Journal of Vibration Testing and System Dynamics

Journal homepage: https://lhscientificpublishing.com/Journals/JVTSD-Default.aspx

CombOpNet: a Neural-Network Accelerator for SINDy

Siyuan Xing[1†], Qingyu Han[2], Efstathios G. Charalampidis[3,4]

[1] Mechanical Engineering Department, California Polytechnic State University, San Luis Obispo, CA, 93407-0403, USA
[2] Electric Engineering Department, California Polytechnic State University, San Luis Obispo, CA, 93407-0403, USA
[3] Department of Mathematics and Statistics, San Diego State University, San Diego, CA 92182-7720, USA
[4] Computational Science Research Center, San Diego State University, CA 92182-7720, USA

Submission Info

Communicated by C Steve Suh
Received 21 July 2024
Accepted 21 August 2024
Available online 1 January 2025

Keywords

SINDy
CombOpNet
Neural Networks
Dynamical Systems

Abstract

In the present work, we develop and assess a compact neural-network architecture called hereafter the Combinatorial Operation Neural Network (CombOpNet) which is designed to mitigate the curse of dimensionality of Sparse Identification of Nonlinear Dynamics (SINDy). Within CombOpNet, nonlinear terms in dynamical equations are represented by a chain of multiplication of univariate functions. These functions correspond to neurons of a multi-layer neural network which allows the construction of nonlinear terms in dynamical equations by employing forward propagation. This way, the CombOpNet can form and perform summations of all possible combinations of univariate functions now using far fewer weights, which themselves can reconstruct the elements in the coefficient matrix of SINDy. If n and p denote the number of dimensions and order of nonlinearity, respectively, the present method reduces the number of SINDy coefficients from $n\binom{n+p}{n}$ to $\mathcal{O}(n^3 p)$. This reduction facilitates the scalability of SINDy, making SINDy applicable to multidimensional systems such as power grids and climate models. We discuss the implementation of CombOpNet in Tensorflow, and demonstrate its applicability to several complex nonlinear dynamical systems with an eye towards solving high-dimensional problems.

1 Introduction

The discovery of predictive dynamical equations that govern the behavior of complex nonlinear phenomena purely from data has been a focal point of mathematical, physical, and engineering modeling [1–3]. With the advances in sparse regression [4], innovative data-driven methods have been discovered, such as the so-called Sparse Identification of Nonlinear Dynamics or SINDy [5] which are capable of identifying nonlinear dynamical equations of a pertinent system. Indeed, those data-driven methods are

†Corresponding author.
Email address: sixing@calpoly.edu

ISSN 2475-4811, eISSN 2475-482X/$-see front materials
DOI:10.5890/JVTSD.2025.03.001

pivotal in predicting and unraveling the behavior of complex nonlinear dynamical systems that are difficult to model from first principles, including, e.g., power grids [6], physicochemical modeling [7], climate systems [8], and human neurobiology [9], among many others.

In the realm of discovering physical laws from data, the pioneering works in [10] and [11] employed symbolic regression (SR) combined with genetic algorithms to search for governing equations. The optimal model therein is identified by leveraging a Pareto front to achieve the balance of accuracy and model complexity. More recently, SR has been further enhanced by incorporating deep-learning techniques. For example, the work in [12] employed a neural network (NN) to assist the identification of intrinsic properties of a system such as symmetries, separability, and compositionality. These properties were then used to decompose the original problem into lower-dimensional subproblems that are more trackable to solve. Indeed, the work in [13] integrated reinforcement learning into SR to accelerate the search of intrinsic properties in the space of small models/symbolic expressions.

In parallel, the breakthrough of compressive sensing [14, 15] has sparked tremendous interest in utilizing sparse regression for discovering governing equations in dynamical systems. In particular, the work in [16] formulated the system identification of nonlinear systems as an l_1-minimization problem of coefficients in the series expansion form. A few years later, the SINDy method was introduced in [5] which adopts a sequential-threshold-least-squares algorithm (abbreviated hereafter as STLSQ) to promote sparsity. This method is more robust in dealing with noisy data, and its convergence was proved in [17]. Since then, many variants of SINDy have been developed, such as SINDy with control [18], and partial differential equations (PDEs) [19, 20] together with the weak SINDy (or WSINDy) that utilizes the weak formulation of PDEs and eliminates pointwise derivative approximations [21], implicit SINDy [22,23], Hyper-SINDy [24] for stochastic dynamical systems, and sparse tensor regression method or "reactive SINDy" [25], along with their applications in many areas and fields. Those include model predictive control [26], reinforcement learning [27], fluids [28], biological processes [29], Poincaré maps and identification of slow timescale dynamics [30, 31], and discrete systems [32], to name just a few among of those applications.

