamentally discrete in nature.

One of the earliest data-driven approaches to model dynamical systems builds on Takens' embedding theorem and represents systems as time delay models by embedding the original system's states into delayed snapshots.[3–5] In comparison, modern machine learning strategies employ recurrent neural networks (RNNs) which have memory properties as a result of feedback loops. Both long–short-term memory (LSTM)[6] and reservoir computers (RCs)[7] are examples of RNNs that have been employed to predict two-dimensional fluid flows[8] and to describe the chaotic dynamics of a one-dimensional Kuramoto–Sivashinsky model.[9] However, both time delay models and RNN-based approaches only output the solutions of dynamical systems at prescribed time intervals. Although the solutions may appear continuous, the continuity of the original system is inevitably lost. In fact, this lack of model continuity is one of the reasons for the incompatibility between knowledge and data-driven models. There have been attempts to combine knowledge with reservoir computers[9] for modeling spatiotemporally chaotic systems. Although this hybrid approach showed improved performance, only the solutions of a first principle model were incorporated instead of the model itself. Consequently, the reservoir computer and knowledge are still disjoint.

Most recently, there has been an increased focus in developing strategies that explicitly extract differential equations of a system to represent its dynamics. One method uses sparse regression to determine the combination of basis functions that best describes the vector field from observations[10,11] and, therefore, is able to extract interpretable mathematical models. Nevertheless, a fundamental challenge with this method is the need for a predetermined library of basis functions. If the correct terms are missing from the library, the resulting models may be inadequate at describing the system dynamics. In addition, sparse regression does not scale with high-dimensional systems as the library of functions would become very large.

Chen *et al.* introduced a new family of neural networks, commonly referred to as neural ordinary differential equations (NODE).[12] NODE and its variants[13–16] combine the power of modern machine learning with the formalism of differential equations and are fundamentally continuous.[12] Separately, a scientific machine learning library was built using NODE to incorporate physical constraints into data-driven models. This framework has demonstrated how to incorporate a wide range of knowledge into scientific machine learning.[17] Nonetheless, existing NODE formulations assume stable dynamics for the systems they model in order to achieve convergence.[18] As a result, these and other NODE variants have yet to generalize to systems with unstable or temporally chaotic dynamics.

In this paper, we introduce K-NODE, a novel universal framework for modeling system dynamics from observations. Our proposed framework is universal in that it generalizes the learning task to any continuous-time dynamical system including those with unstable and chaotic behaviors. It is also robust to noisy and irregularly sampled data and can scale to high-dimensional systems. Most importantly, our framework can readily incorporate first principles domain knowledge into neural networks, significantly improving their extrapolation ability. We redefine the system identification task as a constrained optimization problem, which converges regardless of the stability of systems. Similar to NODE, the optimization problem can be efficiently solved using the adjoint sensitivity method,[19] which allows for constant-memory gradient propagation regardless of the number of interpolation steps between the observation time intervals.

We will first demonstrate the effectiveness of our framework by modeling a variety of nonlinear systems using only neural networks. Then, we will show how different forms of knowledge can be incorporated to significantly reduce the amount of data needed for training and to improve the extrapolation power of the models beyond the sampled data.

II. KNOWLEDGE-BASED NEURAL ODES (K-NODE)

We are given m observations of the trajectory generated by a dynamical system sampled at $T = \{t_1, t_2, \ldots, t_m\}, t_i \in \mathbb{R}$,

$$\mathbf{Z} = \begin{bmatrix} (\mathbf{z})^\top (t_1) \\ (\mathbf{z})^\top (t_2) \\ \vdots \\ (\mathbf{z})^\top (t_m) \end{bmatrix} = \begin{bmatrix} z_1(t_1) & z_2(t_1) & \cdots & z_n(t_1) \\ z_1(t_2) & z_2(t_2) & \cdots & z_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ z_1(t_m) & z_2(t_m) & \cdots & z_n(t_m) \end{bmatrix},$$

where $\mathbf{Z} \in \mathbb{R}^{m \times n}$ is the matrix containing the observations, and the vector $\mathbf{z}(t_i) \in \mathbb{R}^n$ is the observation of the state at t_i. Assume the true model of this dynamical system is given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, t),$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the n-dimensional state vector at time t. Let $\tilde{f}$ denote the model based on our current understanding of the system. Then, $\tilde{f}$ is given by

$$\dot{\mathbf{x}} = \tilde{f}(\mathbf{x}, t). \tag{1}$$

We say $\tilde{f}$ is the *knowledge* we have about the system. This knowledge may be perfect, partially known, and/or partially correct, i.e., imperfect—either lacking the correct nonlinear terms or not having the correct steady state behavior of the true model. Given this knowledge $\tilde{f}$, our goal is to approximate the function f with a hybrid model $\hat{f}(\mathbf{x}, t, \tilde{f}(\mathbf{x}, t), \boldsymbol{\theta})$, which incorporates the knowledge $\tilde{f}$, and is parameterized with the vector $\boldsymbol{\theta}$. In this work, we represent $\hat{f}$ using a combination of artificial neural networks and the knowledge $\tilde{f}$.

While it is flexible how $\tilde{f}$ gets incorporated into $\hat{f}$, in this work we consider linearly coupling the outputs from $\tilde{f}$ and the neural network using a matrix $\mathbf{M}_{out} \in \mathbb{R}^{p \times n}$ with output biases. Both $\mathbf{M}_{out}$ and the biases are co-trained with the neural network. Given the input size p, $\mathbf{M}_{out}$ and the biases are initialized uniformly random from $[-1/\sqrt{p}, 1/\sqrt{p}]$. The hybrid architecture of $\hat{f}$ is illustrated in Fig. 1. Note that our approach to knowledge incorporation differs fundamentally from those of hybrid reservoir computers.[9] While hybrid reservoir computers first perform numerical integration for both

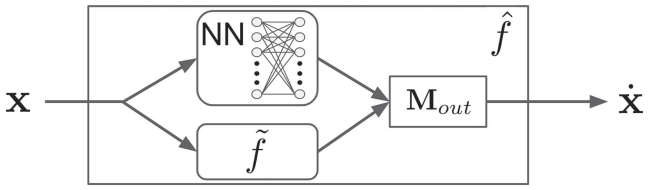

FIG. 1. A hybrid architecture incorporating an imperfect vector field $\tilde{f}$ into the model $\hat{f}$. Outputs from the neural network and the knowledge block $\tilde{f}$ are then linearly coupled.

the knowledge and its reservoirs separately and then linearly couple their respective solutions, K-NODE first linearly couples the vector fields and then performs numerical integration. K-NODE is able to directly couple the vector field owing to its explicit nature in modeling differential equations.

