# Data-Driven Discovery of High-Dimensional Dynamical Systems with Sparse Interpretable Neural Networks

Siyuan Xing [1,*] Qingyu Han,[2] Efstathios G. Charalampidis [3,4] and Ying-Cheng Lai [5]

[1]*Mechanical Engineering Department, California Polytechnic State University, San Luis Obispo, California 93407-0403, USA*
[2]*Electrical Engineering Department, California Polytechnic State University, San Luis Obispo, California 93407-0403, USA*
[3]*Department of Mathematics and Statistics, San Diego State University, San Diego, California 92182-7720, USA*
[4]*Computational Science Research Center, San Diego State University, San Diego, California 92182-7720, USA*
[5]*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, Arizona 85287, USA*

(Received 6 March 2026; accepted 19 August 2026; published 29 September 2026)

Existing approaches to explicit data-driven discovery of nonlinear dynamical systems face a curse of dimensionality because candidate libraries grow combinatorially with system dimension. This challenge is particularly severe for high-dimensional systems arising from spatially discretized fields and large complex networks. We develop sparse regression embedded interpretable network (SREINet), a machine-learning framework that embeds sparse regression in an interpretable neural network and uses a sparsity-promoting periodic pruning scheme. Across six paradigmatic systems, SREINet accurately identifies velocity fields with more than 100 dimensions and extrapolates coherent structures from untrained data. Head-to-head benchmarks against representative equation-discovery and predictive methods on clean and noisy systems show that SREINet combines scalable computation with accurate, parsimonious equation recovery, particularly in high-dimensional settings. We further validate the framework using empirical data from a triple-pendulum experiment. These results demonstrate a practical route toward interpretable model discovery for large-scale nonlinear systems and suggest that the framework can be extended to systems with thousands of dimensions.

DOI: 10.1103/pdkf-98zb

## I. INTRODUCTION

High-dimensional dynamical systems serve as the mathematical foundation for describing a vast array of complex phenomena in science and engineering, ranging from spatiotemporal patterns in fluid turbulence [1] and Bose-Einstein condensation [2] to collective behaviors in large-scale complex networks [3]. In practice, these systems are often represented as hundreds or thousands of coupled ordinary differential equations (ODEs), whether arising from the spatial discretization of continuous fields or from the inherent coupling of discrete components. The governing equations generating these complex phenomena have mostly been derived from first principles (e.g., symmetries and conservation laws [4]) or are based on empirical/phenomenological considerations. Situations have become increasingly common where large amounts of high-dimensional data are collected through experiments and observations, and it is of interest to find the underlying governing equations from data. There are three main data-driven approaches to system modeling and discovery, which are based, respectively, on symbolic regression (SR), sparse optimization, and machine learning. In spite of the existing approaches, explicit data-driven model discovery of high-dimensional dynamical systems remains a formidable challenge, owing not only to their intrinsic high dimensionality but also to the trade-off between interpretability and scalability, as schematically summarized in Fig. 1.

In nonlinear dynamics, the problem of discovering systems from data was recognized quite early, when an information-theoretic method was proposed and tested on low-dimensional chaotic maps [5]. Another early approach focused on the approximation of a nonlinear system by a set of linear equations in different regions of the phase space [6]. Other "classical" approaches include those based on chaotic synchronization [7] and least-squares best approximation [8]. A relatively recent approach is symbolic regression [9,10], which aims to identify the best-fit equation for the input-output relationship of the data by leveraging genetic algorithms [11] to search through an expression space. Usually, the equations are represented as trees, mathematical operators as nodes, and constants/variables as leaves. Recently, methods such as AI-Feynman [12] were introduced to mitigate this difficulty, where the genetic algorithm is enhanced using a divide-and-conquer strategy combined with neural networks (NNs) to identify symmetry, separability, and composability of the underlying system. Similarly, deep symbolic regression [13] was developed, which uses reinforcement learning to expedite the search process. Nevertheless, the applicability of most SR methods has been predominantly restricted to small-scale problems with less than ten input variables.

*Contact author: sixing@calpoly.edu

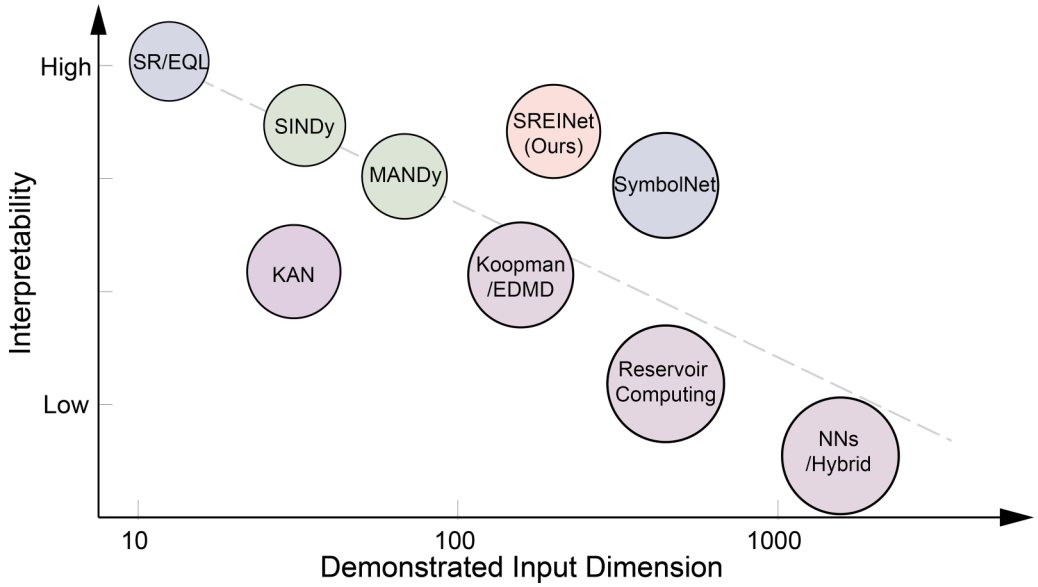


FIG. 1. Schematics of selected equation-discovery methods across input dimension and model interpretability. The dashed line illustrates the general trade-off where higher-dimensional capability often comes at the cost of interpretability. SREINet (ours) aims to maintain interpretability in the high-dimensional regime. Circle size reflects the relative model size.

Another approach is sparse optimization [14,15], where a linear combination of a small number of elementary mathematical functions is used to generate a parsimonious representation of the velocity field or map functions of the system, and the goal is to find the coefficients associated with these functions through optimization methods. Given that only a small subset of elementary functions are relevant in the function library, the number of actual terms or, equivalently, the number of nonzero coefficients, is small, rendering effective sparse optimization. The idea was first demonstrated in 2011 [14] using compressive sensing [16] and the later work such as SINDy (2016) [17] was based on the same sparsity assumption. The sparse-optimization approach, by its name, is most effective for dynamical systems with a sparse equation structure. However, the approach can struggle for systems that violate the sparsity condition, such as the Ikeda or optical-cavity map [18]. Similar to symbolic regression, sparse optimization also suffers from the curse of dimensionality, where the number of candidate functions grows combinatorially with the increase of dimensions. Although recent advances like MANDy [19] reduce memory consumption using tensor decomposition, they still face significant computational challenges with large data volumes. For these reasons, many reported sparse-optimization demonstrations have focused on systems with dozens of dimensions or fewer.

For systems with hundreds of dimensions or more, machine-learning and related data-driven lifting approaches have become important routes for implicitly modeling complex dynamical systems. The pioneering work in Ref. [20] demonstrated the potential of NNs in learning both continuous and discrete forms of the vector field associated with differential equations, i.e., the rate functions. Architectures such as deep neural networks [21], recurrent neural networks [22], neural ODEs [23], and reservoir computing [24–26] have been developed for modeling dynamical systems. A recent work [27] proposed a hybrid approach that integrates autoencoders and sparse regression to simultaneously discover the nonlinear mapping and the latent space. Similarly, graph neural networks combined with symbolic regression have been used to extract symbolic models from learned representations, including applications to cosmological simulations [28]. Related lifting approaches based on the Koopman operator [29] offer a method to linearize the dynamics by lifting the state variables into an infinite-dimensional Hilbert space. Finite-dimensional approximations [30] that are computationally tractable can be achieved through selected basis functions or observables. While these methods can achieve high predictive accuracy, their learned representations often depend on black-box architectures, latent variables, or prescribed observables, making it difficult to recover the underlying physical laws.

To bridge the gap between scalability of NNs and interpretability of discovered equations, a recent line of work has focused on developing symbolic/interpretable neural architectures for data-driven discovery of dynamical systems. For example, equation learner (EQL) [31] represents symbolic expressions using an interpretable feedforward network, with activation functions replaced by atomic operations such as identity, cos, sin, and $\times$. This representation enables applying sparse optimization to find the best-fit symbolic expressions, bypassing expensive searching algorithms. SymbolNet [32] further advances this direction by introducing a dynamic pruning scheme for symbolic neural networks, substantially improving scalability and demonstrating high-dimensional symbolic learning on benchmark datasets such as MNIST. In addition, Kolmogorov-Arnold networks (KANs) [33,34], motivated by the classical Kolmogorov-Arnold theorem [35,36], have recently been applied to data-driven model discovery of nonlinear dynamical systems [37]. By representing multivariate functions through learnable univariate components, KANs can yield compact and interpretable models that go

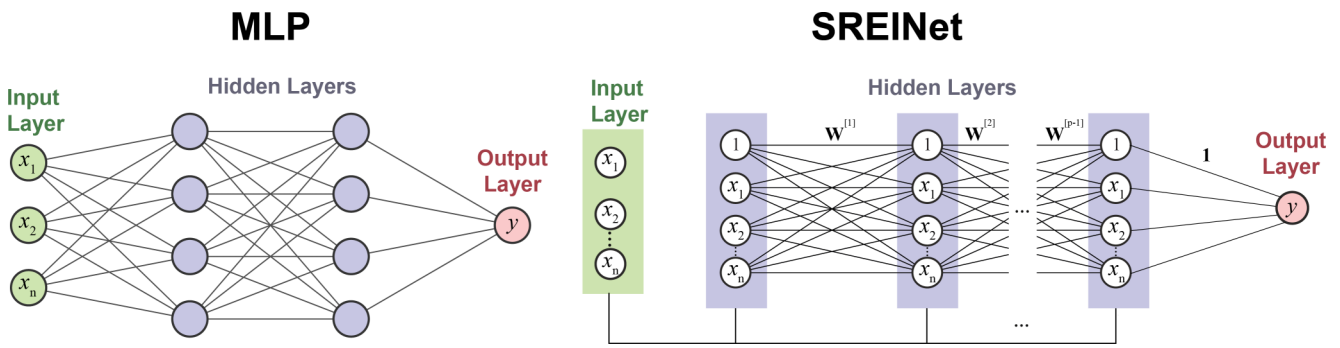


FIG. 2. Structure of MLP (multilayer perceptron, left) in comparison with the proposed SREINet architecture (right). The MLP passes outputs sequentially from one layer to the next, while the proposed architecture directly shares input with all the hidden layers. For simplicity, the activation functions are represented as a constant value of one and the monomial of the input. Each hidden layer generates its output by multiplying its weight matrix with the activation functions, leading to forward propagation summing all possible combinations of multivariate monomials up to a depth-equivalent order. The proposed NN architecture is interpretable as the contribution of each weight to the output is understandable.

beyond the sparsity assumption [37]. However, KANs typically provide interpretable function approximators rather than explicit governing equations of the underlying system. Another recent attempt [38] to use interpretable networks for equation discovery has shown promise but is limited to moderate-scale systems due to computational complexity and training difficulties.

To overcome the scaling barrier, this paper presents a general machine-learning framework specifically tailored to high-dimensional dynamical systems, which reduces the computational cost to $\mathcal{O}(pn^2 m_b)$, where $p$ is the order of nonlinearity, $n$ is the dimension of the system, and $m_b$ is minibatch size. This framework can also be extended to support nonseparable atomic candidate functions, enhancing its representational capacity. The NN architecture is paired with a new sparsity-promoting pruning scheme based on an $S$-shaped periodic schedule that scales the method to systems with hundreds of dimensions, a regime previously unattainable. We establish the theoretical foundation of this network by proving that any arbitrary function expressible as a linear combination of separable multivariate functions can be exactly represented by this network. We name the framework SREINet (sparse regression embedded interpretable network), as it represents a conceptual shift: It embeds the principles of sparse regression within a machine-learning framework, bridging the interpretability of traditional matrix-based sparse regression with the computational advantages of neural-network training, allowing for tapping into the power of automatic differentiation, stochastic gradient descent, and end-to-end optimization.

We apply SREINet to four paradigmatic spatially discretized spatiotemporal dynamical systems and two complex networks with nonlocal connections to demonstrate its general applicability and scalability in regimes that are difficult for conventional explicit-library sparse-optimization methods under comparable computational constraints. The learned governing equations are capable of accurately extrapolating physically observable phenomena in spatiotemporal systems such as coherent structures. This represents a significant improvement over traditional network-based approaches that are limited in their extrapolation capabilities. Our framework is robust against intermittent noise and incomplete data, being able to maintain high accuracy across a wide range of dimensions. It also demonstrates strong performance when tested on an empirical dataset from a triple-pendulum system. When sufficient data are available, the computational cost remains consistent as the data volume increases, making it a suitable choice for tackling large-scale problems involving big data. We further provide head-to-head comparisons against representative equation-discovery and predictive baselines, quantitative analyses of computational efficiency through memory and run-time scaling versus matrix-based baselines, and training-dynamics analyses across representative systems to show how periodic pruning stabilizes convergence under different optimization landscapes. Our scalable, equation-finding SREINet provides an effective data-driven approach for probing and understanding large-scale complex dynamical systems arising from different fields of science and engineering.

## II. METHODS

We focus on high-dimensional dynamical systems represented by coupled ODEs or nonlinear PDEs that are cast as coupled ODE systems upon spatial discretization. In either case, the high-dimensional dynamical system has the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{R}^n, \tag{1}$$

and the aim is to learn the governing equations from a dataset $\{\boldsymbol{x}_i, \dot{\boldsymbol{x}}_i\}_{i=1}^m$, where the derivatives can be estimated by, e.g., a finite difference approximation. The vector field can be represented as the linear combination of a library of candidate functions in the form $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\xi}\boldsymbol{\Theta}(\boldsymbol{x})$, where $\boldsymbol{\xi}$ denotes an $n$-by-$r$ sparse coefficient matrix and $\boldsymbol{\Theta}(\boldsymbol{x})$ represents a vector with $r$ numbers of the candidate functions. Going beyond the previous method [14,17], we embed the matrix representation into the following interpretable neural-network structure, as illustrated in Fig. 2:

$$\mathcal{N}_i(\boldsymbol{x}) = \boldsymbol{\xi}_i\boldsymbol{\Theta}(\boldsymbol{x}), \quad i = 1, 2, \ldots, n, \tag{2}$$

where $\boldsymbol{\xi}_i$ denotes the $i$th row of the coefficient matrix, which can be reconstructed from the neural weights. This