Recent efforts have been concentrated on improving the performance of SINDy from various aspects. These include the development of new sparse-promoting regularization schemes, such as Sparse Relaxed Regularized Regression (SR3) [33], reweighted l_1-regularized least squares [34], and thresholding with sensitivity analysis [35] to rank important terms effectively. Strategies for denoising, such as a priori denoising [36], and bootstrapping [37] have also been explored. Furthermore, leveraging automatic differentiation [38, 39] has shown a great promise in improving SINDy's performance. Techniques for noise quantification, such as Sparse Priors Bayesian Quantification [40] and Sparse Threshold Bayesian regression [41], have also been investigated. However, most existing examples of SINDy have been limited to low-dimensional dynamical systems. When applied to ultra high-dimensional nonlinear systems, SINDy faces the curse of dimensionality, where the size of the coefficient matrix increases exponentially (see Section 2.1).

Indicatively, Gelß et al [42] attacked this problem by applying tensor decomposition to mitigate the memory consumption required to store the large coefficient matrix. Additionally, autoencoders [43] have been integrated into SINDy to project high-dimensional data into a low-dimensional space where a sparse representation of dynamics can be discovered by the SINDy algorithm. However, the black-box nature of autoencoders, typically constructed using multilayer perceptrons (MLP), makes it difficult to interpret the transformation from the original space to the reduced dimension. However, in our present work, we aim to approach this problem from a different angle by developing a novel neural-network structure to replace the computationally expensive matrix-based formulation of SINDy.

Indeed, what we pursue herein is not a universal approximator but instead, we aim at *accelerating* SINDy through the use of the multi-layer structure of forward networks. The rationale is that with the proper selection of library functions, the current matrix representation of SINDy leads to a redundant

(sparse) matrix whose size grows exponentially with increasing dimensions, directly contributing to the curse of dimensionality. By replacing the matrix-based representation with its NN counterpart, we can potentially free significant computational resources, thus enabling the algorithm to scale to very high-dimensional, i.e., multi-dimensional problems. We achieve this goal by leveraging the fact that many nonlinear terms of dynamical systems can be represented by a chain of multiplication of univariate functions. We utilize this insight by making these univariate functions the neurons of a multilayer NN, and employing forward propagation to reconstruct the multiplication chain. This yields a new NN structure capable of forming and summing all possible combinations of the univariate functions with far fewer number of weights. These weights can be combined to reconstruct the elements in the coefficient matrix of SINDy. We term this structure *Combinatorial Operation Neural Networks* (CombOpNet).

The CombOpNet has two appealing properties: (1) a stacked structure, and (2) path multiplicity. The former property allows for the decoupling of each output dimension while the latter one helps mitigate the negative effects induced by the cross-coupling of weights shared by different forward paths, thus providing alternative routes when one path is diminished by a zero weight. With this network, we successfully reduce the parameters in the coefficient matrix from $n\binom{n+p}{n}$ to $\mathcal{O}(n^3 p)$, a remarkable reduction that theoretically enables scaling to hundreds or even thousands of dimensions. We have implemented this network in the framework of Tensorflow, with the adaptation of the STLSQ method. We have investigated several example systems to demonstrate the potential of CombOpNet for solving high-dimensional problems.

The remainder of the paper is organized as follows. In Section 2, we review the SINDy algorithm and its limitations while studying high-dimensional problems. To overcome this limitation, we present the structure of Combinatorial Operation Neural Networks (CombOpNet), and discuss its properties in Section 3. The training scheme for CombOpNets is introduced in Section 4 where the STLSQ method is adapted to fit in the training scheme in Tensorflow. Moreover, we show the method of reconstructing composite weights of coefficients contained by the governing equations therein. In Section 5, multiple examples are provided to demonstrate the efficacy of this method. Finally, we conclude this paper and discuss possible future directions in Section 6.

2 Sparse system identification of nonlinear dynamics (SINDy)

The algorithm of sparse identification of nonlinear dynamics (SINDy) [5] is applied to dynamical systems modeled by ordinary differential equations (ODEs) in the form of

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}). \tag{1}$$

This algorithm aims to identify a parsimonious representation of nonlinear dynamic systems purely from data. To do so, it introduces a library of candidate functions $\boldsymbol{\Theta}(\boldsymbol{x})$ such that

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{\Theta}(\boldsymbol{x})\boldsymbol{\Xi}, \tag{2}$$

where $\boldsymbol{\Xi}$ is the coefficient matrix of $\boldsymbol{\Theta}$. In addition, $\boldsymbol{\Xi}$ is expected to be sparse, as SINDy assumes that, given a sufficient number of candidate functions, $\boldsymbol{f}$ can be faithfully represented by using only a few active terms from the library of functions embodied in $\boldsymbol{\Theta}$.