We then pose this system identification task as the following constrained optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & L(\boldsymbol{\theta}) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \hat{f}(\mathbf{x}, t, \tilde{f}(\mathbf{x}, t), \boldsymbol{\theta}), \\ & \mathbf{x}(t_s, \mathbf{z}(t_s)) = \mathbf{z}(t_s), \ t_s \in T, \end{aligned} \tag{2}$$

where the first constraint is the differential equation defined by the hybrid model $\hat{f}$, the second constraint specifies the initial conditions, and the parameters $\boldsymbol{\theta}$ can then be estimated by $\boldsymbol{\theta} = \text{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$.

For the objective function in (2), we define an L^2 loss function between the observed trajectory and the trajectory generated by $\hat{f}$ given by

$$L(\boldsymbol{\theta}) = \frac{1}{m-\alpha} \sum_{i=1}^{m-\alpha} \frac{1}{\alpha} \int_{t_i}^{t_{i+\alpha}} \delta(t_s - \tau) \|\mathbf{x}(\tau, \mathbf{z}(t_i)) - \mathbf{z}(\tau)\|^2 \, d\tau, \tag{3}$$

where $t_s \in T$ is any sampling time point, and δ is the Dirac delta function. In general, for any given initial condition, a trajectory can be generated from $\hat{f}$ using a suitable numerical integration scheme. Thus, $\mathbf{x}(\tau, \mathbf{z}(t_i))$ in $L(\boldsymbol{\theta})$ is defined as the state at time τ generated by $\hat{f}$ with the initial condition $\mathbf{x}(t_i, \mathbf{z}(t_i)) = \mathbf{z}(t_i)$ at time t_i given by

$$\mathbf{x}(\tau, \mathbf{z}(t_i)) = \mathbf{z}(t_i) + \int_{t_i}^{\tau} \hat{f}\left(\mathbf{x}(\omega, \mathbf{z}(t_i)), \omega, \tilde{f}(\mathbf{x}, \omega), \boldsymbol{\theta}\right) d\omega. \tag{4}$$

Note that the state at t_{j+1} generated by $\hat{f}$ with the initial condition $\mathbf{z}(t_j)$ is called a one-step-ahead diffeomorphic flow associated with $\hat{f}$. In our loss function given by (3), the integral from t_i to $t_{i+\alpha}$ requires an α-step-ahead diffeomorphism associated with $\hat{f}$, and we call α the *lookahead*. While some existing work assumes $\alpha = 1$,[10,20,21] we have found that using a larger lookahead can sometimes yield better results. Hence, we treat α as a hyperparameter to be tuned. This constrained optimization formulation is illustrated with a 1D example in Fig. 2. Formulation similar to (2) has been used to learn chaotic systems using models, which heavily incorporates the physics but the optimization task is not explicitly stated.[22] In the original NODE formulation, the loss is defined over the entire trajectory predicted

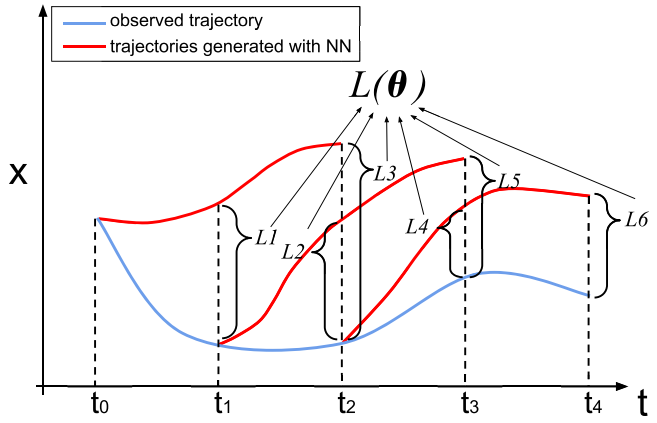

FIG. 2. A 1D example of the constrained optimization problem. The blue curve is the observation, and the red curves are the trajectories generated by the neural network $\hat{f}$. In this example, $\alpha = 2$ and, therefore, the neural network generates a trajectory of length 2 beginning with the sampled state at every time step. The loss is then computed between the blue and red curves at every sampled time steps.

using a neural network. We note that this formulation is one of the main contributors for the numerical instability when learning chaotic systems since chaotic trajectories are deemed to diverge exponentially fast in time. While there is no theoretical guarantee for convergence using our optimization formulation in (2), our empirical results have shown that our formulation is capable of achieving convergence for systems with chaotic or unstable dynamics.

While the optimization problem in (2) can be solved using the conventional backpropagation, the adjoint sensitivity method is a memory efficient alternative that is used in our work. The method description and its derivation are included in S2 in the supplementary material. In this work, the neural network $\hat{f}$ is implemented using the PyTorch nn module, and the Adam optimizer is used to update the neural network parameters. Last, the adjoint sensitivity method is implemented as a custom autograd function for gradient propagation.[23]

The ability to incorporate knowledge should require less training data for the learning framework to capture the true dynamics of the system of interest. And most importantly, the resulting hybrid model will be able to extrapolate beyond the sampled data, as we will demonstrate in our results.

III. RESULTS

Using our universal learning framework, we consider different dynamical systems of varying complexities. For each system, training data are simulated with a suitable integration scheme. The simulation and training parameters are summarized in S3 in the supplementary material. Note that the training solver and its parameters must be chosen in a way such that it can realize the system's trajectories. Generally, solver selection should be based on criteria including system dynamics, solution stability, computation efficiency, and solver robustness.[24] For instance, Euler's method with

a large step size may not suffice for stiff systems such as the Van der Pol oscillator as it will result in numerical instability. The proposed framework is evaluated with respect to its ability to reproduce the dynamics of the actual system and to extrapolate or predict future observations. We also validate the framework for learning the dynamics of systems without prior knowledge and with limited prior knowledge.