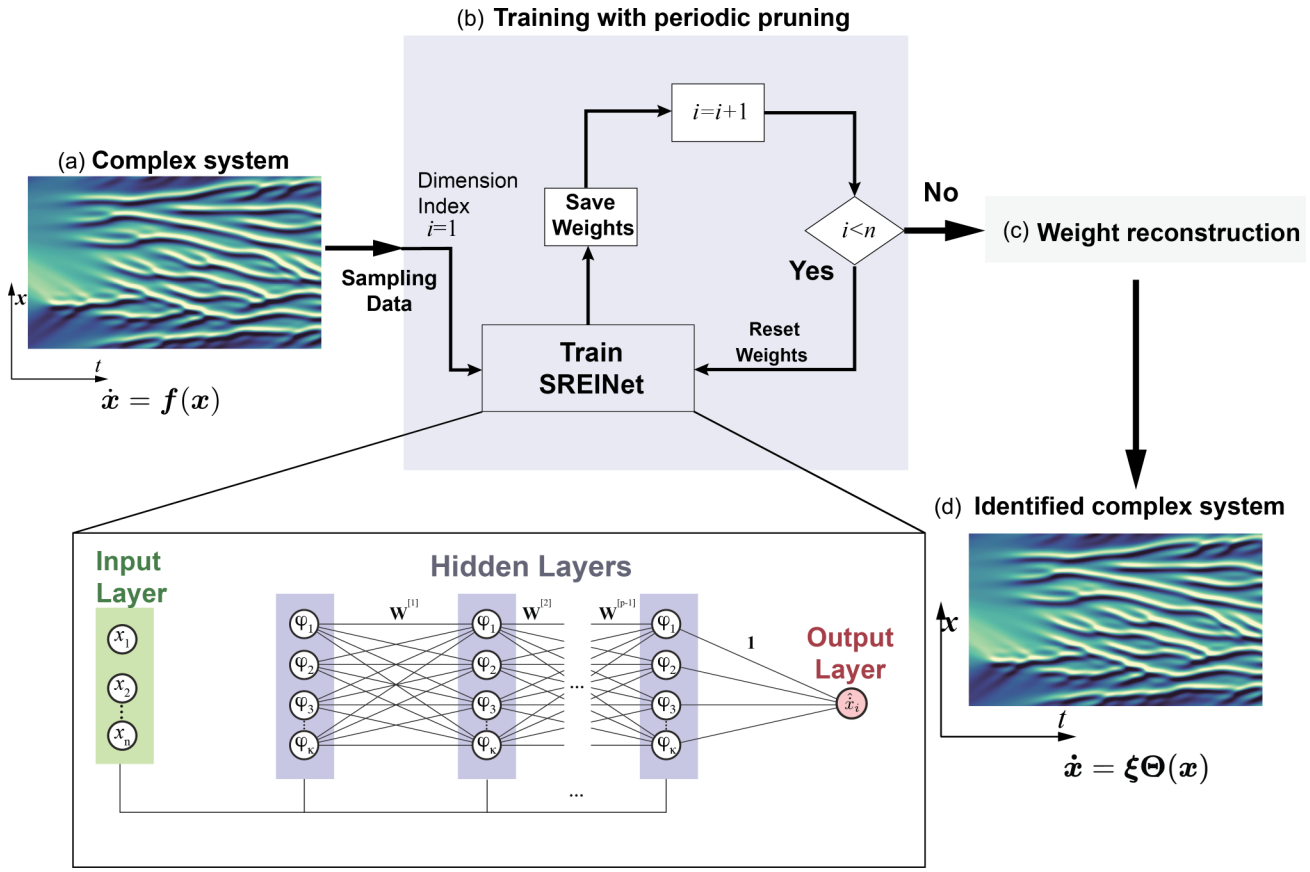


FIG. 3. Schematic illustration of how SREINet is trained and deployed to find the equations of high-dimensional dynamical systems. The training algorithm separately identifies the system equation of each dimension.

neural-network architecture is interpretable because it has no composition of activation functions: Its neurons are the univariate functions resulting from the decomposition of multivariate candidate functions $\boldsymbol{\Theta}(\boldsymbol{x})$ (e.g., $x_1^2, x_2x_3^2$). As a result, forward propagation leads to an output summing a chain of univariate functions that can recover the multivariate candidate functions as well as their weights, which will be explained further subsequently.

*Structure of SREINet*. We now present a generalized structure of SREINet, shown at the left bottom of Fig. 3. The input to SREINet is the state variables $\boldsymbol{x} = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, and the output includes the time derivative $\dot{x}_i$ associated with a specific dimension. We use this neural architecture to learn the vector field of each dimension separately. The activation functions of a layer are univariate candidate functions in the form

$$\boldsymbol{\Phi}^{[j]} = \left[\varphi_1^{[j]}, \varphi_2^{[j]}, \ldots, \varphi_k^{[j]}\right]^T \ (k > n), \tag{3}$$

where

$$\varphi_i^{[j]} \in \mathcal{G}, \quad \mathcal{G} = \{g_\ell : \mathbb{R} \to \mathbb{R}\}_{\ell=1}^{q}, \tag{4}$$

where $\mathcal{G}$ is a user-defined library of univariate atomic candidate functions. Each $\varphi_i^{[j]}$ is evaluated at an input feature $\zeta \in \{x_1(t), x_2(t), \ldots, x_n(t)\}$. The choice of $\mathcal{G}$ is problem dependent and is not fixed by the SREINet architecture. For polynomial systems, we use constant and identity functions, whereas trigonometric functions such as $\sin(r\zeta)$ and $\cos(r\zeta)$ are included only when motivated by the expected dynamics, as in systems with angular variables. Other analytic functions may also be included when appropriate. To enhance the expressive power, neurons with repeated candidate functions are allowed (see Theorem 1). As opposed to a multilayer perceptron, these activation functions share the same input from the input layer and are then multiplied elementwise by the weighted output of the previous layer. More specifically, the output of the $j$th hidden layer is

$$\boldsymbol{y}^{[j]} = \boldsymbol{\Phi}^{[j]}(\boldsymbol{x}) \odot \boldsymbol{W}^{[j-1]} \cdot \boldsymbol{y}^{[j-1]}, \tag{5}$$

where $\boldsymbol{W}^{[j-1]}$ denotes the weights of the $(j-1)$th hidden layer, $\cdot$ denotes matrix multiplication, and $\odot$ indicates the elementwise Hadamard product. The output layer of SREINet is a unit-weight dense layer, aggregating the output of the last hidden layer. The SREINet output is thus given by

$$\begin{aligned}\hat{\dot{x}}_i = \mathcal{N}(\boldsymbol{x}) = \mathbf{1}_{1\times k} \cdot \boldsymbol{\Phi}^{[p]}(\boldsymbol{x}) \odot \boldsymbol{W}^{[p-1]} \cdots \boldsymbol{W}^{[2]} \\ \cdot \boldsymbol{\Phi}^{[2]}(\boldsymbol{x}) \odot \boldsymbol{W}^{[1]} \cdot \boldsymbol{\Phi}^{[1]}(\boldsymbol{x}),\end{aligned} \tag{6}$$

where the operations are evaluated from right to left. In addition, $\mathbf{1}_{1\times k}$ represents the fixed weights of the output layer, and $p$ is the depth of SREINet, equivalent to the highest order of nonlinearity. As Eq. (6) does not contain any composition of the activation functions, the explicit form of network output can be written down in terms of the sum of all combinations of the univariate functions up to the $p$th order.

The SREINet structure essentially embeds the matrix formulation of sparse regression, because the right-hand side of

the differential equations can be represented by the superposition of linear and nonlinear terms that are the product of univariate functions. For example, the velocity of the equation $\dot{x}_1 = x_1^2 + x_1x_2x_3^2$ is the addition of two nonlinear terms $x_1^2$ and $x_1x_2x_3^2$, where $x_1^2$ is the product of two $x_1$, and $x_1x_2x_3^2$ is the product of $x_1$, $x_2$ as well as two $x_3$. The velocity field can be reconstructed through the forward propagation of a four-layer SREINet with the activation functions $\mathbf{\Phi} = [1, x_1, x_2, x_3]^T$ in each layer. This can be achieved by manipulating the weights along the paths that yield the two desired terms, while neglecting the others. Indeed, the forward propagation of SREINet can reconstruct all possible combinations of these univariate functions up to the order equal to the depth of the neural architecture. In this sense, the mapping $\mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}_i)$ can naturally be viewed as an embedded representation of $\boldsymbol{\xi}_i\mathbf{\Theta}(\boldsymbol{x})$.

In the following, we prove that SREINet can represent any function that is a linear combination of products of univariate functions, which includes the right-hand side of the governing equations of nonlinear dynamical systems as a special case.

*Theorem 1. (Universal representation of sum of univariate products).* For any expression $f(\boldsymbol{x}) = \sum_{i=1}^{M} a_i\phi_i(\boldsymbol{x})$, where each $\phi_i(\boldsymbol{x})$ is a product of univariate functions (e.g., $x_1^2$ or $x_1x_2^2$), there exists a configuration of SREINet that exactly represents $f(\boldsymbol{x})$.

*Proof.* See Note 1 in the Supplemental Material [39]. ■

*Remark.* The universal representability relies on replicating specific neurons to avoid structural conflicts among paths. Since identifying the exact neurons to repeat is nontrivial *a priori*, we introduce a practical hyperparameter $C$ that uniformly replicates the base neuron set. In practice, a greedy search can find the smallest sufficient $C$ by increasing $C$ until performance converges. A key strength of SREINet lies in its ability to maintain structural efficiency without requiring a large number of copies $C$. Due to the nature of the network, even with $C = 1$, the model can search through all possible combinations of atomic functions up to an order determined by its depth. Since the governing equations in most physical systems are inherently sparse, a small $C$ is normally sufficient to recover the relevant dynamics. However, for systems where the sparsity condition is violated or the exact functional form lies outside the predefined candidate library, the discovered equations become dense and less interpretable. In such scenarios, while the model may not yield a concise symbolic expression, it can still function as an effective data-driven predictor, maintaining its predictive capabilities and exhibiting a degree of extrapolation potential (see the triple-pendulum example in Sec. III).

The main advantage of this network structure is that it reduces the search space for candidate functions from $\mathcal{O}((n+p)!/(n!p!))$ to $\mathcal{O}(pn)$, where $n$ is the system dimension, and $p$ is the order of nonlinearity. As a consequence, the number of parameters to be trained is reduced from $\mathcal{O}((n+p)!/(n!p!))$ to $\mathcal{O}(pn^2)$, which is a significant reduction in computational cost and memory consumption. This reduction is crucial for addressing the curse of dimensionality and making the method scalable to high-dimensional systems. We demonstrate this through further quantitative analysis in Sec. III. In addition, the network structure allows for efficient training using stochastic gradient descent and automatic differentiation, further enhancing its scalability and applicability to large datasets.

Another noteworthy property of SREINet is the nonuniqueness of weight-level optima. A simple pathwise example is $\xi = W_{ij}^{[1]}W_{jk}^{[2]}$. For fixed $\xi$, infinitely many weight pairs satisfy this relation via reciprocal rescaling. This implies that the same coefficient can be realized by infinitely many weight configurations. Consequently, the loss landscape may contain manifolds of equivalent minima rather than isolated minimizers. Such geometry may improve optimization accessibility in practice, although a formal proof is left for future work. Importantly, equation recovery remains unchanged because equivalent weight solutions map to the same reconstructed coefficients.

*Coefficient reconstruction.* We establish a nonlinear mapping connecting an arbitrary coefficient in the matrix formulation with the corresponding weights. For simplicity, we assume that $C = 1$ in the following. We first define a set of indices $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_k]$ with a total degree $|\boldsymbol{\alpha}| \leqslant p$. Next, we denote the associated coefficient of the candidate function $\varphi_1^{\alpha_1}\varphi_2^{\alpha_2}\cdots\varphi_k^{\alpha_k}$ as $\xi_{\boldsymbol{\alpha}}$. The nonlinear mapping reads

$$\xi_{\boldsymbol{\alpha}} = \sum_{\text{cyc}} W_{j_1j_2}^{[1]}W_{j_2j_3}^{[2]}\cdots W_{j_{p-1}j_p}^{[p-1]}, \tag{7}$$

Here, $W_{j_{p-1}j_p}^{[p-1]}$ denotes the weight connecting neuron $j_{p-1}$ in the $(p-1)$th layer to neuron $j_p$ in the $p$th layer. The index set $J = \{j_1, j_2, \ldots, j_p\}$ satisfies $\text{Count}(J, i) = \alpha_i$, where $\text{Count}(J, i)$ denotes the number of times the value $i$ appears among the indices in set $J$. In practice, the coefficients can be reconstructed using a recursive depth-first search algorithm to explore all active (nonzero) paths from the input to the output layer (see Note 3 in the Supplemental Material [39]).

*Inclusion of nonseparable candidate functions.* Although SREINet is built upon the separable property of candidate functions, it can be extended to include nonseparable functions, such as $\sin(x_1x_2)$. Inspired by the lifting operator in extended dynamic mode decomposition [30], we introduce a lifting layer after the input layer (see Fig. 4). This lifting layer defines a nonlinear map $\mathbf{z} = \Psi(\mathbf{x})$ that enables the inclusion of nonseparable atomic candidate functions, which can then be used to form more complex candidate functions through forward propagation. Additionally, the lifting design is modular and scalable: It allows selective inclusion of nonseparable candidate functions guided either by domain knowledge or data-driven cues, thereby avoiding uncontrolled growth of the library. In Note 7 in the Supplemental Material [39], we demonstrate the efficacy of this extended network through a modified Lorenz-96 system. Remarkably, the atomic candidate functions in the lifting layer can potentially be paired with trainable parameters, such as $\sin(\omega x_1x_2)$ or $\exp(-a(x_1-\mu)^2)$, where $\omega$, $a$, and $\mu$ are trainable parameters, though this capability is not demonstrated in this paper because it requires more sophisticated training procedures. In the Sec. III, we focus on the structure where neurons are equipped with only univariate candidate functions to demonstrate the efficacy of SREINet for high-dimensional dynamical systems.

*Training algorithm.* The training algorithm for identifying sparse dynamics of each dimension individually is as follows. The algorithm begins with collecting a dataset $\{\mathcal{X}_i, \dot{\mathcal{X}}_i\}_{i=1}^m$ by

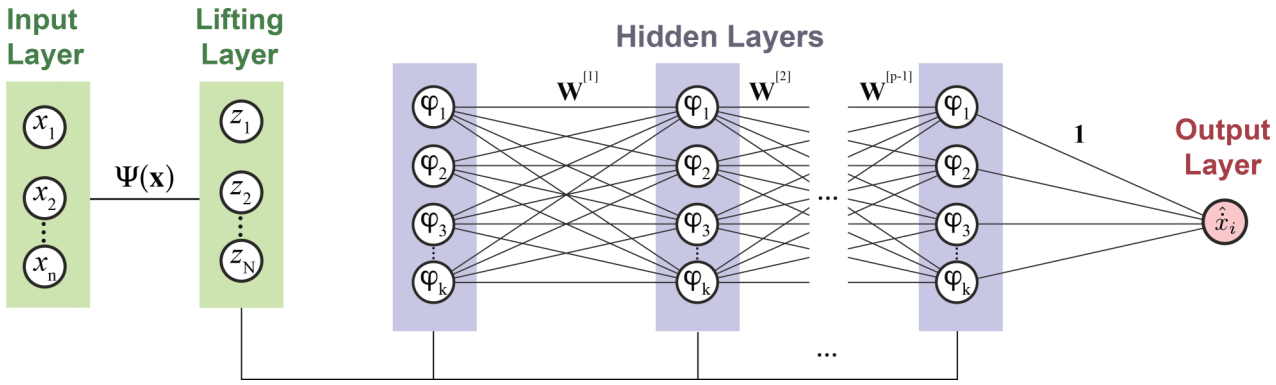


FIG. 4. The SREINet with a lifting layer that maps input coordinates $\boldsymbol{x}$ into a higher-dimensional space $\boldsymbol{z}$. This extension allows SREINet to include nonseparable atomic candidate functions such as $\sin(x_1 x_2)$.

uniformly sampling $m$ points on $N$ ($N = 1, 2, \ldots$) trajectories of the target system. Sampling from single or multiple trajectories is problem dependent, with the primary goal of generating a dataset that captures the full dynamics across all dimensions. The trajectories can be produced either from numerical simulations with random initial conditions or from physical experiments. In our work, we initially sample the derivative values that are needed for assessing the scalability of the algorithm to high-dimensional problems. We then add random noise to the data to assess the algorithm performance. The collected dataset is divided into training and validation sets with an 80-20 ratio.

We apply a dimensionwise scheme to learn the governing equations of spatiotemporal systems with different types of nonlinearity. We start from random initial weights to learn the $j$th dimension vector field

$$\hat{\dot{x}}_j = \hat{\boldsymbol{f}}_j(\boldsymbol{x}) \approx \mathcal{N}(\boldsymbol{x}, \boldsymbol{\theta}_j), \quad j = 1, 2, \ldots, n \tag{8}$$

through the following optimization problem:

$$\arg\min_{\boldsymbol{\theta}_j} \sum_{i=1}^{m} \frac{1}{m} \left\| \dot{x}_j^{(i)} - \mathcal{N}\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}_j\right) \right\|_2^2 + \sum_{k=1}^{p-1} R\left(\boldsymbol{W}_j^{[k]}\right), \tag{9}$$

where $\boldsymbol{\theta}_j$ denotes the corresponding network weights (reset for each dimension) and $R$ represents a sparsity-promoting operator applied to the weights of all layers. Theoretically, the sparsity-promoting operator should be applied to the reconstructed coefficients from Eq. (7). However, the reconstruction will lead to the calculation of $\frac{(n+p)!}{n!p!}$ number of coefficients during each training step, which is not scalable for high-dimensional systems. Therefore, we apply the sparsity-promoting operator to the weights of all layers directly. In addition, the presented iterative training scheme can be implemented in parallel, which shall significantly reduce the training time with parallel computing. Prior to discussing the learning process, we present the implementation of the sparsity-promoting operator.

*S-shaped periodic pruning.* Traditionally, the sparsity-promoting operator is selected from lasso [41] regression, ridge [42] regression, or sequentially thresholded least squares (STLSQ) [17]. Among them, STLSQ appears more robust when dealing with noisy data [17]. However, directly integrating STLSQ into stochastic gradient descent in machine learning presents two challenges. First, the basic data structure (tensor) in machine-learning tools such as TensorFlow is immutable and disallows dynamic resizing. This means implementing STLSQ can only be done through a dynamic computational graph, which can significantly reduce computation efficiency. Second, as opposed to the one-shot computation of pseudoinverse, gradient descent iteratively converges toward the optimal solution. As a result, adopting a hard threshold as STLSQ during gradient descent could inadvertently remove critical terms before their weights exceed the threshold.

The first challenge can be addressed by applying binary masking to filter out the weights that are less than a threshold so that they are excluded from forward and backward propagation [38,43]. To address the second challenge, we extend the adaptive pruning method [38] by constructing an $S$-shaped periodic pruning schedule, as shown in Fig. 5. The scheduler begins with several epochs where the threshold remains at zero or increases at a near-zero rate, providing a starting point for pruning while avoiding convergence to an overfitted coefficient space. The threshold value is then slowly ramped up, followed by a transition to a fast-varying region until it reaches its maximum and then holds there. This schedule can be implemented through various monotonically increasing functions that exhibit an $S$-shaped trajectory, such as sigmoidal functions or piecewise constant mappings integrated with scaled-and-lifted sinusoids. Subsequently, the scheduler zeros the threshold to restart the pruning process. The pruning profile can be repeated multiple times until the conditions for early stopping are satisfied. It turns out that the repetition can significantly improve the robustness of the algorithm (see training dynamics in Sec. III), as adaptive pruning alone for high-dimensional systems may require painstaking fine-tuning that can be especially challenging when different dimensions possess distinct symbolic forms. Without the repetition process, training will lead to a suboptimal/overfitted solution if a critical term is accidentally pruned. The resetting mechanism provides extra opportunities for convergence through restarting from a subspace with weights close to the correct values.

Notably, the masking/gating approach is not new for training symbolic networks. For example, Zhang *et al.* [44] developed a binary gate parametrized by a stochastic

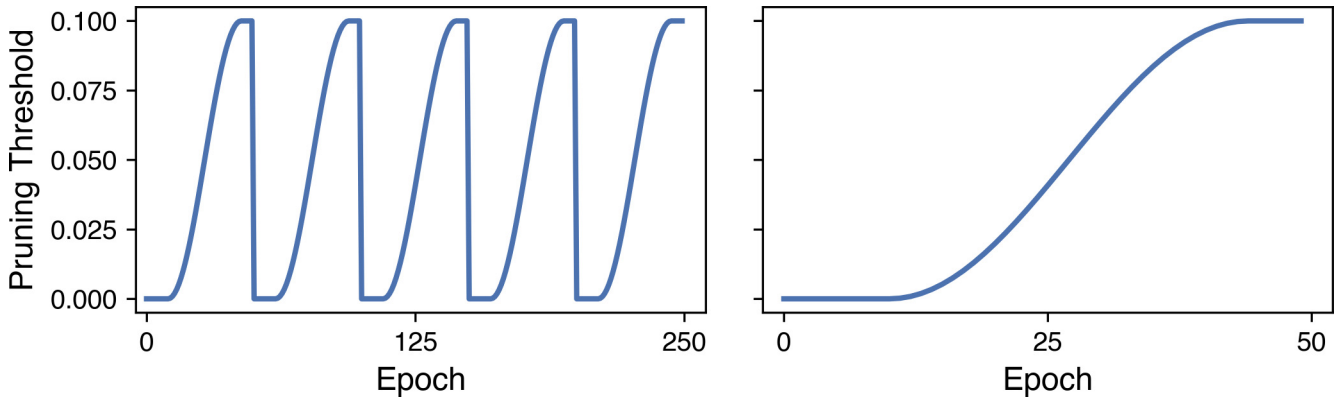


FIG. 5. An example of *S*-shaped periodic pruning scheme. The left panel is an overview of the pruning schedule, while the right panel provides a closer look at a single pruning period.

variable such that it is differentiable and can be dynamically adjusted during training. Tsoi *et al*. [32] proposed SymbolNet, which performs advanced adaptive dynamic pruning of model weights, input features, and mathematical operators within a single training process. Each prunable component is associated with a trainable threshold and a binary step-function mask in the forward pass, while a smooth surrogate gradient based on the sigmoid derivative is used in the backward pass. In addition, SymbolNet introduces sparsity regularization terms for different pruning types, whose strengths are adjusted adaptively to drive the model toward prescribed target sparsity ratios. These approaches address limitations associated with fixed thresholds by allowing the effective sparsity pattern to evolve during training. Our contribution is complementary: Rather than focusing on the differentiable gate itself, we introduce a periodic pruning schedule that repeatedly relaxes and tightens the pruning threshold. In our high-dimensional experiments, this periodic schedule substantially improves training stability and reduces the chance of prematurely removing important terms.

*Training process.* To learn the optimal weights that give best fit to the sample set, we train our SREINet using the Adam optimizer at a fixed learning rate. The trainable weights are initialized randomly with zero mean and standard deviation equal to or less than $10^{-2}$, because the sparsity assumption implies that the majority of weights will diminish. The training is performed sequentially from the first to the last dimension. Take the $i$th dimension as an example. The input to SREINet is the set $\mathcal{X}$, while the output takes the $i$th-dimensional derivative data $\dot{\mathcal{X}}_i$. In each epoch, we update a threshold value given by the scheduler, which will be used to generate binary masks for pruning. By applying the masks to the weight matrices and gradients, we trim the weights below the threshold, preventing them from participating in forward and back propagation. We also implement an early-stopping strategy, which is activated either upon reaching both the minimum loss and target pruning threshold, or when the loss plateaus for a set number of consecutive periods. Upon the completion of training one dimension, we record and reset the weights, proceeding with training the next dimension.

Upon the completion of all dimensions, we obtain the collected weight matrices to reconstruct the coefficients of the original system following Eq. (7). This can be implemented through a recursive algorithm with, again, a pruning threshold to remove the insignificant terms. The pseudocode for our algorithms is provided in Note 3 in the Supplemental Material [39].

*Data sufficiency and coverage.* For data size, we find that the heuristic solution [45] holds for shallow SREINet, which recommends using data points greater than 5–10 times the number of trainable weights. For SREINet with deep layers, significantly more data could be needed. In addition to data size, the coverage of data in phase space is also significant. Ideally, the data should cover sufficiently broad regions such that the dynamics of the system are well represented. If the data only cover linear regions or regions with weak nonlinearity, SREINet is likely to learn a lower-order model (see Note 9 in the Supplemental Material [39]).

*Hyperparameter tuning.* We list the values of several important hyperparameters and the tuning approaches. Regarding the learning rate, we empirically use three levels: 0.1 for datasets with fewer than 5000 points, 0.1–0.01 for datasets with 5000–50 000 points, and 0.001–0.01 for datasets with more than 50 000 points. The selection of minibatch size is partially dependent on the step number per epoch. Ideally, each epoch should contain enough steps for the loss to sufficiently decrease before adjusting the pruning threshold. In most examples, we use a minibatch size of 64 for datasets with fewer than 8000 samples, and 128 for datasets with samples between 10 000 and 60 000. To expedite computations, a larger minibatch size can be considered with more than 100 000 points. For early stopping, the loss threshold can be set between $10^{-8}$ and $10^{-10}$ in the absence of noise, with higher thresholds when noise is present. Similarly, the early-stopping pruning threshold can be set to zero or a small value without noise but adjusted to a slightly higher value with noise.

## III. RESULTS

### A. Identifying governing equations of spatiotemporal systems

Spatiotemporal dynamical systems, when represented by coupled ordinary differential equations, often involve hundreds or even thousands of dimensions. This feature makes them ideal candidates to test the capability of SREINet. We apply SREINet to discover the governing equations of four

TABLE I. Neural-network configurations of SREINet and training results of the six complex systems in subsections A and B. The quantity $L_{\text{mean}}$ represents the average loss value across all dimensions and $|e_{\text{coeff}}|_{\text{mean}}$ denotes the mean absolute error of coefficients. SREINet can precisely recover the coefficients of the governing equations.

| System | Nodes | Dimensions | Network structure | Data size | Batch size | $L_{\text{mean}}$ | $\lvert e_{\text{coeff}}\rvert_{\text{mean}}$ |
|---|---|---|---|---|---|---|---|
| Lorenz-96 | 100 | 100 | [101, 101] | 60 000 | 128 | $8.4 \times 10^{-11}$ | $3.0 \times 10^{-6}$ |
| $\phi^4$ | 50 | 100 | [101, 101, 101] | 60 000 | 128 | $8.0 \times 10^{-12}$ | $1.1 \times 10^{-7}$ |
| DNLS | 50 | 100 | [101, 101, 101] | 160 000 | 512 | $3.9 \times 10^{-12}$ | $8.5 \times 10^{-8}$ |
| AL | 64 | 128 | [129, 129, 129] | 200 000 | 512 | $1.7 \times 10^{-11}$ | $7.4 \times 10^{-7}$ |
| Hindmarsh-Rose | 25 | 75 | [76, 76, 76] | 45 000 | 128 | $8.8 \times 10^{-11}$ | $3.1 \times 10^{-6}$ |
| Kuramoto | 60 | 60 | [121, 121] | 50 000 | 128 | $5.4 \times 10^{-11}$ | $9.9 \times 10^{-7}$ |

paradigmatic spatiotemporal systems: the Lorenz-96 climate model [46] with a quadratic nonlinearity, the discrete $\phi^4$ model [47] with a cubic nonlinearity, the discrete nonlinear Schrödinger equation (DNLS) [48], and the Ablowitz-Ladik (AL) model [49]. The dimensions of these systems (as well as the two in the subsequent subsection), the data volumes, and the neural-network structures are listed in Table I. For each of these systems, we conduct numerical simulations with random initial conditions to generate the datasets for machine learning, where the numerical time derivatives $\dot{\boldsymbol{x}}$ are collected. All training is conducted on a MacBook Pro with Apple M3 chip and 18 GB RAM. The hyperparameter values used for each example are listed in Table S6 in the Supplemental Material [39]. More details on data generation and training are provided in Note 5 in the Supplemental Material [39]. Figure 6(a) shows that SREINet yields the precise vector field of each system, with a consistent level of loss (error) below $10^{-10}$. In fact, all components of the differential equations have been correctly and precisely identified. These tests target a regime that is difficult for conventional explicit-library sparse-regression methods. For instance, for sparse-optimization based on the PySINDy library [50], the algorithm identifies the correct ODEs in only about 75% of dimensions for the Lorenz-96 system, and our PySINDy runs encounter memory limitations for systems with a cubic nonlinearity. While the depth of SREINet in these examples is equal to the highest polynomial order of the systems, the model remains effective even when the depth is further increased (see Note 8 in the Supplemental Material [39]).

*Lorenz-96 climate model*. The model [46] describes the time evolution of temperature along a latitude circle on the earth, which is given by

$$\dot{x}_i = x_{i-1}(Ax_{i+1} - Bx_{i-2}) - Cx_i + F, \quad i = 1, 2, \ldots, n, \tag{10}$$

with the periodic boundary conditions: $x_{-1} = x_{n-1}$, $x_0 = x_n$, and $x_{n+1} = x_1$, where $F$ is a constant forcing that represents external influences such as solar heating and the quadratic terms simulate advection. We divide the latitude circle into 100 sectors, so the dimension of the system is $n = 100$. We generate a dataset with six trajectories originating from random initial conditions, where each trajectory has a duration of $T = 10$ and the sampling time step is $dt = 10^{-3}$. The NN has a two-layer structure with first-order monomial activation functions (plus a function $\Theta = 1$): $\boldsymbol{\Phi}(\boldsymbol{x}) = [1, x_1, x_2, \ldots, x_n]^T$. There are 101 neurons per layer and the total number of trainable weights is $101^2 = 10\,201$. During the training, each dimension takes approximately three pruning periods before early stopping with an average loss of $8.4 \times 10^{-11}$, leading to a mean absolute coefficient error at the level of $10^{-6}$. Figure 6(b) compares the discovered vector field $\dot{x}_i(t)$ with the ground truth with remarkable agreement and consistently small error across all dimensions.

*Discrete $\phi^4$ model*. The spatially discrete $\phi^4$ model is given by

$$\ddot{u}_i = C(u_{i+1} + u_{i-1} - 2u_i) + 2\left(u_i - u_i^3\right),$$
$$u_i := u_i(t) \in \mathbb{R}, \quad i \in \mathbb{Z},\ t \in \mathbb{R}, \tag{11}$$

where $C$ is the coupling constant. The system originates from the continuum limit model [47] in scalar field theories in physics [51] with diverse applications in particle physics, phase transitions, and the Higgs phenomenon [47], where the second-order spatial derivative is replaced by a centered finite-difference scheme with the lattice spacing $h$: $C = 1/h^2$. We set the system dimension to be $n = 50$ and choose $C = 2$ (or equivalently, $h = 1/\sqrt{2}$). In the continuum limit, the homogeneous Neumann boundary conditions are often used in different contexts [47], which correspond to free boundary conditions in the spatially discrete system: $u_{24} = u_{25}$, $u_{-25} = u_{-26}$. For discovering the system, we use a three-layer SREINet with the candidate functions $\boldsymbol{\Phi}(x) = [1, x_{-25}, x_{-24}, \ldots, x_{74}]^T$, where $x_i = u_i$ and $x_{i+n} = \dot{u}_i$, so the number of trainable weights is $2 \times 101^2 = 20\,402$. The training is complete when its loss reaches the low value $8 \times 10^{-12}$. As shown in Fig. 6(c), the coefficients in the system equation can be accurately identified with an (average) absolute error of $10^{-7}$.