To help better visualize the evolution of system states, we plot the trajectories of the Hopf normal form and the chaotic Lorenz system in gradient color, where their trajectories start in blue (RGBA: [0.0, 0.0, 0.5, 1.0]) and end in yellow (RGBA: [0.9, 1.0, 0.0, 1.0]).

A. Learning without knowledge

In this subsection, we demonstrate learning without assuming any knowledge about the system, i.e., only a neural network is used to model the systems and $\tilde{f} = \mathbf{0}$.

Van der Pol oscillator. We first demonstrate our framework in learning the dynamics of a stiff system. Consider the Van der Pol oscillator, which is a non-conservative oscillator with a limit cycle behavior[25] and has important applications in the modeling of periodic biological systems. The oscillator's second-order differential form is[26]

$$\frac{d^2x}{dt^2} = \mu(1 - x^2)\frac{dx}{dt} - x, \tag{5}$$

but it is often written in its first-order form as the system of equations

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -x + \mu(y - x^2 y). \end{aligned} \tag{6}$$

This system is known for its increasing *stiffness* as μ is increased. Here, we consider the stiff Van der Pol oscillator with $\mu = 10$. The training data are a trajectory of 1000 points simulated with the initial condition $[x_0, y_0]^\top = [2, 0]^\top$. The trained model correctly captures the stiffness and limit cycle behavior of the Van der Pol oscillator as shown in Fig. 3.

Hopf normal form. A bifurcation occurs when a dynamical system undergoes a sudden change in behavior when its parameters are varied. We consider a canonical model for a Hopf bifurcation, the Hopf normal form, which is given as

$$\begin{aligned} \dot{x} &= \mu x + y - x(x^2 + y^2), \\ \dot{y} &= -x + \mu y - y(x^2 + y^2), \end{aligned} \tag{7}$$

where μ is the system parameter which, when varied, gives rise to a Hopf bifurcation. We incorporate the parameter μ into our model as a third dimension to the input vector, and set its time derivative to 0, i.e., $\dot{\mu} = 0$. Training data, as shown in Fig. 4(a), are simulated with three parameter values $\mu = \{-0.1, 0.35, 0.8\}$. The trained model correctly captures the bifurcation as shown in Fig. 4(b). The trained model can accurately interpolate between $\mu = -0.1$ to $\mu = 0.8$, and extrapolate from $\mu = -0.2$ to $\mu = 1.0$, which is beyond the range of μ in the training data. This demonstrates the generalization power of the trained model over system parameters.

Chaotic Lorenz system. Next, we consider the chaotic Lorenz system[27]

$$\begin{aligned} \dot{x} &= 10(y - x), \quad \dot{y} = x(28 - z) - y, \\ \dot{z} &= xy - (8/3)z. \end{aligned} \tag{8}$$

We simulate training data with the initial condition $[x(0), y(0), z(0)]^\top = [-8, 7, 27]^\top$. As shown in Fig. 5, the trained model is able to capture the bistable structure of the Lorenz attractor both within and beyond the duration of the training data.

To determine whether the resulting model is truly chaotic, we adopt the 0-1 test proposed by Gottwald and Melbourne,[28] which directly applies to time series data. The 0-1 test is binary, i.e., under ideal conditions, it outputs 1 if the system is chaotic and 0 otherwise. In practice, the outputs are approximately 1 and 0 for chaotic and non-chaotic systems. Details about the 0-1 test are included in S1 in the supplementary material.

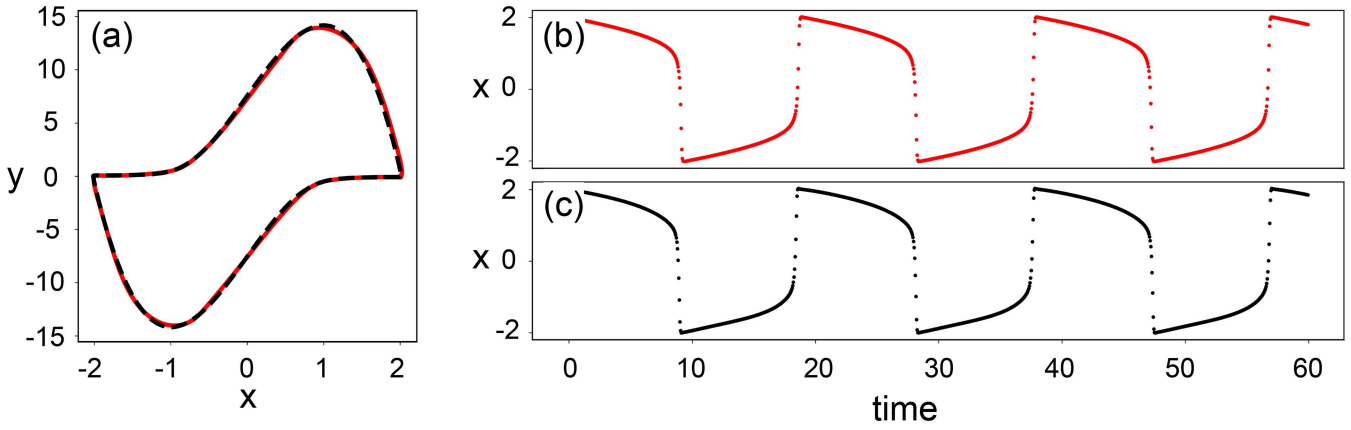

FIG. 3. Prediction of the stiff Van der Pol oscillator with $\mu = 10$. (a) Simulated trajectory (red) vs the model predicted trajectory (dotted black). (b) True trajectory and (c) predicted trajectory of x as a function of time for the Van der Pol oscillator.