*Discrete nonlinear Schrödinger equation and Ablowitz-Ladik model*. The discrete nonlinear Schrödinger equation [48] with a focusing cubic nonlinearity is given by

$$i\dot{u}_j = -\frac{C}{2}(u_{j+1} + u_{j-1} - 2u_j) - |u_j|^2 u_j, \tag{12}$$

where $C = 1/h^2$. It arises in fields such as nonlinear optics and condensed matter physics [52]. Its solution $u_j := u_j(t) \in \mathbb{C}$ describes, e.g., the envelope of the electric field along an optical waveguide array or the quantum-mechanical wavefunction of a Bose-Einstein condensate in an optical lattice. Similar to the $\phi^4$ model as in Eq. (11), the DNLS stems from a second-order accurate, centered finite difference discretization of the one-dimensional focusing yet integrable nonlinear Schrödinger equation (NLS) [53]. To be concrete, we use $C = 4$, $n = 50$, and periodic boundary conditions,

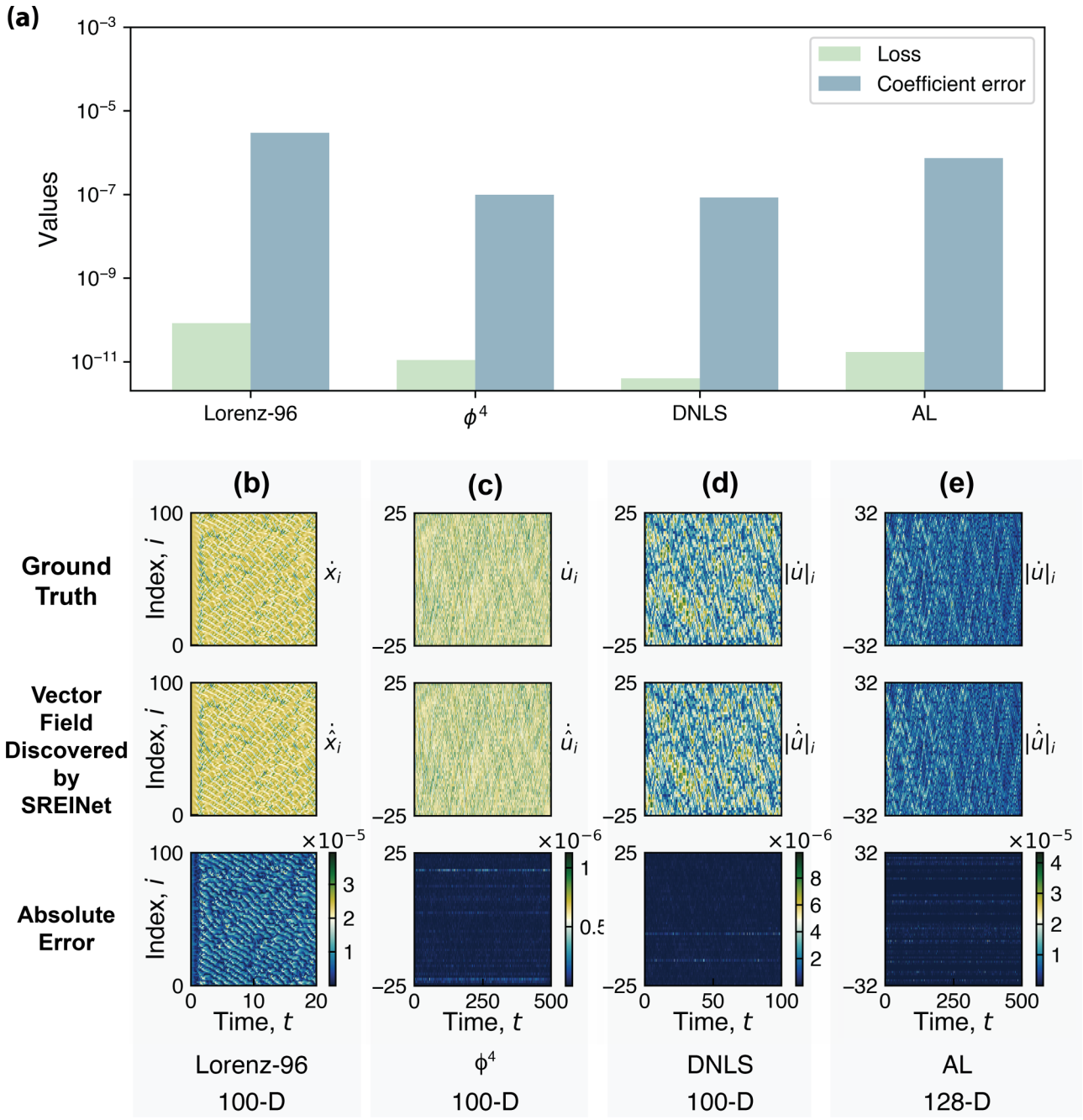


FIG. 6. Demonstration of SREINet's capability for discovering high-dimensional vector fields for four paradigmatic spatiotemporal systems. From left to right: Lorenz-96 model (100-dimensional), discrete $\phi^4$ model (100-dimensional), DNLS system (100-dimensional), and AL system (128-dimensional). (a) Inference loss of vector fields and absolute coefficient errors of the discovered governing equations. (b)–(e) Vector fields discovered by SREINet (middle row) in comparison with the ground truth (top row), with the absolute error of the vector fields in the bottom row.

e.g., $u_{-26} = u_{24}$ and $u_{25} = u_{-25}$. We reorganize the equations by splitting the real and imaginary parts of the dynamical variables: $u_j = \alpha_j + i\beta_j$ for $j = -25, -24, \ldots, 24$, leading to a 100-dimensional input to a three-layer neural-network architecture, where there are 101 neurons per layer with the monomial candidate functions. As shown in Fig. 6(d), SREINet successfully captures the dynamics of the DNLS, achieving a low loss of $3.9 \times 10^{-12}$ with an average error of $8.5 \times 10^{-8}$ in the reconstructed equation coefficients.

The DNLS in Eq. (12) is not integrable; i.e., it does not possess an infinite number of conserved quantities [48], and thus no explicit solutions are available. An integrable discretization of the NLS is the following Ablowitz-Ladik model [49]:

$$i\dot{u}_j = -\tfrac{1}{2}(u_{j+1} - 2u_j + u_{j-1}) - \tfrac{1}{2}|u_j|^2(u_{j+1} + u_{j-1}), \quad (13)$$

which possesses explicit solutions. In fact, a subset of the solutions correspond to extreme events in the underlying

physical system such as rogue waves [54]. We set $C = 1$ and use periodic boundary conditions. The neural network has 64 nodes, corresponding to 128 dimensions (real and imaginary parts). The equation-finding performance of SREINet is demonstrated in Fig. 6(e), where a training loss of $1.7 \times 10^{-11}$ and an average absolute coefficient error of $7.4 \times 10^{-7}$ are achieved.

### B. Learning dynamics of complex networks

As previous models contain only local interactions between nodes, we now show the applicability of SREINet for complex networks that include nonlocal connections.

*Hindmarsh-Rose network.* We first consider a sparse Erdős-Rényi type of homogeneous network, where nodal dynamics are characterized by coupled Hindmarsh-Rose neurons [55]:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i) + \sum_{j=1, j\neq i}^{N} \boldsymbol{C}_{ij}(\boldsymbol{x}_j - \boldsymbol{x}_i) \quad (i = 1, \ldots, N), \tag{14}$$

with

$$\boldsymbol{f}_i(\boldsymbol{x}_i) = \begin{bmatrix} y_i - ax_i^3 + bx_i^2 - z_i + I \\ c - dx_i^2 - y_i \\ r(s(x_i - x_0) - z_i) \end{bmatrix}, \quad \boldsymbol{x}_i = [x_i, y_i, z_i]^T, \tag{15}$$

where $a$, $b$, $c$, $d$, $r$, and $s$ are parameters, $\boldsymbol{C}_{ij}$ is the coupling matrix, $I$ is the external input, and $x_0$ is the equilibrium point. We generate a 25-node Erdős-Rényi network with a connection probability of $p = 0.2$ (resulting in 66 edges) using NetworkX, as shown in Fig. 7(a). This configuration yields a 75-dimensional nonlinear dynamical system.

To identify the dynamics of this complex system, we use a three-layer SREINet with first-order monomial candidate functions, trained with a sigmoid pruning profile, with steepness 20. As the system response exhibits slow and fast timescales, discovering the governing equations is particularly challenging. Following an extended average pruning period, the network captures the vector field with a mean loss of $8.8 \times 10^{-11}$ and coefficients with a mean absolute error of $3.1 \times 10^{-6}$. Furthermore, the identified model faithfully replicates the state evolution of the original system, as shown in Fig. 7(c).

*Kuramoto network.* We then consider a fully connected Kuramoto network [see Fig. 7(b)] with external fields [56], described by

$$\dot{\theta}_i = \omega_i + \frac{K}{n}\sum_{j=1}^{n} \sin(\theta_i - \theta_j) + h\sin(\theta_i), \quad i = 1, 2, \ldots, n, \tag{16}$$

where $\omega_i$ is the natural frequency of the $i$th oscillator, $K$ is the coupling strength, and $h$ is the amplitude of the external force. We set $n = 60$, $K = 2$, and $h = 0.2$, with natural frequencies $\omega_i$ taken evenly from $[-5, 5]$, so that the coupled phase oscillators are not synchronized, and the oscillators oscillate at their natural frequencies $\omega_i$. The neural dynamics in the two-layer SREINet architecture are described by the cosine and sine functions, so effectively there are 121 neurons per layer (the first neuron being the constant one). The total number of trainable weights is $121^2 = 14\,641$. The converged result includes multiple pairs of $\cos^2\theta_i$ and $\sin^2\theta_i$, and a symbolic postprocessing program that performs string manipulations to simplify these pairs of trigonometric functions into constants is used to sum such terms when recovering the governing equations. On average, the training loss is about $5.4 \times 10^{-11}$, indicating accurate recovery of the coefficients with a mean absolute error of about $9.9 \times 10^{-7}$. While the state comparison in Fig. 7(d) shows a high degree of visual agreement, the small constant frequency errors eventually lead to a gradual phase shift. Due to the large scale of the state variables, this discrepancy is not immediately apparent in the heatmap but is clearly quantified in the error evolution plots provided in Note 5 in the Supplemental Material [39].

### C. Learning triple-pendulum dynamics with empirical data

To evaluate SREINet's performance with real-world data, we test it on an empirical dataset collected from a triple-pendulum system [57], as illustrated in Fig. 8. We choose this lumped-parameter dynamical system because many existing datasets for spatiotemporal systems, such as weather data and ECG recordings, lack complete access to all relevant state variables and/or external inputs, both of which are essential for accurate system identification. The triple-pendulum system provides a controlled experimental setting in which all state variables can be either directly measured or reliably estimated through numerical differentiation. However, this system poses significant challenges for sparse identification, as the governing equations contain rational nonlinearities that are difficult to incorporate into candidate functions. To our knowledge, no prior data-driven study has successfully identified the full governing equations of this system from experimental data.

The experimental setup in panel (a) consists of a cart fixed on a rail with three pendulum arms that swing freely from an initial displacement. Angular positions of each arm are measured using optical encoders at a 10 kHz sampling rate, with angular velocities obtained through numerical differentiation. Angular accelerations are estimated using downsampled data processed with Savitzky-Golay smoothed differentiation [58] (see Note 6 in the Supplemental Material [39] for details). Panel (b) shows a representative subset of the training data, displaying the states for the lowest arm over approximately 60 s of oscillatory motion.

We construct a three-layer SREINet with $\boldsymbol{\Phi}(\boldsymbol{x}) = [1, \boldsymbol{x}^T, \cos(\boldsymbol{x})^T, \sin(\boldsymbol{x})^T, \cos(2\boldsymbol{x})^T, \sin(2\boldsymbol{x})^T]^T$, where $\cos(\boldsymbol{x}) = [\cos(x_1), \cos(x_2), \ldots, \cos(x_n)]^T$. The network input is $\boldsymbol{x} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$. This produces a network with 31 neurons per layer, whereas the corresponding matrix formulation would involve $\binom{30+3}{30} = 5456$ candidate functions. Trained with two sampled datasets, the network successfully identifies the governing equations for angular velocities, where the dominant dynamical terms are well represented by the chosen candidate functions. The higher loss values [see panel (c)] for angular accelerations arise from two sources: accumulated errors from numerical differentiation and lack of dominant terms in the candidate function library. Despite these limitations, the identified system effectively predicts the evolution of position, velocity,

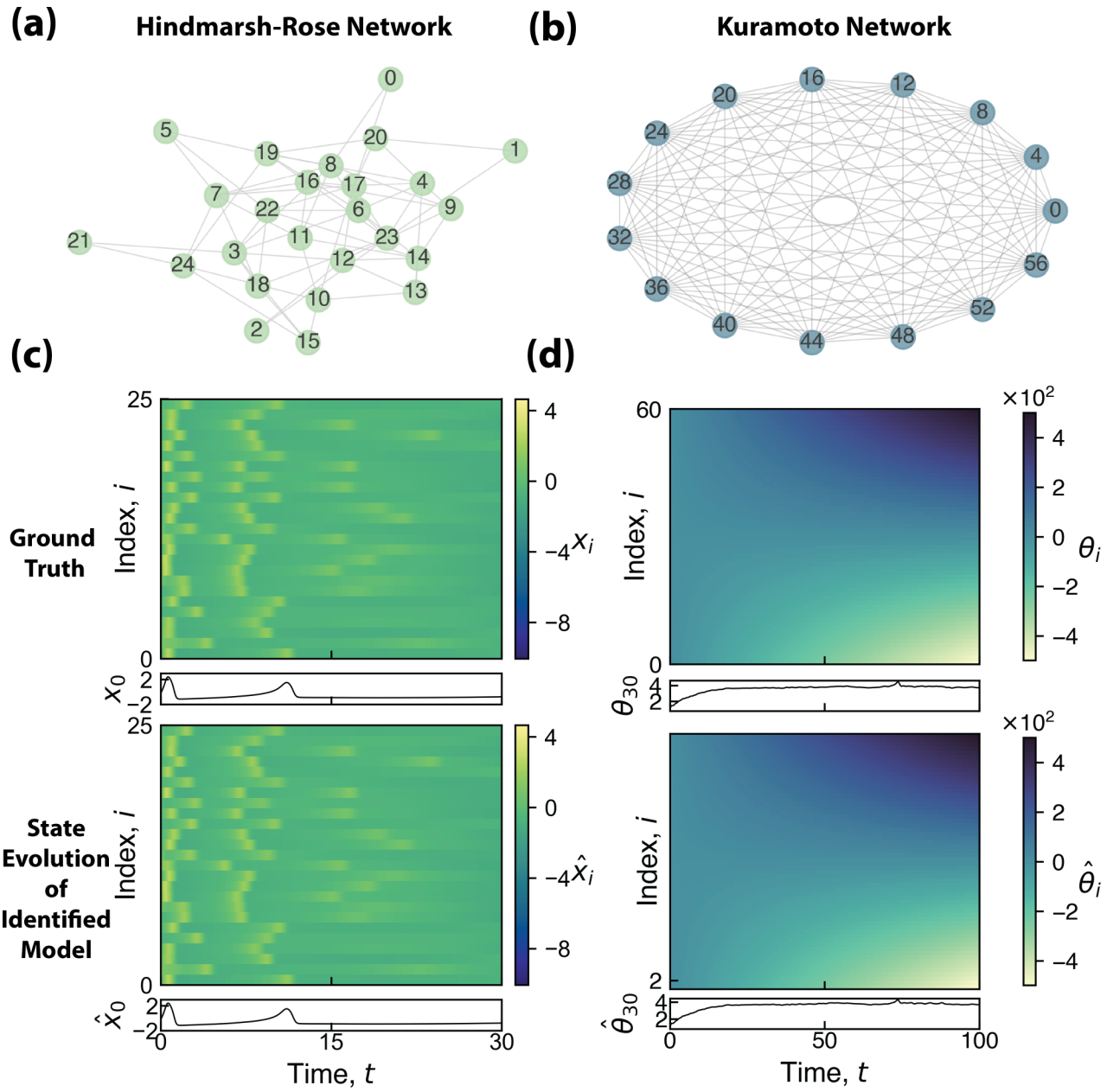


FIG. 7. SREINet can discover dynamics of complex networks with nonlocal connections. Network topology: (a) A 25-node Erdős-Rényi homogeneous network with Hindmarsh-Rose nodes (in total 75 dimensions); (b) a 60-node fully connected Kuramoto network (only nodes with indices that are multiples of 4 are displayed). The first row of panels (c) and (d) shows the ground truth of state evolution, while the second row displays the states predicted by the identified model through SREINet.

and even acceleration over a 10-s period for an untrained validation set, as shown in panel (d).

### D. Extrapolating coherent structures in spatiotemporal dynamics

For traditional neural networks, a challenging problem is to extrapolate physical solutions that are not contained in the training dataset. Once the governing equations of the spatiotemporal system have been uncovered, is the SREINet capable of extrapolating previously unseen solutions or physical phenomena? To address this question, we focus on coherent structures that are fundamental to spatiotemporal dynamical systems. The main results are summarized in Fig. 9, where the contour plots of the spatiotemporal dynamics from the identified systems are shown (right column) in comparison with the ground truth (left column). Specifically, Fig. 9(a) shows a coherent structure from the discrete $\phi^4$ model, where the initial condition is a discrete kink given by

$$u_i(t=0) = \tanh\left(\frac{x_i}{\sqrt{1-v^2}}\right), \tag{17}$$

with $x_i$ denoting the grid points taken uniformly from the range $[-\frac{n}{2\sqrt{C}}, \frac{n}{2\sqrt{C}}]$ with $n = 50$ and $v = 0$. The kink solution is advanced forward in time and travels to the left of the

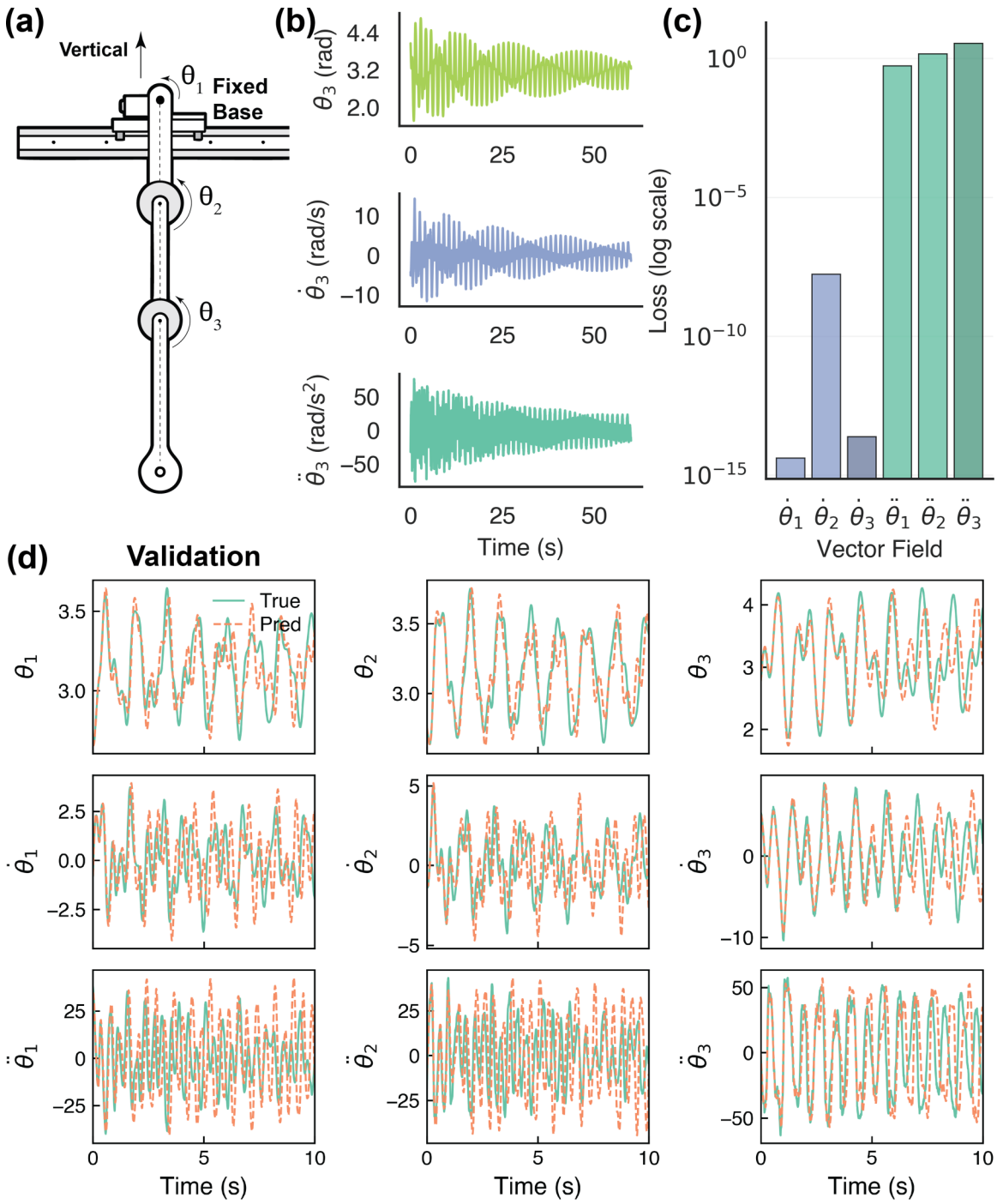


FIG. 8. Evaluation of SREINet on empirical data from a triple-pendulum system. With the selected candidate functions, the network size is equivalent to that of a 30-dimensional dynamical system with polynomial basis functions. (a) Schematic of the experimental setup with a fixed base, where angular positions $\theta_1$, $\theta_2$, and $\theta_3$ are measured from the vertical. (b) A representative subset of the training data, where angular acceleration $\ddot{\theta}_3$ is estimated using smoothed numerical differentiation. (c) Training loss across all dimensions, plotted on a logarithmic scale. (d) Validation results comparing experimental measurements with SREINet predictions. The initial conditions are from an untrained dataset. Strong agreement is observed across all variables and their derivatives over a 10-s segment.

lattice. There is dissipation as Eq. (17) is not an exact solution of the discrete $\phi^4$ system. Remarkably, the spatiotemporal dynamics generated by SREINet agree well with the ground truth. Similar results hold for the DNLS and AL systems, as shown in Figs. 9(b)–9(d), respectively, corresponding to breather solutions [panel (b)], as well as Kuznetsov-Ma (KM) breather [panel (c)] and doubly localized Peregrine breather [panel (d)]. For the DNLS, a Gaussian initial pulse of width $\sigma = 2$ is evolved forward in time, leading to an oscillatory behavior in the waveform in the central site ($n = 0$), i.e., a breathing motion that was reported in Ref. [59]. For the AL system, we used the doubly localized Peregrine breather in

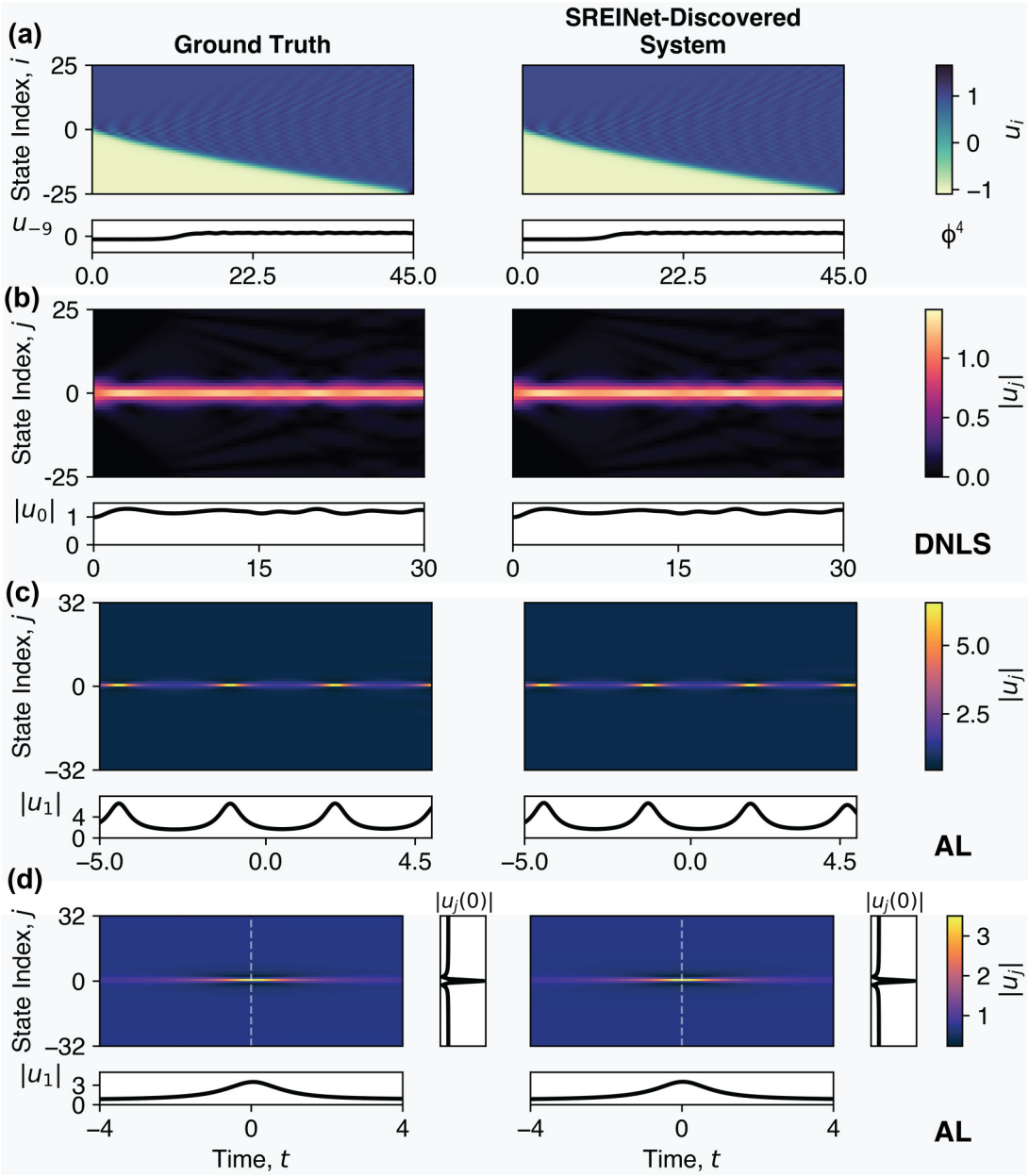


FIG. 9. Numerical results on extrapolation of SREINet-discovered system models emulating coherent structures. (a) Kink solution in the discrete $\phi^4$ model, (b) evolution of a Gaussian pulse in the DNLS equation, (c) KM-breather in the AL model, and (d) Peregrine breather in the AL model. The left and right columns correspond to the ground truth and the waveforms generated by the discovered model, respectively.

Fig. 9(d), given explicitly after rescaling by

$$u_k(t) = \frac{1}{\sqrt{2}}\left[1 - \frac{6(1+it)}{1+2k^2+\frac{3}{2}t^2}\right], \tag{18}$$

and the time-periodic, spatially localized KM breather:

$$u_k(t) = \frac{1}{\sqrt{2}}\frac{\cos\left(\frac{1}{2}\omega t + i\theta\right) + G\cosh(rk)}{\cos\left(\frac{1}{2}\omega t\right) + G\cosh(rk)} \tag{19}$$

in Fig. 9(c) as initial conditions (the values of the parameters $r$, $\theta$, and $G$ are from Ref. [60]). It should be noted in passing that Eq. (19) "breathes" over time with period $T = 4\pi/\omega$ (angular frequency is $\omega/2$) and approaches the Peregrine breather of Eq. (18) in the $T \mapsto \infty$ limit. Both solutions describe rogue waves in nonlinear optics. These results indicate that SREINet-found system model is capable of generating the same spatiotemporal dynamics as the ground truth with the capability of resolving waveform evolution even in nonintegrable systems, demonstrating a remarkable extrapolability of coherent structures in diverse wave systems in physics.

### E. Quantitative and qualitative analysis

*Effect of continuous noise on dynamics discovery.* The examples of successful extrapolation of the coherent structures in Fig. 9 were for noiseless systems. What is the effect of noise on the extrapolation capability of SREINet for system

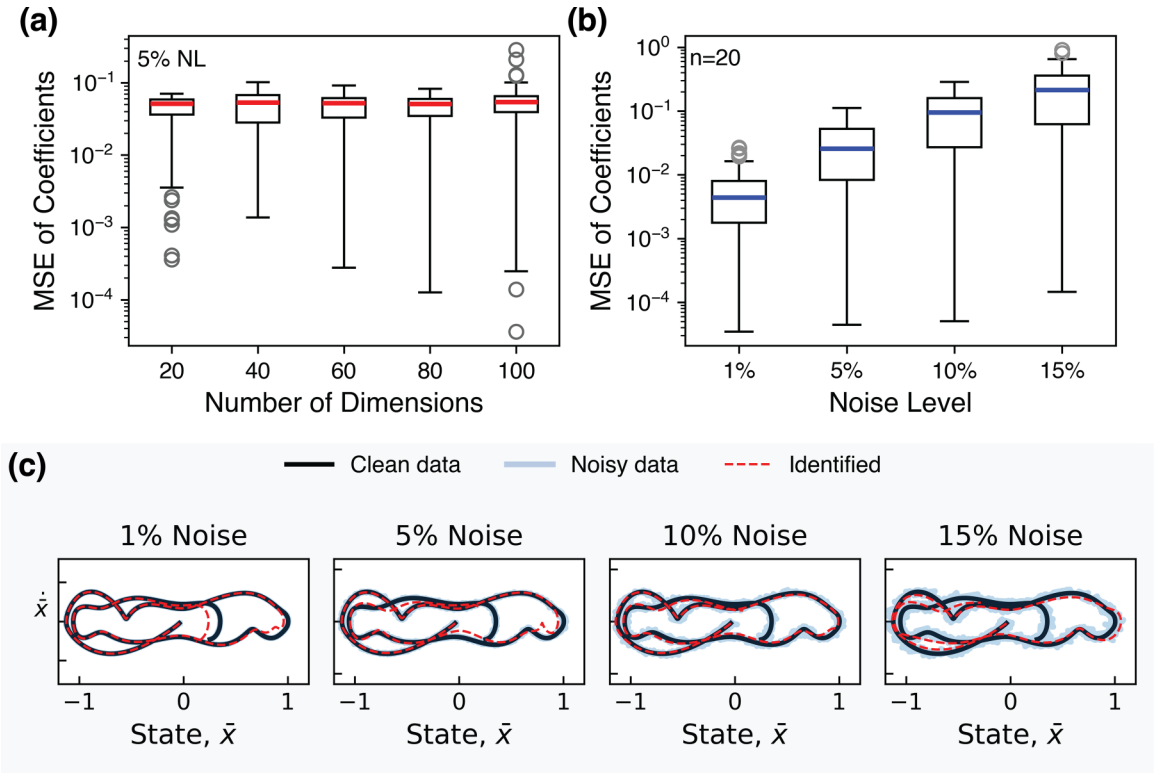


FIG. 10. Effect of white Gaussian noise on the accuracy of model reconstruction. (a) Statistical distribution of MSE of the coefficients vs the number of dimensions for 5% noise level. (b) Distribution of the coefficients MSE vs the noise level for fixed system dimension $n = 20$. In both panels, five independent runs are used to calculate the distribution. (c) Dynamical trajectory of the average state $\bar{x}$ and its derivative $\dot{\bar{x}}$ under different noise levels shown on the phase plane.

discovery? To address this question, we study the $\phi^4$ system with free boundary conditions because its vector field takes different symbolic forms across dimensions. To be systematic, we study the $\phi^4$ system with 10, 20, 30, 40, and 50 nodes, and the underlying dynamical system is 20-, 40-, 60-, 80-, and 100-dimensional, respectively, and add zero-mean Gaussian white noise with a standard deviation equal to 5% of the standard deviation of the sampled data. Figure 10(a) shows that the MSE in the coefficients of the vector field remains approximately constant as the dimension of the system increases, except for a few outliers in the 100-dimensional system, which can be readily removed by slightly varying the hyperparameter values.

To study the effect of noise on the accuracy of the recovered coefficients, we fix the system dimension at 20 (10 nodes) and incorporate white Gaussian noise at levels of 1%, 5%, 10%, and 15% into the data. In particular, at each noise level, five independent runs are performed. In each run, we simulate the $\phi^4$ system with noise added to both $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ during sampling. This setup corresponds to observational noise, where the measurements are corrupted but the underlying deterministic trajectory is not perturbed. The noisy data are the input to the neural-network architecture for the system coefficients to be identified. The resulting statistics of the coefficient error versus noise level are shown in Fig. 10(b), where high accuracy and consistency are maintained for noise levels up to 10%. Figure 10(c) compares the trajectories of the average state variable $\bar{x} = \sum x_i/n$ and its derivative $\dot{\bar{x}}$ of the discovered model with the ground truth. As the noise level increases, the duration over which the trajectories of the identified system precisely match the ground truth decreases. Insofar as the noise level is below 15%, the agreement is reasonable.

*Effects of incomplete data and intermittent noise.* In applications, data are often incomplete due to transmission loss and can be affected by intermittent noise induced by sporadic external disturbances such as signal interference and weather conditions. While interpolation can compensate for missing data to a certain extent, it also introduces additional error. Intermittent noise may lead to a nonzero mean and non-Gaussian distribution, further deteriorating the data quality. These challenges underscore the importance of algorithms that are resilient to incomplete data and intermittent noise.