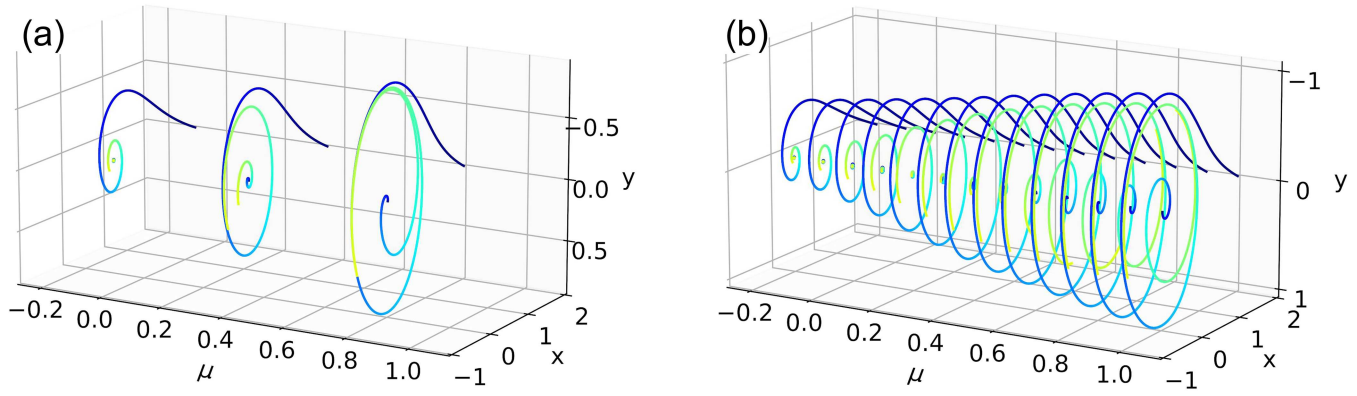

FIG. 4. Learning the Hopf normal form. (a) Training data. (b) The trained model interpolates between $\mu = -0.2$ and $\mu = 1.0$.

Performing the 0-1 test on the predicted trajectory gives $K_c = 1.079$, showing that the identified system has chaotic dynamics.

Kuramoto–Sivashinsky equation. Moreover, our framework can learn the dynamics of spatiotemporally chaotic PDEs. Consider the one-dimensional *Kuramoto–Sivashinsky equation* (KS) with periodic boundary conditions $y = y(x,t)$ and $y(x+L,t) = y(x,t)$,[29,30]

$$\frac{\partial y}{\partial t} = -y\frac{\partial y}{\partial x} - \frac{\partial^2 y}{\partial^2 x} - \frac{\partial^4 y}{\partial^4 x}. \tag{9}$$

We consider a 64-grid KS equation with $L = 60$. The most positive Lyapunov exponent is $\Lambda_{max} = 0.089$, and the Kaplan–Yorke dimension of the attractor is $D_{KY} = 13.56$.[31] We use a natural time scale, the Lyapunov time $\Lambda_{max}t$, for model evaluation. Since this system is spatially high-dimensional, we first use convolutional layers in our neural network for dimensionality reduction and then a linear coupling matrix to restore the spatial dimension of the output. To generate training data, we discard the first 1000 steps of transient data and simulate a total of 8000 steps. Training data are then taken to be the first 5000 steps, which equates to 111.25 Lyapunov time. Testing data are taken from the 6000th to the 8000th steps of the simulated data. Figure 6 shows that the trained neural network can capture the first 4 Lyapunov times with high accuracy before the trajectories diverge. Even after the trajectories diverge, the system behaves similarly to the actual system.

To qualitatively evaluate the performance of the trained model, we adopt the dimensionality reduction technique, proper orthogonal decomposition (POD),[32] which maps the *energy* of a given system. The term *energy* is analogous to data variance in a principal component analysis.[33] Figure 7 shows the scatterplot of the *energy* for each of the 64 dimensions in the testing data and the predictions made by the neural network. It can be observed that the prediction has a similar energy signature as the testing data.

B. Knowledge-based learning

In this subsection, we consider learning with imperfect knowledge about a system. First, assume the true system is the Lorenz system from (8). However, we only have the knowledge of a Lorenz

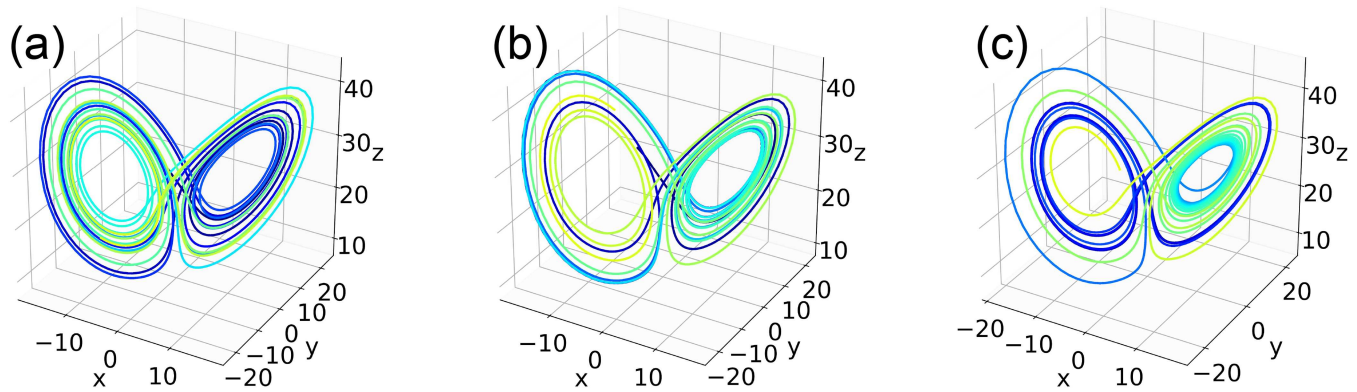

FIG. 5. Learning the chaotic Lorenz system with a neural network without knowledge. (a) Training data ($t = 0$ to $t = 20$). (b) Model prediction ($t = 0$ to $t = 20$). (c) Model extrapolation ($t = 20$ to $t = 40$).