We evaluate the performance of SREINet for different data missing rates and pollution ratios using the $\phi^4$ system. The top panel of Fig. 11 presents the absolute coefficient ratio of the identified values to the ground truth for 0%, 20%, and 50% missing rates (from left to right), where the corresponding percentage data are randomly sampled and removed from the sample set. Practically, missing data can occur in continuous intervals due to communication failures. This case is not significantly different from random loss for our SREINet framework, as the optimization operates on a point-by-point basis. An exception occurs if the lost interval contains critical dynamics (e.g., fast-varying or high-frequency segments [61]), which may lead to erroneous results. However, we believe this is a matter of data representativeness, rather than

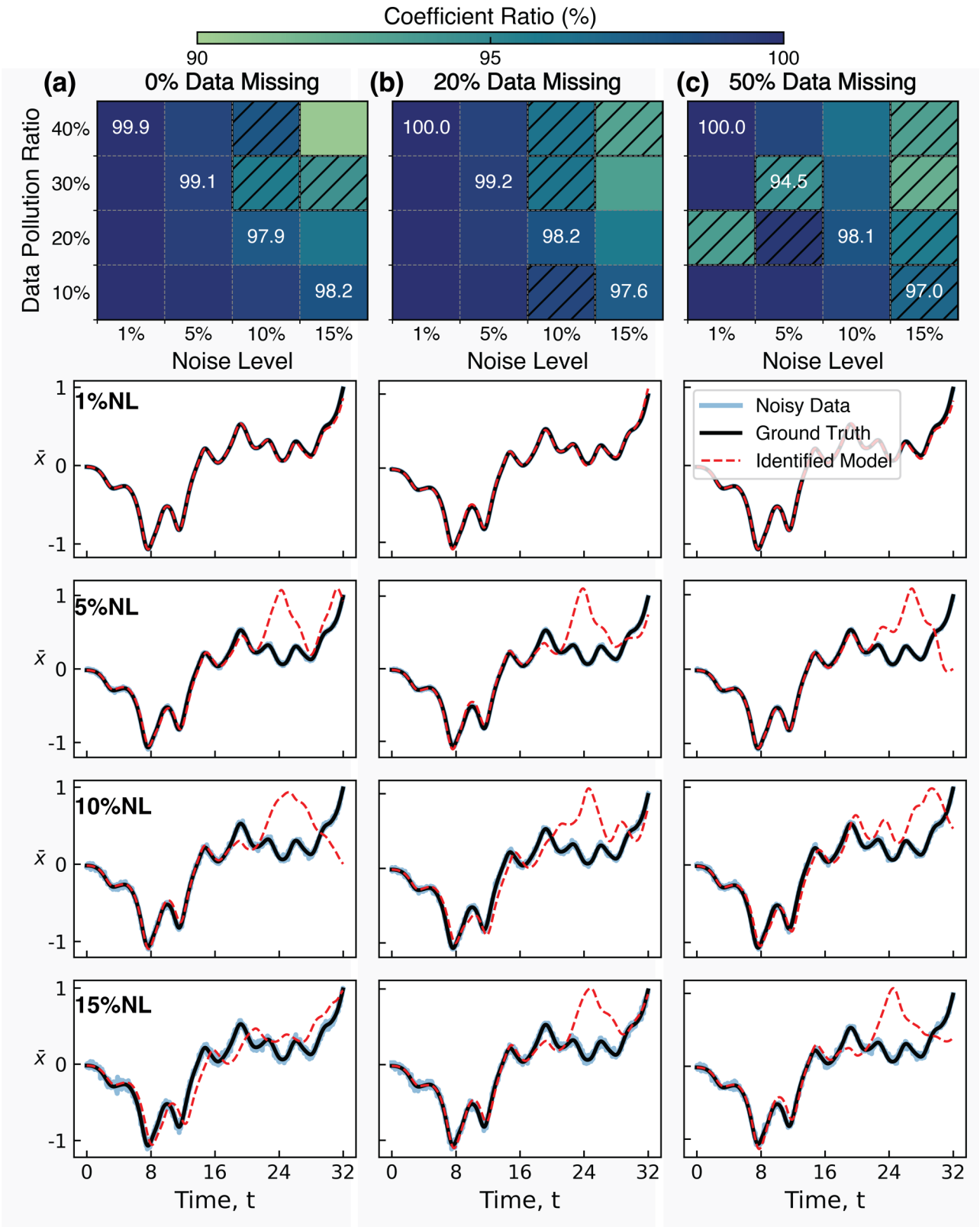


FIG. 11. Effects of incomplete data and intermittent noise on the equation-finding performance of SREINet. The top panel (from left to right) presents the average ratio of the estimated coefficients to the ground truth, for 0%, 20%, and 50% missing rates. Each heatmap illustrates the coefficient ratios across different noise levels and percentages of noisy data. These ratios are calculated by averaging five independent runs. The shaded squares indicate the locations where incorrect equations are identified in at least one run. The time series plots in panels (a)–(c) compare the evolution of the average state $\bar{x}$ of the ground truth with that of the identified models across different missing data rates (with 40% data polluted by noise).

TABLE II. Head-to-head comparison for the clean and noisy $\phi^4$ benchmark. Additional head-to-head comparisons for the Lorenz-96 and Kuramoto benchmarks are reported in Tables 8 and 9 in the Supplemental Material [39]. The entries $\bar{C}/\bar{C}_*$ report the pair of recovered and ground-truth mean term counts. $E_c$ is the mean coefficient root-mean-square error on the ground-truth support, and $E_{\text{roll}}$ is the full-interval rollout normalized root-mean-square error over [0, 2]. Detailed definitions are provided in Note 11 in the Supplemental Material [39]. Reservoir computing is used here as a predictive baseline and does not report symbolic complexity or coefficient error. OOM: out of memory; N/A: metric not defined for the method; –: unavailable result.

| | | Dim. 20 | | | | Dim. 50 | | | | Dim. 100 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Noise | Method | Train loss | $\bar{C}/\bar{C}_*$ | $E_c$ | $E_{\text{roll}}$ | Train loss | $\bar{C}/\bar{C}_*$ | $E_c$ | $E_{\text{roll}}$ | Train loss | $\bar{C}/\bar{C}_*$ | $E_c$ | $E_{\text{roll}}$ |
| Clean | SINDy | $1.94\times10^{-28}$ | 2.3/ 2.3 | $6.48\times10^{-7}$ | $1.02\times10^{-6}$ | OOM | – | – | – | OOM | – | – | – |
| | MANDy | $1.12\times10^{-28}$ | 2.3/ 2.3 | 0.0000 | $1.86\times10^{-16}$ | OOM | – | – | – | OOM | – | – | – |
| | RC | $8.17\times10^{-7}$ | N/A | N/A | 4.7441 | $2.94\times10^{-6}$ | N/A | N/A | 2.1820 | $1.75\times10^{-6}$ | N/A | N/A | 2.0152 |
| | EQL | 0.1263 | 5.5/ 2.3 | 0.8892 | 2.1620 | 0.1206 | 9.8 / 2.4 | 0.8782 | 1.7410 | 0.1715 | 8.2 / 2.5 | 1.1003 | 1.8525 |
| | SymbolNet | 0.0088 | 3.1/ 2.3 | 0.1018 | 0.2785 | 0.0531 | 2.8 / 2.4 | 0.0986 | 0.3066 | 0.0590 | 3.5 / 2.5 | 0.1347 | 0.2680 |
| | SREINet (ours) | $8.67\times10^{-9}$ | 2.3/ 2.3 | $1.41\times10^{-4}$ | $1.93\times10^{-4}$ | $6.18\times10^{-10}$ | 2.4 / 2.4 | $3.31\times10^{-5}$ | $2.83\times10^{-5}$ | $6.44\times10^{-9}$ | 2.5 / 2.5 | $1.05\times10^{-4}$ | $1.13\times10^{-4}$ |
| 1% | SINDy | 0.0018 | 2.3/ 2.3 | 0.0022 | 0.0037 | OOM | – | – | – | OOM | – | – | – |
| | MANDy | $8.27\times10^{-4}$ | 394.8 / 2.3 | 0.5057 | 0.2619 | OOM | – | – | – | OOM | – | – | – |
| | RC | $7.80\times10^{-5}$ | N/A | N/A | 4.7664 | $9.09\times10^{-5}$ | N/A | N/A | 2.0352 | $9.05\times10^{-5}$ | N/A | N/A | 2.0171 |
| | EQL | 0.1280 | 5.5/ 2.3 | 0.8842 | 1.8843 | 0.1233 | 9.6 / 2.4 | 0.8889 | – | 0.1738 | 7.9 / 2.5 | 1.1018 | 1.8553 |
| | SymbolNet | 0.0020 | 3.9/ 2.3 | 0.0190 | 0.0090 | 0.0388 | 3.7 / 2.4 | 0.1450 | 0.3909 | 0.0565 | 3.9 / 2.5 | 0.1290 | 0.5073 |
| | SREINet (ours) | 0.0018 | 2.3/ 2.3 | 0.0054 | 0.0218 | 0.0020 | 2.4 / 2.4 | 0.0049 | 0.0134 | 0.0023 | 2.5 / 2.5 | 0.0074 | 0.0262 |
| 5% | SINDy | 0.0437 | 2.3/ 2.3 | 0.0503 | 0.0931 | OOM | – | – | – | OOM | – | – | – |
| | MANDy | 0.0160 | 424.5 / 2.3 | 0.7543 | 0.5637 | OOM | – | – | – | OOM | – | – | – |
| | RC | 0.0017 | N/A | N/A | 4.7079 | 0.0018 | N/A | N/A | 2.1411 | 0.0019 | N/A | N/A | 2.0535 |
| | EQL | 0.1665 | 5.2/ 2.3 | 0.9188 | 1.8395 | 0.1764 | 9.1 / 2.4 | 0.8520 | 1.5629 | 0.2157 | 7.2 / 2.5 | 1.1031 | 1.7323 |
| | SymbolNet | 0.0999 | 6.0/ 2.3 | 0.2167 | 1.3745 | 0.0552 | 5.1 / 2.4 | 0.1129 | 0.2485 | 0.1613 | 5.9 / 2.5 | 0.2360 | 0.7367 |
| | SREINet (ours) | 0.0407 | 2.3/ 2.3 | 0.0525 | 0.1016 | 0.0488 | 2.4 / 2.4 | 0.0437 | 0.0709 | 0.0553 | 2.5 / 2.5 | 0.0456 | 0.0815 |
| 10% | SINDy | 0.1529 | 6.9/ 2.3 | 0.3355 | 0.0848 | OOM | – | – | – | OOM | – | – | – |
| | MANDy | 0.0534 | 382.2 / 2.3 | 0.6900 | 0.6263 | OOM | – | – | – | OOM | – | – | – |
| | RC | 0.0065 | N/A | N/A | 4.6061 | 0.0071 | N/A | N/A | 2.0769 | 0.0074 | N/A | N/A | 2.0504 |
| | EQL | 0.2546 | 5.7/ 2.3 | 0.9422 | 1.3317 | 0.3000 | 6.4 / 2.4 | 0.9033 | 1.6347 | 0.3459 | 6.0 / 2.5 | 1.1451 | 1.8630 |
| | SymbolNet | 0.2645 | 6.7/ 2.3 | 0.4496 | 0.8999 | 0.2394 | 6.1 / 2.4 | 0.2545 | 0.3812 | 0.2524 | 5.0 / 2.5 | 0.2772 | 0.6230 |
| | SREINet (ours) | 0.1406 | 2.3/ 2.3 | 0.1812 | 0.3404 | 0.1684 | 2.4 / 2.4 | 0.1486 | 0.2042 | 0.1619 | 2.5 / 2.5 | 0.1409 | 0.1808 |

algorithmic deficiency. In Fig. 11, each heatmap colors the coefficient ratios varying with noise (contamination) ratio and noise level, with results averaged over five consecutive runs. The shaded boxes indicate at least one run incorrectly identifying the differential equations, typically occurring in one or two dimensions, where some extra dominant terms are discovered. Generally, the coefficient ratio decreases as the noise level and percentage increase, which also correlates with a higher rate of incorrect equation identification. While the missing rate does not significantly affect the coefficient ratio, it increases the chances of incorrectly identifying the equations. Shown below the top panel are the time series of the average state $\bar{x}$ at four different noise levels for each missing rate, with noise ratio (polluted rate) fixed at 40%. While the prediction horizon aligned with the ground truth decreases as the noise level increases, the framework remains effective for short-term predictions up to 15% noise level, demonstrating robustness of SREINet against incomplete data and intermittent noise.

*Head-to-head comparison.* Having demonstrated the performance of SREINet with noisy data, we now perform head-to-head comparisons with representative methods for dynamics discovery, including both explicit methods (e.g., SINDy and its variant MANDy, EQL and its variant SymbolNet) and an implicit method (e.g., reservoir computing). We note that our goal is not to establish a leaderboard of these methods with perfectly tuned hyperparameters, but to provide a practical comparison based on reasonable hyperparameter-tuning efforts. We select three examples: the $\phi^4$, Lorenz-96, and Kuramoto systems. The $\phi^4$ comparison is presented in Table II, while the Lorenz-96 and Kuramoto comparisons are reported in Tables S8 and S9 in the Supplemental Material [39], respectively. The comparisons concentrate on training loss, recovered symbolic complexity and coefficient error on the ground-truth terms (for methods that yield explicit equations), and rollout error of a trajectory with untrained initial conditions for extrapolation. The details of the hyperparameter-tuning procedure are documented in Note 11 in the Supplemental Material [39].

As shown in Table II, SINDy and MANDy can exactly recover the governing equations for the clean 20-dimensional $\phi^4$ system, achieving the correct symbolic complexity and nearly zero training and rollout errors. However, both methods suffer from OOM issues when scaled to 50 and 100 dimensions in this benchmark. For SINDy, this is due to the combinatorial increase in the number of candidate functions. For MANDy, although the tensor-train representation reduces the explicit-library size, the algorithm still scales poorly with data volume

and dimension in the present setting. In addition, under noisy data, MANDy tends to recover substantially more terms than the ground truth, yielding overfitted governing equations with much higher symbolic complexity. SINDy remains accurate at relatively low noise for the 20-dimensional case, but when the noise level increases to 10%, it also begins to introduce redundant terms.

Reservoir computing (RC) provides a useful predictive baseline. Under the present hyperparameter-tuning protocol, RC can achieve relatively low training loss, especially in noisy cases. However, it does not provide symbolic governing equations or coefficient estimates, and its rollout errors on trajectories with untrained initial conditions are larger than those of the successful equation-discovery methods in the $\phi^4$ benchmark.

The learning-based symbolic methods show a different behavior. Under the present benchmark protocol, EQL is capable of producing sparse symbolic models, but it often converges to suboptimal expressions, leading to larger training losses, coefficient errors, and rollout errors. With its dynamic pruning scheme, SymbolNet is more robust and noise tolerant than EQL, and it yields substantially smaller losses and rollout errors in many cases. However, under the same protocol, SymbolNet can still recover models with higher symbolic complexity than the ground truth as the noise level increases.

SREINet balances fitting accuracy and symbolic parsimony in the $\phi^4$ benchmark. Across all tested dimensions and noise levels, it recovers the ground-truth symbolic complexity and maintains relatively small coefficient and rollout errors. The Lorenz-96 and Kuramoto results in Tables S8 and S9 in the Supplemental Material [39] further show that the relative performance is system dependent: SINDy can remain highly competitive when the candidate library is computationally manageable. However, its performance is sensitive to threshold selection. When the true coefficients are relatively large, a higher threshold can remove many noise-induced terms and improve equation recovery. For systems with small coefficients, such as high-dimensional Kuramoto where the coupling coefficients scale as $K/n$, a more conservative threshold is needed to avoid eliminating true coupling terms. This conservative thresholding can increase the risk of retaining noise-induced redundant terms, as reflected by the higher recovered symbolic complexity observed for SINDy in the noisy Kuramoto cases. In contrast, SREINet can use a more aggressive periodic pruning schedule because pruned terms can be reactivated in later periods. Overall, the main advantage of SREINet is not simply lower training loss, but scalable equation recovery with controlled symbolic complexity and reliable coefficient identification in high-dimensional noisy settings.

*Data scalability.* The head-to-head comparison has shown that accurate equation recovery in high dimensions must be considered together with computational scalability. We therefore evaluate the wall-clock time of the explicit methods for high-dimensional systems (SREINet, MANDy, EQL, and SymbolNet) as the number of training data points increases. SINDy is omitted from this timing-scaling plot because its run-time on the ten-node Fermi-Pasta-Ulam (FPU) example is too small to yield a meaningful data-size scaling curve; its main scalability bottleneck is the dimension-dependent and nonlinearity-dependent growth of the candidate library, discussed separately in the memory analysis.

We test the methods using a ten-node Fermi-Pasta-Ulam-Tsingou nonlinear lattice [62], a prototype system in MANDy's code library. For SREINet, EQL, and SymbolNet, we use the same hyperparameter choices as in the head-to-head comparison for the FPU system, except for the minibatch size and the number of epochs. The minibatch size is set to 64 for SREINet and 512 for EQL and SymbolNet, because the latter two methods train more stably with larger minibatches. Since a larger minibatch yields fewer parameter-update steps per epoch for a fixed dataset, we allow EQL and SymbolNet to run for more epochs so that their losses have sufficient opportunity to decrease. This adjustment is intended to provide each learning-based method with a reasonable optimization budget rather than to exactly equalize the number of gradient updates; in fact, the total number of update steps for EQL and SymbolNet remains smaller than that used for SREINet.

As shown in Fig. 12, the computational time of MANDy increases rapidly with the number of data points. This is because although MANDy uses memory-efficient tensor-train representation [63], it relies on pseudoinversion of tensor-train format to approximate the coefficients, which requires two additional expensive steps: matricization and singular value decomposition of the tensor-train representation. The computational complexity of pseudoinversion in tensor decomposition [19] is $\mathcal{O}(pnm^3)$, with $m$ being the number of data points, which scales poorly with data volume.

All learning-based methods (SREINet, EQL, and SymbolNet) exhibit more favorable time scaling than MANDy as the number of data points increases, primarily because they use minibatch training. EQL and SymbolNet show a mild, approximately linear increase in wall-clock time over the tested range. However, their training is sensitive to weight initialization, which can lead to suboptimal convergence and failure to discover the exact governing equations in some dimensions, as indicated by the hollow markers in Fig. 12. In our implementation and selected hyperparameter settings, SymbolNet exhibits a steeper increase in wall-clock time than EQL because its dynamic pruning scheme introduces additional trainable masks/thresholds and sparsity-regularization terms, which increase the computational overhead. In these cases, neither EQL nor SymbolNet reaches the training-loss threshold for early stopping, so the full prescribed training budget is used. This contributes to the increase in computational time as the data volume grows.

In contrast, SREINet maintains a relatively constant computational time as more data points are used. It seems the required number of training steps for convergence remains statistically constant when sufficient data are available. In fact, the memory consumption of SREINet is $\mathcal{O}(pn^2 + pnm_b)$, while, for a fixed minibatch size $m_b$, the computational load per training step is $\mathcal{O}(pn^2)$ for both forward and backward propagation. The computational time of SREINet also has greater fluctuations than other methods, as the number of pruning periods required for convergence can vary across runs and system dimensions. Despite this variability, the pruning scheme in our framework is robust enough to manage the

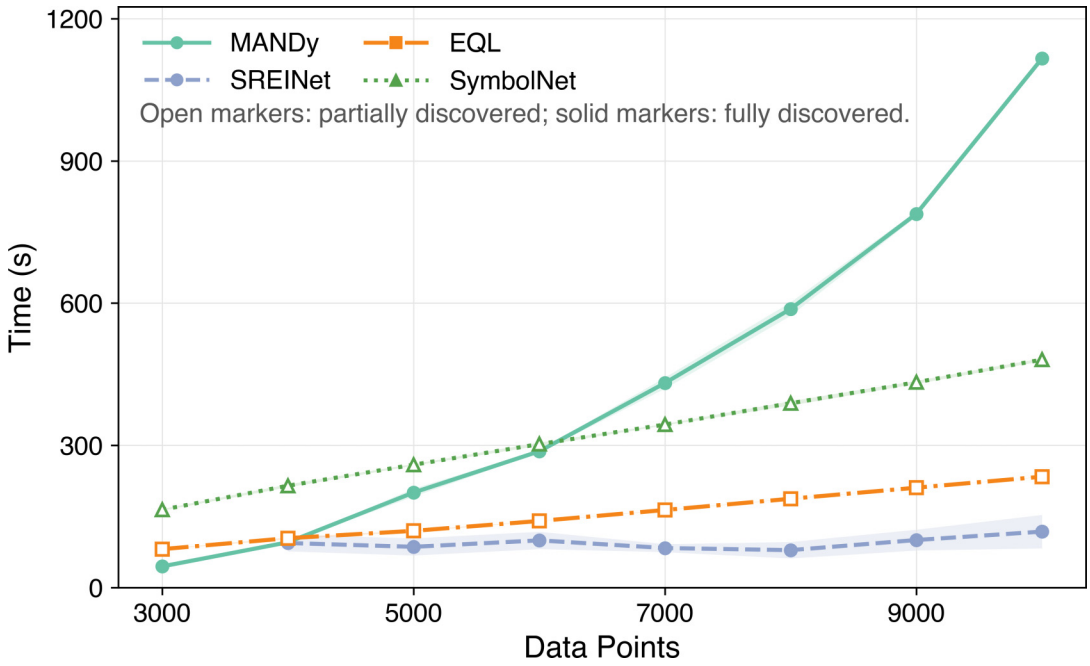


FIG. 12. Computational time versus the number of FPU training data points for MANDy, SREINet, EQL, and SymbolNet. Filled markers denote complete equation recovery, whereas open markers denote partial equation recovery. The SREINet curve begins at 4000 data points because training failed for several output dimensions at 3000 data points.

randomness, ultimately achieving convergence over several pruning periods.

*Memory efficiency.* In comparison with the conventional matrix-based optimization method [14,17], the reduction in the memory consumption of the proposed architecture is substantial. As shown in Fig. 13, the memory reduction becomes increasingly pronounced as both the system dimension and the order of nonlinearity increase. For $n = 100$, SREINet reduces the memory requirement by approximately 3, 4, and 7 orders of magnitude for $p = 2$, $p = 3$, and $p = 5$, respectively. For $p = 5$ and $n = 1000$, the reduction exceeds 9 orders of magnitude. A more detailed comparison of the computational (space and time) complexity of the proposed SREINet architecture with the matrix-based method is presented in Note 2 in the Supplemental Material and Tables S1 and S2 in the Supplemental Material [39].

*Training dynamics.* Figure 14 illustrates the training dynamics of representative dimensions from the $\phi^4$ model, the Kuramoto oscillators, and the Hindmarsh-Rose network.

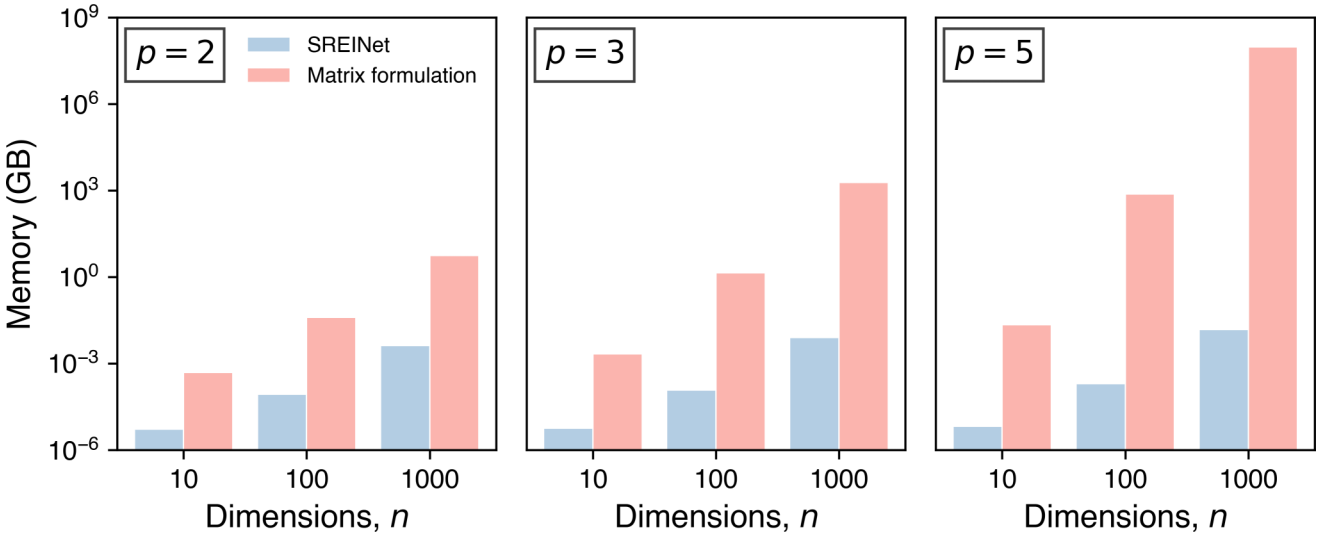


FIG. 13. Comparison of memory consumption of SREINet with the conventional matrix formulation across different system dimensions and orders of nonlinearity. For $p = 3$ and $n = 100$, SREINet reduces the memory consumption by more than 4 orders of magnitude. For $p = 5$ and $n = 1000$, the reduction exceeds 9 orders of magnitude, making SREINet scalable to systems with hundreds and even thousands of dimensions. The detailed numbers are listed in Table S1 in the Supplemental Material [39].

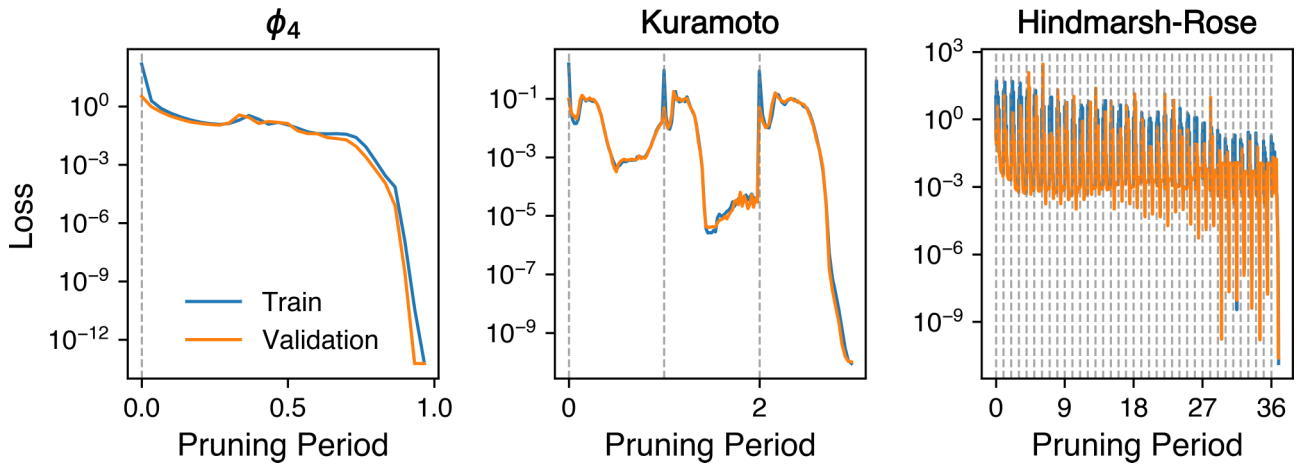


FIG. 14. Training dynamics of SREINet. Representative dimensions from the $\phi^4$ model, Kuramoto oscillators, and Hindmarsh-Rose network are selected, which demonstrate the training dynamics with fast (left), moderate (middle), and robust (right) convergence.

These cases exemplify the varying degrees of convergence handled by SREINet: fast, moderate, and robust convergence.

The left panel (fast convergence) depicts a scenario where the loss decays almost monotonically. This typically occurs in systems with a smoother optimization landscape, where the library terms are efficiently identified in the initial period.

The middle panel (moderate convergence) demonstrates a case where convergence is achieved through three reset periods. We observe characteristic segments where the loss stagnates or temporarily increases before the end of a period, which is often a result of the pruning of dominant library terms. Crucially, the resetting mechanism allows the optimizer to escape these local plateaus. Despite the transient increase in loss following a reset of pruning threshold, the model discovers a lower-loss trajectory in the subsequent period, eventually reaching a high-precision sparse solution.

The right panel (robust convergence) showcases a highly complex scenario requiring 37 pruning periods. In this case, the algorithm undergoes an extensive exploration phase, evidenced by the fluctuations in the local loss minima during the intermediate periods. This extended trajectory underscores the robustness of the *S*-shaped pruning schedule; the periodic resetting prevents premature commitment to an incorrect sparse structure, allowing the model to iteratively refine its selection until the true governing dynamics are captured.

To further demonstrate the effects brought by the network and pruning scheme, we perform an ablation study (see Note 10 in the Supplemental Material [39]). The results show that the *S*-shaped pruning schedule significantly improves the convergence of the training process, while the network architecture itself contributes to the compact representation with reduced memory consumption. The combination of the two components in SREINet leads to a scalable and robust training process for discovering governing equations of high-dimensional systems.

It should be noted that while the pruning scheme significantly alleviates the sensitivity to hyperparameter tuning, it does not guarantee global convergence in every extreme configuration. For example, an ill-configured pruning profile (e.g., an excessive maximum threshold or an inappropriate pruning schedule) may lead to overpruning of coefficients in each period. Consequently, the loss may stagnate without further reduction. However, the ability of SREINet to "reactivate" candidate functions provides a crucial error-correction buffer that traditional static thresholding methods lack.

## IV. CONCLUSION

Discovering governing equations of high-dimensional systems from data is of interest for engineering and physics applications as it enables the extraction of interpretable models without prior knowledge of the underlying physics. Unlike black-box models of NNs, interpretable models offer insights into the internal dynamical mechanisms of the system to facilitate prediction, monitoring, and control. For high-dimensional dynamical systems, the curse of dimensionality leading to an exponential increase of searching space or candidate functions becomes a major obstacle for data-driven model or equation discovery. In spite of the tremendous interest in finding the equations, most efforts to date were limited to relatively low-dimensional dynamical systems or situations where the available data are scarce.

Our present work addresses the curse of dimensionality by articulating an interpretable machine-learning framework with embedded sparsity, SREINet, for extracting the equations for high-dimensional dynamical systems described by a large number of coupled ordinary differential equations. The basic idea is to embed sparse regression into an inherently interpretable neural network. A single neural network is used to sequentially learn the sparse dynamics of each dimension, followed by reconstructing the governing equations of the original system. The reformulation of sparse regression with an effective pruning method based on an *S*-shaped periodic pruning profile to facilitate the learning of high-dimensional dynamics reduces the space complexity. In particular, the memory consumption and computational load can be reduced by orders of magnitude for high-dimensional systems in comparison with conventional explicit-library equation-finding formulations. Intuitively, resetting the pruning schedule pro-

vides more opportunities to restart the pruning process from a subspace closer to the optimal solution, thereby avoiding the situation where the dominant term is accidentally pruned during training. Our SREINet incorporating the *S*-shaped periodic pruning enables us to readily scale equation finding to systems of over 100 dimensions on a single laptop. In principle, with cloud computing, SREINet can be applied to systems with more than 1000 dimensions.

We tested SREINet on a number of high-dimensional systems arising from climate science and complex networks. These benchmarks are challenging for conventional equation-finding methods because of their high dimensionality and the associated large data volume. The learned governing equations accurately capture the coherent structures, highlighting the capability of our framework in discovering physical solutions in spatiotemporal dynamics. Besides the ability to handle high dimensions, SREINet can tolerate noise levels up to 10% across all dimensions. In comparison with MANDy, SREINet is advantageous for big datasets by statistically maintaining a constant wall-clock time, which is particularly suitable for applications with big data. Our dimensionwise training scheme enables parallel training of multiple dimensions. In addition, SREINet is capable of handling large datasets because of the use of minibatch training. With the increasing number of sensors deployed for scientific research and industrial activities, this feature can make SREINet advantageous relative to traditional sparse-regression formulations. The head-to-head benchmarks further demonstrate SREINet maintains relatively small coefficient and rollout errors while recovering models whose symbolic complexity remains close to the ground truth. These results highlight SREINet's ability to combine computational scalability, robustness to noise, and explicit recovery of parsimonious governing equations.