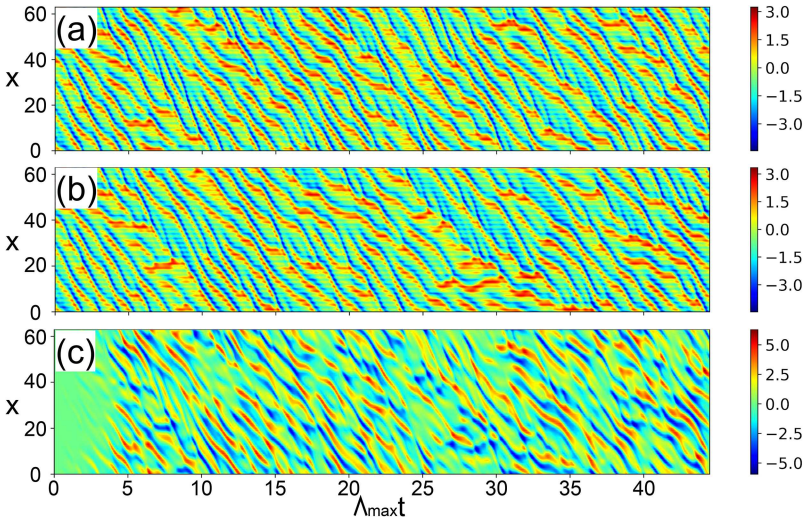

FIG. 6. Trained KS model prediction on the testing data. (a) Testing data. (b) Prediction with the trained model using the first step of test data as the initial condition. (c) Difference between the test data and prediction.

system with incorrect coefficients. The incorrect system keeps $\dot{x}$ and $\dot{z}$ the same but has $\dot{y} = x(-4.8 - z) + 7.2y$. This incorrect system is, therefore, our knowledge $\tilde{f}$. Figure 8(b) shows the trajectory of this incorrect system, which has periodic rather than chaotic behavior. We combine this incorrect system with a neural network according to the hybrid scheme shown in Fig. 1. The training data, shown in Fig. 8(a), are generated with the initial condition $[x(0), y(0), z(0)]^{\top} = [-8, 7, 27]^{\top}$. Note that this training data have only 2/5 the length used for training without knowledge. Figure 8(c) shows that the trained model restores the bistable structure and chaotic dynamics. Performing the 0-1 test on the knowledge gives $K_c = 0.036$, and the predicted trajectory gives $K_c = 1.11$, showing that the trained model has chaotic dynamics, even though the knowledge was not chaotic.

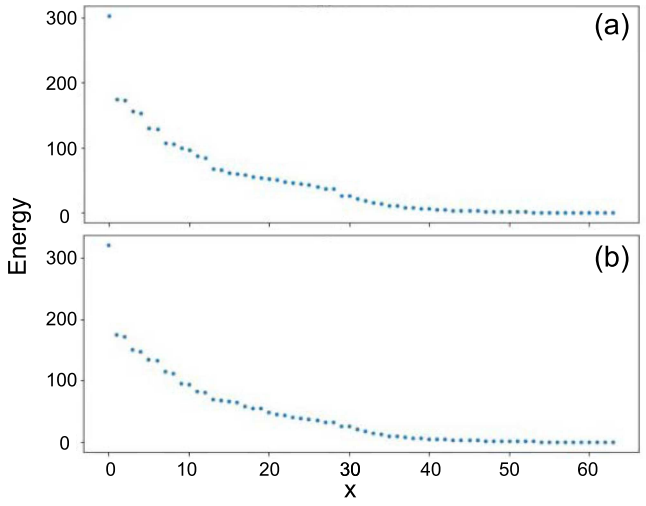

FIG. 7. Comparing the POD decomposition for testing data and predictions. The scatterplots show the energy of the spatial modes from the POD decomposition of (a) the testing data and (b) the predictions on the testing data.

Next, we consider incorporating the knowledge generated by SINDy, a system identification method that uses sparse regression on a library of basis functions.[10] When the correct nonlinear terms are missing from the library, SINDy may fail to capture the actual dynamics. Here, we present an example of learning the chaotic Lorenz system using an imperfectly identified model from SINDy. We simulate observations for the chaotic Lorenz system using RK4 with a step size of 0.01 from $t = 0$ to $t = 80$ and sample training data with a step size of 0.02. We then perform SINDy on these data using a library of polynomials up to the fourth order. SINDy identifies the system shown in Fig. 9(a), which correctly captures the bistable structure of the chaotic Lorenz system. However, if the nonlinearities xz and xy are excluded from the library of functions, SINDy identifies a system shown in Fig. 9(c), which is not chaotic as the trajectory eventually forms a Fig. 8 shaped limit cycle. If x, xz, and xy are excluded, the identified system, as shown in Fig. 9(b), fails completely to capture the bistable structure.

The incorrectly identified system in Fig. 9(c) has the form

$$\begin{aligned} \dot{x} &= -9.913x + 9.913y, \\ \dot{y} &= -7.175x + 20.507y - 0.613yz, \\ \dot{z} &= -3.05z + 0.504x^2 + 0.479y^2, \end{aligned} \tag{10}$$

which we will use as the knowledge $\tilde{f}$. Since the terms xy and xz are excluded from the regression library, both $\dot{y}$ and $\dot{z}$ have incorrect nonlinearities. By incorporating this incorrectly identified model according to the hybrid scheme, the trained model can correctly restore the bistable structure as shown in Fig. 9(d) by only using training data from $t = 0$ to $t = 8$. Note again that the training data are only 2/5 the length used for training without knowledge. Performing the 0-1 test on the knowledge gives $K_c = 0.064$, and the predicted trajectory gives $K_c = 0.832$, showing that the trained model has chaotic dynamics.

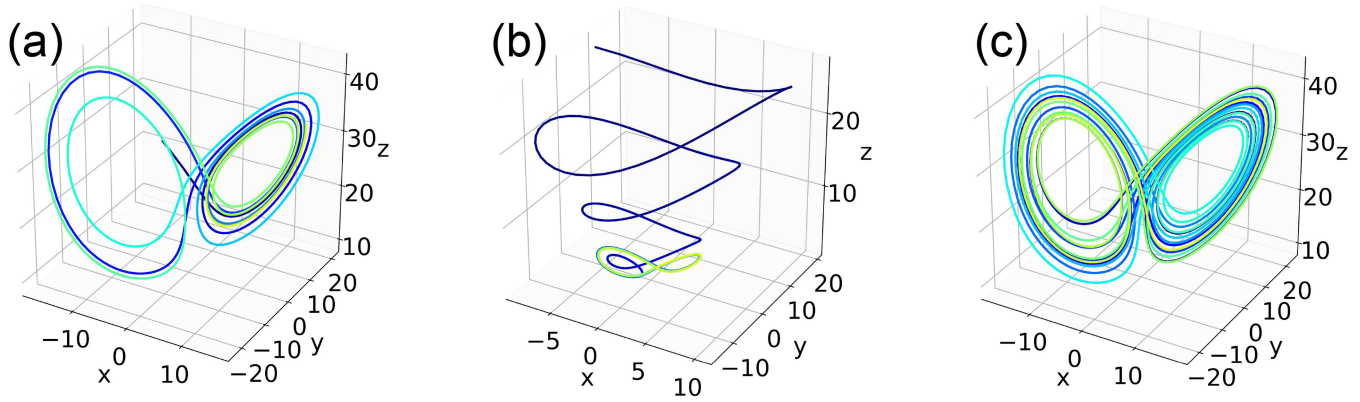

FIG. 8. Learning the Lorenz system with knowledge. (a) Training data ($t = 0$ to $t = 8$). (b) Trajectory ($t = 0$ to $t = 20$) of the incorrect Lorenz system. (c) Extrapolation ($t = 8$ to $t = 28$) using a trained model with the incorrect system as knowledge.