SREINet has a number of limitations. For example, selecting an appropriate set of library functions remains a challenge. Without suitable candidate functions, the framework may fail to capture the true dynamics. A further limitation is the requirement for full-state observability, a condition that is often not satisfied in practice. For example, many available datasets for high-dimensional systems, such as climate data and electrophysiological recordings, often involve high-dimensional latent processes, partial measurements, and unmeasured inputs, which present additional challenges for the application of SREINet. SREINet solves a nonconvex optimization problem and, therefore, may converge to a local minimum without carefully selected hyperparameters.

There are several directions for future research that stem from the present work. First, the pruning method could be improved by simplifying the hyperparameter-tuning process. Currently, the parameters controlling the *S*-shaped pruning curve must be tuned jointly with other training hyperparameters (e.g., learning rate, batch size), making the process time consuming. Future work could investigate principled guidelines for selecting these parameters to reduce tuning complexity. Second, while we have demonstrated that the resetting mechanism of *S*-shaped pruning can significantly improve the convergence, a formal analysis is needed to understand the convergence and robustness of this pruning method. Third, although the integration with adaptive stochastic gradient descent optimizer makes SREINet scale with data size, the available data are scarce for many scientific problems. It is worthwhile to study the performance of SREINet by applying second-order optimizers for scarce data. We also note that SREINet admits nonunique weight-level optima, while coefficient reconstruction preserves equation-level consistency; a formal analysis of this optimization geometry is left for future work. Finally, successful identification of governing equations requires that the training data cover sufficiently informative regions of the state space. In systems, where trajectories are concentrated near a low-dimensional attractor or synchronized manifold, such as strongly synchronized oscillator networks, it can be difficult to distinguish candidate functions without sufficiently diverse trajectories or initial conditions. A potential direction is to study dynamical noise, which may aid model discrimination by driving trajectories away from low-dimensional attractors or synchronized manifolds when its magnitude is appropriate [64,65].

In addition, it is possible to extend SREINet to handle rational nonlinearity. This can be done either through the lifting layer or directly injecting rational functions at the neurons. Beyond the matrix-based formulation, the parameters of these rational functions can also be trainable, allowing for learning not only the coefficients but also candidate functions. However, this modification could make the problem more complex. Therefore, new training schemes should be developed to ensure convergence. It is also possible to adapt SREINet for network inference by using known or estimated connectivity to impose structured masks on learnable interaction paths, in the spirit of topology-informed reservoir-computing approaches [66]. Another potential direction is to follow the approach of Refs. [67,68], where rational functions can be reformulated by multiplying their denominators to the right-hand side. This transforms rational terms into multiplicative forms compatible with the SREINet framework. However, this approach will lead to a null-space optimization problem. Although Ref. [68] has proposed bypassing this problem by iteratively designating a candidate function on the left-hand side until the best fit is achieved, it remains computationally prohibitive and difficult to scale for high-dimensional problems. We believe that a new training scheme should be developed to tackle this problem. Furthermore, while the triple-pendulum example demonstrates SREINet's ability to identify a generalizable model even when the candidate library lacks critical terms, investigating whether such discovered models can be effectively integrated into downstream control applications remains a promising future direction. Finally, extending SREINet to discover and predict spatiotemporal phenomena modeled by partial differential equations is another yet timely direction for future research. All the above are under consideration, and results will be reported in forthcoming publications.

## ACKNOWLEDGMENTS

S.X. thanks Brendon G. Anderson and Matt Haberland for helpful discussions, as well as Kai Li for suggestions on figure formatting. This work was supported by Keysight Technologies, Inc., the Research, Scholarly, and

Creative Activities (RSCA) Program awarded by the Cal Poly Division of Research, Economic Development and Graduate Education (S.X.), and the U.S. National Science Foundation under Grant No. DMS-2204782 (E.G.C.). The work at Arizona State University was supported by the US Army Research Office under Grants No. W911NF-26-2-A002 and No. W911NF-24-2-0228.

S.X. conceived the research, developed the software, performed the data analysis, and cowrote the initial manuscript with E.G.C. Q.H. contributed to software development and made a key contribution to the development of the pruning algorithm in this work. E.G.C. and Y.-C.L. contributed to the study design, interpretation of results, and provided intellectual input throughout the project. S.X., E.G.C., and Y.-C.L. revised the manuscript. All authors reviewed and approved the final version of the manuscript.

The authors declare no competing interests.

## DATA AVAILABILITY

The custom code implementing SREINet is publicly available on Zenodo [69]. The corresponding development repository is available on GitHub, as cited in Ref. [70]. Code used for the benchmarking comparisons is not included in this public release. The data supporting the findings of this study are publicly available on Zenodo [71].

[1] P. Davidson, *Turbulence: An Introduction for Scientists and Engineers* (Oxford University Press, Oxford, UK, 2004).

[2] P. G. Kevrekidis, D. J. Frantzeskakis, and R. Carretero-González, *Emergent Nonlinear Phenomena in Bose-Einstein Condensates: Theory and Experiment* (Springer, Berlin, Heidelberg, 2008), Vol. 45.

[3] A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchronization: A Universal Concept in Nonlinear Sciences* (Cambridge University Press, Cambridge, UK, 2001).

[4] J. V. José and E. J. Saletan, *Classical Dynamics: A Contemporary Approach* (Cambridge University Press, Cambridge, UK, 1998).

[5] J. P. Crutchfield and B. McNamara, Equations of motion from a data series, Complex Syst. **1**, 417 (1987).

[6] T. Sauer, Reconstruction of dynamical systems from interspike intervals, Phys. Rev. Lett. **72**, 3811 (1994).

[7] U. Parlitz, Estimating model parameters from time series by autosynchronization, Phys. Rev. Lett. **76**, 1232 (1996).

[8] C. Yao and E. M. Bollt, Modeling and nonlinear parameter estimation with Kronecker product representation for coupled oscillators and spatiotemporal systems, Physica D **227**, 78 (2007).

[9] J. Bongard and H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA **104**, 9943 (2007).

[10] M. Schmidt and H. Lipson, Distilling free-form natural laws from experimental data, Science **324**, 81 (2009).

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, USA, 1989).

[12] S.-M. Udrescu and M. Tegmark, AI Feynman: A physics-inspired method for symbolic regression, Sci. Adv. **6**, eaay2631 (2020).

[13] B. K. Petersen, M. Landajuela, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, in *Proceedings of the International Conference on Learning Representations* (OpenReview.net, Virtual Event, Austria, 2021).

[14] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, and C. Grebogi, Predicting catastrophes in nonlinear dynamical systems by compressive sensing, Phys. Rev. Lett. **106**, 154101 (2011).

[15] Y.-C. Lai, Finding nonlinear system equations and complex network structures from data: A sparse optimization approach, Chaos **31**, 082101 (2021).

[16] R. G. Baraniuk, Compressive sensing, IEEE Signal Process. Mag. **24**, 118 (2007).

[17] 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. USA **113**, 3932 (2016).

[18] K. Ikeda, Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system, Opt. Commun. **30**, 257 (1979).

[19] P. Gelß, S. Klus, J. Eisert, and C. Schütte, Multidimensional approximation of nonlinear dynamical systems, J. Comput. Nonlinear Dyn. **14**, 061006 (2019).

[20] R. González-García, R. Rico-Martínez, and I. Kevrekidis, Identification of distributed parameter systems: A neural net based approach, Comp. Chem. Eng. **22**, S965 (1998).

[21] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, and Prabhat, Deep learning and process understanding for data-driven Earth system science, Nature (London) **566**, 195 (2019).

[22] D. Durstewitz, G. Koppe, and M. I. Thurm, Reconstructing computational system dynamics from neural data with recurrent neural networks, Nat. Rev. Neurosci. **24**, 693 (2023).

[23] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, in *Advances in Neural Information Processing Systems (NeurIPS)* (Curran Associates, Inc., Red Hook, NY, 2018), pp. 6571–6583.

[24] H. Jaeger, The "Echo State" Approach to Analysing and Training Recurrent Neural Networks, GMD Report No. 148 (GMD Forschungszentrum Informationstechnik, Sankt Augustin, Germany, 2001), doi: 10.24406/publica-fhg-291111.

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

[26] L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, Reservoir computing as digital twins for nonlinear dynamical systems, Chaos **33**, 033111 (2023).

[27] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, Data-driven discovery of coordinates and governing equations, Proc. Natl. Acad. Sci. USA **116**, 22445 (2019).

[28] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, in *Proceedings of the 34th Conference on Neural Information Processing Systems* (Vancouver, Canada, 2020), pp. 17429–17442.

[29] I. Mezić, *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*, *Lecture Notes in Control and Information Sciences* (Springer, Cham, Switzerland, 2020), Vol. 484.

[30] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, A kernel-based method for data-driven Koopman spectral analysis, J. Comp. Dyn. **2**, 247 (2015).

[31] S. Sahoo, C. Lampert, and G. Martius, in *International Conference on Machine Learning* (PMLR, 2018), pp. 4442–4450.

[32] H. F. Tsoi, V. Loncar, S. Dasu, and P. Harris, SymbolNet: Neural symbolic regression with adaptive dynamic pruning for compression, Mach. Learn.: Sci. Technol. **6**, 015021 (2025).

[33] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, KAN: Kolmogorov-Arnold networks, in *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)* (OpenReview.net, 2025).

[34] Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark, KAN 2.0: Kolmogorov-Arnold networks meet science, arXiv:2408.10205.

[35] A. N. Kolmogorov, On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, Dokl. Akad. Nauk SSSR **114**, 953 (1957).

[36] A. B. Givental, B. A. Khesin, J. E. Marsden, A. N. Varchenko, V. A. Vassiliev, O. Y. Viro, and V. M. Zakalyukin, On the representation of functions of several variables as a superposition of functions of a smaller number of variables, in *Collected Works: Representations of Functions, Celestial Mechanics and KAM Theory, 1957–1965* (Springer, Berlin, 2009), pp. 25–46.

[37] S. Panahi, M. Moradi, E. M. Bollt, and Y.-C. Lai, Data-driven model discovery with Kolmogorov-Arnold networks, Phys. Rev. Res. **7**, 023037 (2025).

[38] S. Xing, Q. Han, and E. G. Charalampidis, CombOpNet: A neural-network accelerator for SINDy, J. Vib. Test. Sys. Dyn. **9**, 1 (2025).

[39] See Supplemental Material at http://link.aps.org/supplemental/10.1103/pdkf-98zb for a mathematical proof, computational-complexity analysis, algorithmic details, data-generation and training procedures, and additional numerical and experimental results, which includes Ref. [40].

[40] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, Automatic differentiation in machine learning: A survey, J. Mach. Learn. Res. **18**, 1 (2018).

[41] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Stat. Soc. Ser. B (Method.) **58**, 267 (1996).

[42] A. E. Hoerl and R. W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics **12**, 55 (1970).

[43] M. Zhu and S. Gupta, in *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Workshop Track Proceedings* (OpenReview.net, 2018).

[44] M. Zhang, S. Kim, P. Y. Lu, and M. Soljačić, Deep learning and symbolic regression for discovering parametric equations, IEEE Trans. Neural Networks Learn. Syst. **35**, 16775 (2024).

[45] C. M. Bishop and H. Bishop, *Deep Learning* (Springer Nature, Cham, Switzerland, 2024).

[46] E. N. Lorenz, Predictability—A problem partly solved, in *Seminar on Predictability* (European Centre for Medium-Range Weather Forecasts (ECMWF), Reading, UK, 1995), Vol. 1, pp. 1–18.

[47] P. G. Kevrekidis and J. Cuevas-Maraver, *A Dynamical Perspective on the $\phi^4$ Model: Past, Present and Future* (Springer, Cham, Switzerland, 2019).

[48] P. Kevrekidis, *The Discrete Nonlinear Schrödinger Equation*, 1st ed. (Springer-Verlag, Heidelberg, 2009).

[49] M. J. Ablowitz and J. F. Ladik, Nonlinear differential-difference equations, J. Math. Phys. **16**, 598 (1975).

[50] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data, JOSS **5**, 2104 (2020).

[51] N. Manton and P. Sutcliffe, *Topological Solitons* (Cambridge University Press, Cambridge, UK, 2004).

[52] Y. S. Kivshar and G. P. Agrawal, *Optical Solitons: From Fibers to Photonic Crystals* (Academic Press, London, 2003).

[53] M. A. Ablowitz and P. A. Clarkson, *Solitons, Nonlinear Evolution Equations and Inverse Scattering*, *London Mathematical Society Lecture Note Series* (Cambridge University Press, Cambridge, UK, 1991).

[54] C. Kharif, E. Pelinovsky, and A. Slunyaev, *Rogue Waves in the Ocean*, *Advances in Geophysical and Environmental Mechanics and Mathematics* (Springer-Verlag, Berlin, Heidelberg, 2009).

[55] J. L. Hindmarsh and R. M. Rose, A model of neuronal bursting using three coupled first order differential equations, Proc. R. Soc. B **221**, 87 (1984).

[56] J. A. Acebrón, L. L. Bonilla, C. J. Pérez Vicente, F. Ritort, and R. Spigler, The Kuramoto model: A simple paradigm for synchronization phenomena, Rev. Mod. Phys. **77**, 137 (2005).

[57] K. Kaheman, U. Fasel, J. J. Bramburger, B. Strom, J. N. Kutz, and S. L. Brunton, The experimental multi-arm pendulum on a cart: A benchmark system for chaos, learning, and control, HardwareX **15**, e00465 (2023).

[58] A. Savitzky and M. J. E. Golay, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. **36**, 1627 (1964).

[59] C. Hoffmann, E. Charalampidis, D. Frantzeskakis, and P. Kevrekidis, Peregrine solitons and gradient catastrophes in discrete nonlinear Schrödinger systems, Phys. Lett. A **382**, 3064 (2018).

[60] W. Zhu, W. Khademi, E. G. Charalampidis, and P. G. Kevrekidis, Neural networks enforcing physical symmetries in nonlinear dynamical lattices: The case example of the Ablowitz–Ladik model, Physica D **434**, 133264 (2022).

[61] B. Prokop and L. Gelens, From biological data to oscillator models using SINDy, iScience **27**, 109316 (2024).

[62] E. Fermi, J. Pasta, and S. Ulam, *Studies of Nonlinear Problems*, I, Technical Report LA-1940 (Los Alamos Scientific Laboratory, Los Alamos, NM, 1955).

[63] I. V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. **33**, 2295 (2011).

[64] M. J. Panaggio, M. V. Ciocanel, L. Lazarus, C. M. Topaz, and B. Xu, Model reconstruction from temporal data for coupled oscillator networks, Chaos **29**, 103116 (2019).

[65] A. Banerjee, J. D. Hart, R. Roy, and E. Ott, Machine learning link inference of noisy delay-coupled networks with optoelectronic experimental tests, Phys. Rev. X **11**, 031014 (2021).

[66] K. Srinivasan, N. Coble, J. Hamlin, T. Antonsen, E. Ott, and M. Girvan, Parallel machine learning for forecasting the dynamics of complex networks, Phys. Rev. Lett. **128**, 164101 (2022).

[67] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Inferring biological networks by sparse identification of nonlinear dynamics, IEEE Trans. Mol. Biol. Multi-Scale Commun. **2**, 52 (2016).

[68] K. Kaheman, J. N. Kutz, and S. L. Brunton, SINDy-PI: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics, Proc. R. Soc. A **476**, 20200279 (2020).

[69] S. Xing and Q. Han, SREINet: Sparse regression embedded interpretable network, version v1.0.0 [software], Zenodo, 2026, https://doi.org/10.5281/zenodo.22048994.

[70] https://github.com/siyuan-xing/SREINet.

[71] S. Xing, SREINet: Data and figure source files for data-driven discovery of high-dimensional dynamical systems with sparse interpretable neural networks, version v2 [dataset], Zenodo, 2026, https://doi.org/10.5281/zenodo.22058158.