We trained on the same amount of data ($t = 0$ to $t = 8$) without incorporating any knowledge. The neural network tends to overfit and "memorize" the data. When further extrapolating in time, the trained model tends to either a limit cycle or fixed point and is unable to reproduce the chaotic dynamics.

Next, we show that knowledge incorporation gives the resulting model better extrapolation power beyond the sampled data. Here, we compare the three trained models by using them to produce trajectories at initial conditions different from $[-8, 7, 27]^\top$, which is used for generating training data. Figure 10 shows the predicted trajectories using the initial conditions $[-8, 7, 12]^\top$, and $[10, -5, 27]^\top$. While the chaotic behavior is completely lost in the neural network trained without any knowledge incorporation, both knowledge-based neural networks can still reproduce the bistable structure. Performing the 0-1 test on both knowledge-based models shows that they are chaotic.

C. Learning from noisy observations

Last, we demonstrate our framework's robustness to observational noise. Considering the Lorenz system, we use the same training data as training without knowledge but now include Gaussian noise with zero mean and 0.1 variance ($\sim N(0, 0.1)$). This noisy trajectory is shown in Fig. 11(a). The trained neural network correctly captures the chaotic dynamics as shown in Figs. 11(b) and 11(c). Performing the 0-1 test on the predicted trajectory gives $K_c = 1.10$.

We also consider noisy observations from the KS equation. Using the same training data as learning without knowledge, we include Gaussian noise with zero mean and 0.01 variance ($\sim N(0, 0.01)$). Figure 12 shows the learning result. Though the trained model can only accurately predict about 1 Lyapunov time before the trajectories diverge, this demonstrates the effect of noise on the chaotic dynamics, i.e., a slight perturbation in the initial

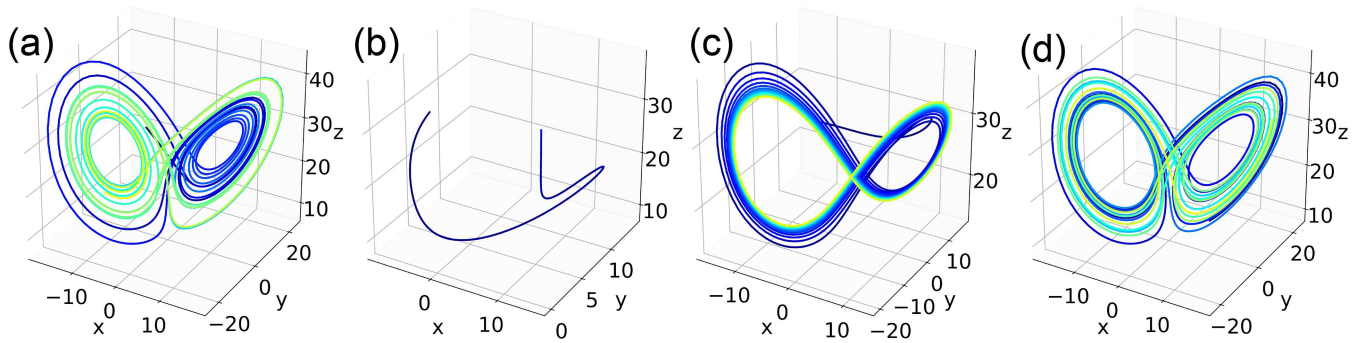

FIG. 9. Systems identified by SINDy using a library of polynomial functions up to the fourth order after excluding different nonlinearities from the function library. Each plot shows a trajectory for $t = 0$ to $t = 20$. (a) System identified if all terms are included in the library. (b) System identified if x, xy, and xz are excluded from the library. (c) System identified if xy and xz are excluded from the library. (d) Extrapolation ($t = 8$ to $t = 28$) using a trained model with the model in (c) as knowledge.

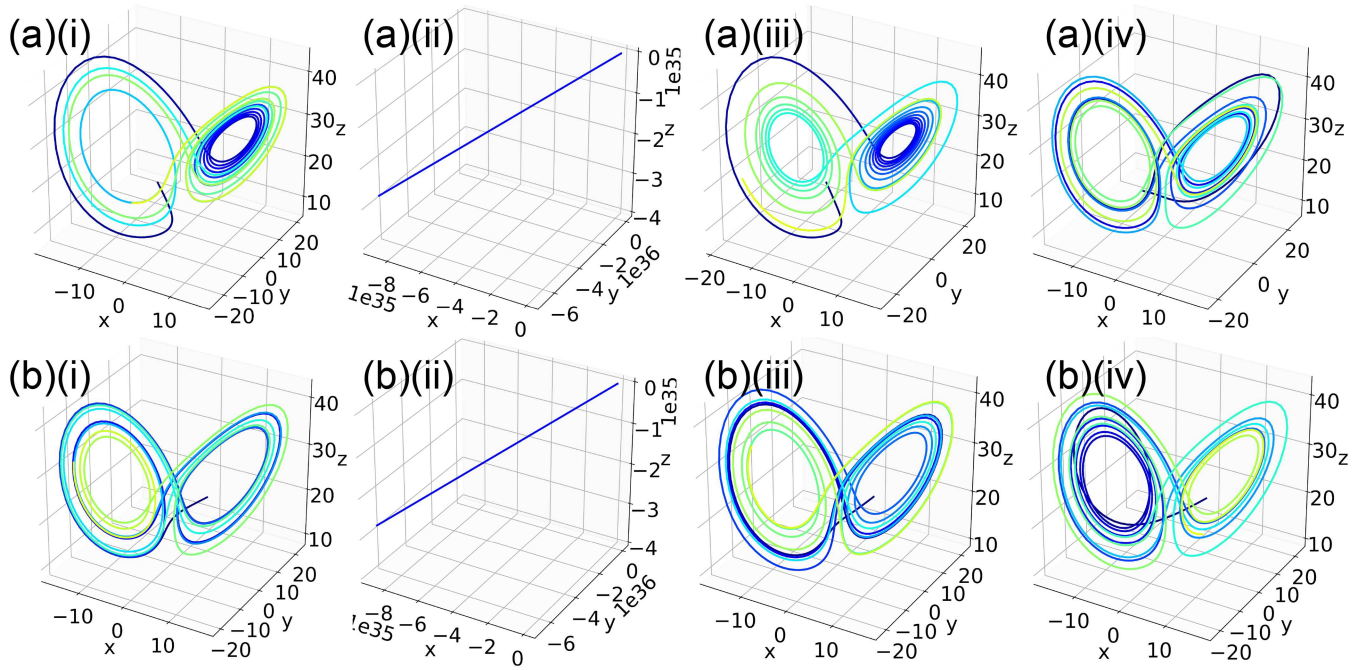

FIG. 10. Model predictions from different initial conditions (t = 0 to t = 12). Trajectories are generated from the initial condition (a) $[-8, 7, 12]^\top$ and (b) $[10, -5, 27]^\top$ using the (i) true Lorenz system, (ii) the neural network without knowledge, (iii) the neural network combined with incorrect Lorenz system, and (iv) the neural network combined with an incorrectly identified model from SINDy.

condition as a result of noise makes the trajectories diverge exponentially fast. However, it can be seen that the predicted trajectories behave similarly to the true system.

We note that the same neural network architecture converges much faster when trained on noisy observations. This is consistent with recent work that adds noise to observations of chaotic systems to stabilize the training process.[9]

IV. DISCUSSION

We have demonstrated the universality of our framework by learning a wide variety of systems including stiff, bifurcating, and chaotic systems. We have also shown that with spatial convolution, our framework can easily scale and model high-dimensional systems. Note that since our framework does not require fixed time intervals for the training data, it can also learn from irregularly

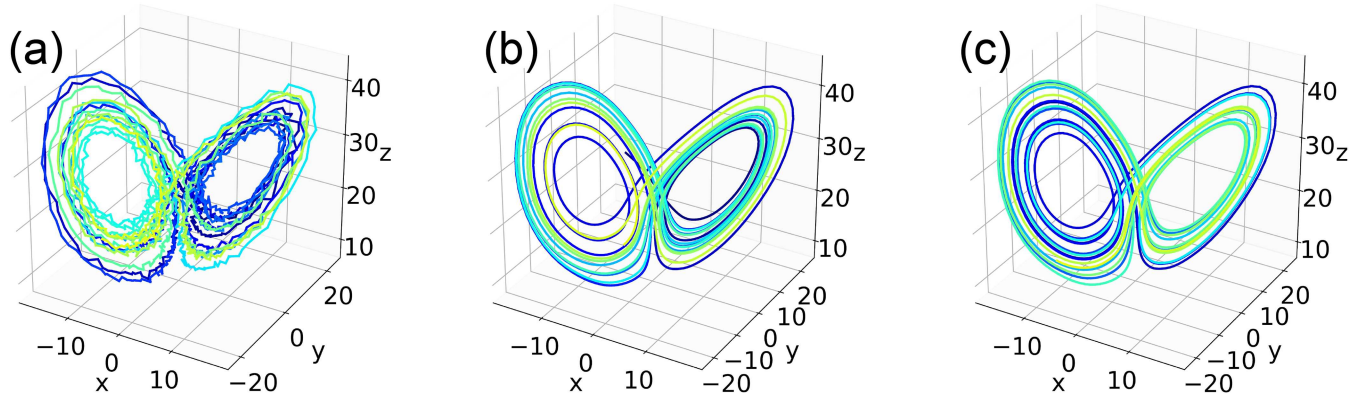

FIG. 11. Learning the Lorenz system from noisy observations. (a) Training data ($t = 0$ to $t = 20$) with observational noise $\sim N(0, 0.1)$. (b) Model prediction ($t = 0$ to $t = 20$). (c) Model extrapolation ($t = 20$ to $t = 40$).

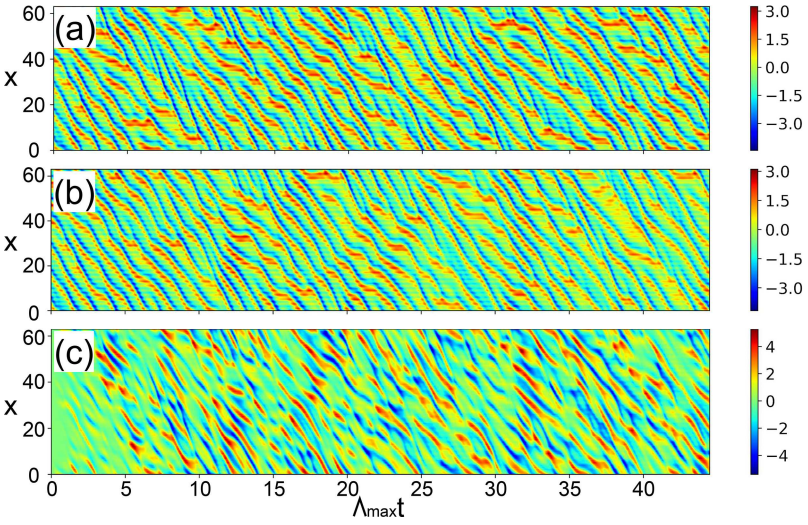

FIG. 12. Learning from noisy observations of KS equation. (a) Testing data with $\sim N(0, 0.01)$ Gaussian noise. (b) Prediction with the trained model using the first step of test data as the initial condition. (c) Difference between test data and prediction.

sampled trajectories (see in S6 in the supplementary material). Most importantly, K-NODE is able to recover the dynamics of imperfect models by using them as knowledge. This demonstrates neural networks' capability to make up for the incorrect nonlinearities in the governing equations of systems.

In this framework, the neural network is trained with respect to a numerical solver, which can be different from the simulation solver used to generate the training data as demonstrated in learning the KS model. This means that we have the freedom to choose the training solver and its parameters depending on our needs. For example, we can use lower order solvers to speed up training at the expense of model accuracy, and we can perform finer temporal interpolations by using a smaller step size for the training solver at the expense of training speed. This is an important feature that ensures excellent accuracy without sacrificing computational speed. Moreover, it ensures that our framework can be applied not only in the case when one has high-fidelity simulation data but also when one is working with real data, which may be noisy or have missing/sparse data.

For future work, we seek to explore more hybrid-learning approaches by incorporating different types of knowledge. Moreover, besides using spatial convolutions for dimension reduction, we will investigate parallel learning using our framework to achieve even better scalability for very high-dimensional systems. Furthermore, in the hybrid-learning scheme, we have full knowledge about the trained linear coupling matrix $\mathbf{M}_{out}$, which dictates how knowledge is combined with neural networks. This matrix $\mathbf{M}_{out}$ could potentially inform us of the correctness of assumptions made as well as help us to understand the role of neural networks in the learned dynamics. In the future, we hope to leverage this to extract additional physical insight into a wide range of dynamical systems. Given the amount of model and real data available to us, and considering the complexity of these systems, our universal learning framework can be used to understand meaningful correlations and establish new relationships between processes, thereby enhancing our knowledge.

SUPPLEMENTARY MATERIAL

See the supplementary material for simulation and training details, tips for choosing the lookahead, additional experimental results on learning from noisy or irregularly sampled data, derivations of the adjoint sensitivity method, and details of the 0-1 test for chaos.

ACKNOWLEDGMENTS

This work was funded by the Office of Naval Research (ONR) (Award No. 14-19-1-2253).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in TomJZ/K-NODE at github.com/TomJZ/K-NODE, Ref. 34.

REFERENCES

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science **349**, 255–260 (2015).
[2] V. Marx, "The big challenges of big data," Nature **498**, 255–260 (2013).
[3] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence*, edited by D. A. Rand and L. S. Young (Springer-Verlag, New York, 1980), pp. 366–381.
[4] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon, "Identification of nonlinear systems using polynomial nonlinear state space models," Automatica **46**, 647–656 (2010).
[5] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," *Time Series Prediction* (Addison-Wesley, 1994), pp. 195–217.
[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).

[7]H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," GMD Technical Report 148, German National Research Center for Information Technology, Bonn, Germany, 2001.
[8]M. Qraitem, D. Kularatne, E. Forgoston, and M. A. Hsieh, "Bridging the gap: Machine learning to resolve improperly modeled dynamics," Physica D **414**, 132736 (2020).
[9]A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott, "Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems," Chaos **30**, 053111 (2020).
[10]S. Brunton, J. Proctor, and J. Kutz, "Discovering governing equations from data: Sparse identification of nonlinear dynamical systems," Proc. Natl. Acad. Sci. U.S.A. **113**, 3932–3937 (2015).
[11]A. A. AlMomani, J. Sun, and E. Bollt, "How entropic regression beats the outliers problem in nonlinear system identification," Chaos **30**, 013107 (2020).
[12]T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018), pp. 6572–6583.
[13]X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. Duvenaud, "Scalable gradients for stochastic differential equations," in *International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research Vol. 108, edited by S. Chiappa and R. Calandra (PMLR, 2020), pp. 3870–3882.
[14]C. Zang and F. Wang, "Neural dynamics on complex networks," abs/1908.06491 (2019).
[15]I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari, "Learning dynamical systems from partial observations," CoRR abs/1902.11136 (2019).
[16]E. Dupont, A. Doucet, and Y. W. Teh, "Augmented neural odes," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2019), Vol. 32, pp. 3140–3150.
[17]C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, "Universal differential equations for scientific machine learning," arXiv:2001.04385 (2020).
[18]I. M. Ross, *A Primer on Pontryagin's Principle in Optimal Control* (Collegiate Publishers, San Francisco, 2009).
[19]Y. Cao, S. Li, L. Petzold, and R. Serban, "Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution," SIAM J. Sci. Comput. **24**, 1076–1089 (2003).
[20]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," arXiv:1801.01236 [math.DS] (2018).
[21]S. Ouala, D. Nguyen, L. Drumetz, B. Chapron, A. Pascual, F. Collard, L. Gaultier, and R. Fablet, "Learning latent dynamics for partially-observed chaotic systems," arXiv:1907.02452 [stat.ML] (2019).
[22]M. Gelbrecht, N. Boers, and J. Kurths, "Neural partial differential equations for chaotic systems," New J. Phys. **23**, 043005 (2021).
[23]M. Surtsukov, "Neural-ODE," https://github.com/msurtsukov/neural-ode (2019).
[24]MathWorks, "Choose a solver—Matlab & Simulink," Web (2021).
[25]R. H. Enns and G. McGuire, "Limit cycles," in *Nonlinear Physics with Maple for Scientists and Engineers* (Birkhäuser, Boston, MA, 1997), pp. 183–208.
[26]B. van der Pol Jun and D. Sc, "LXXXVIII on "relaxation-oscillations," London Edinburgh Dublin Philos. Mag. J. Sci. **2**, 978–992 (1926).
[27]E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[28]G. Gottwald and I. Melbourne, "A new test for chaos in deterministic systems," Proc. R. Soc. London, Ser. A **460**, 603–611 (2004).
[29]Y. Kuramoto, "Diffusion-induced chaos in reaction systems," Prog. Theor. Phys. Suppl. **64**, 346–367 (1978).
[30]G. Sivashinsky, "Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations," Acta Astronaut. **4**, 1177–1206 (1977).
[31]R. A. Edson, J. E. Bunder, T. W. Mattner, and A. J. Roberts, "Lyapunov exponents of the Kuramoto–Sivashinsky PDE," ANZIAM J. **61**, 270–285 (2019).
[32]P. Holmes, J. L. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge Monographs on Mechanics (Cambridge University Press, 1996).
[33]I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," Philos. Trans. R. Soc. A **374**, 20150202 (2016).
[34]T. Z. Jiahao, "K-NODE," https://github.com/TomJZ/K-NODE.