The active (nonzero) coefficients of $\boldsymbol{\Xi}$ are then solved by using an optimization method integrated with sparse regularization. First, m numbers of discrete points (or values of the state variable at m temporal points) represented by

$$\boldsymbol{X} = \left[\boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \cdots, \boldsymbol{x}(t_m)\right]^T \tag{3}$$

are sampled along a trajectory, together with their respective time derivatives

$$\dot{\boldsymbol{X}} = \left[\dot{x}(t_1), \dot{x}(t_2), \cdots, \dot{\boldsymbol{x}}(t_m)\right]^T, \tag{4}$$

which can also be approximated by e.g., finite difference or other methods. When noisy data is collected, denoising techniques [36] may be applied. The data $\boldsymbol{X}$ are then used to construct a library of candidate functions

$$\boldsymbol{\Theta}(\boldsymbol{X}) = \begin{bmatrix} | & | & | & | & | & | & | & | & | \\ 1 & \boldsymbol{X} & \boldsymbol{X}^2 & \dots & \sin(\boldsymbol{X}) & \cos(\boldsymbol{X}) & \sin(2\boldsymbol{X}) & \cos(2\boldsymbol{X}) & \dots \\ | & | & | & | & | & | & | & | & | \end{bmatrix}, \tag{5}$$

where

$$\boldsymbol{X}^2 = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \dots & x_2^2(t_1) & \dots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \dots & x_2^2(t_2) & \dots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \dots & x_2^2(t_m) & \dots & x_n^2(t_m) \end{bmatrix}. \tag{6}$$

Substitution of such a dataset in Eqs. (1)-(2) yields

$$\dot{\boldsymbol{X}} = \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi}. \tag{7}$$

At last, a sparse set of coefficients $\boldsymbol{\Xi}$ that gives the best fit of data is identified by solving a regularized sparse regression problem formulated as

$$\boldsymbol{\Xi} = \text{argmin}_{\boldsymbol{\Xi}} ||\dot{\boldsymbol{X}} - \boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi}||_2 + R(\boldsymbol{\Xi}), \tag{8}$$

where $R(\boldsymbol{\Xi})$ is the penalty term that promotes the sparsity of $\boldsymbol{\Xi}$. The sparse regression such as LASSO [44] and STLSQ [5] are commonly selected. The identified $\boldsymbol{\Xi}$ leads to a parsimonious model that includes essential terms to represent system dynamics. Further details of this approach can be found in [18].

2.1 Curse of dimensionality

The matrix $\boldsymbol{\Xi}$ contains coefficients of both linear and nonlinear terms in the identified model. While many coefficients are zero (yielding sparse matrix representations), the size of $\boldsymbol{\Xi}$ increases significantly with the number of dimensions and the order of nonlinearity. This expansion exacerbates the curse of dimensionality as one scales SINDy for high-dimensional problems.

To illustrate this, we restrict our analysis to polynomials up to the order of p. Herein, we assume that the dimensions of ODEs are n. The number of candidate functions is equivalent to the number of monomials (including the constant 1) up to p-th order. This can be formulated as a classical star-and-bar problem in combinatorics, or distributing n bars to separate p stars. The formula for the number of monomials is $\binom{n+p}{n}$. As each ODE of the form of Eq. (1) has independent combinations of monomials, this results in having the total number of elements in $\boldsymbol{\Xi}$ to be

$$N = n\binom{n+p}{n} = \frac{n(n+p)!}{n!p!}. \tag{9}$$

As depicted in Fig.1, the size of $\boldsymbol{\Xi}$ blows up with dimensions (number of state variables). This trend also holds true for the increase in polynomial order. Assuming $n \gg p$, we obtain

$$N = \frac{n(n+1)(n+2)\dots(n+p)}{p!} \approx \mathscr{O}(n^{p+1}). \tag{10}$$

In a similar vein, we conclude that

$$N = \frac{n(p+1)(p+2)\dots(p+n)}{n!} \approx \mathscr{O}(p^n) \tag{11}$$

holds when $p \gg n$. Evidently, such an exponential increase of coefficients makes SINDy intractable for very high-dimensional problems.

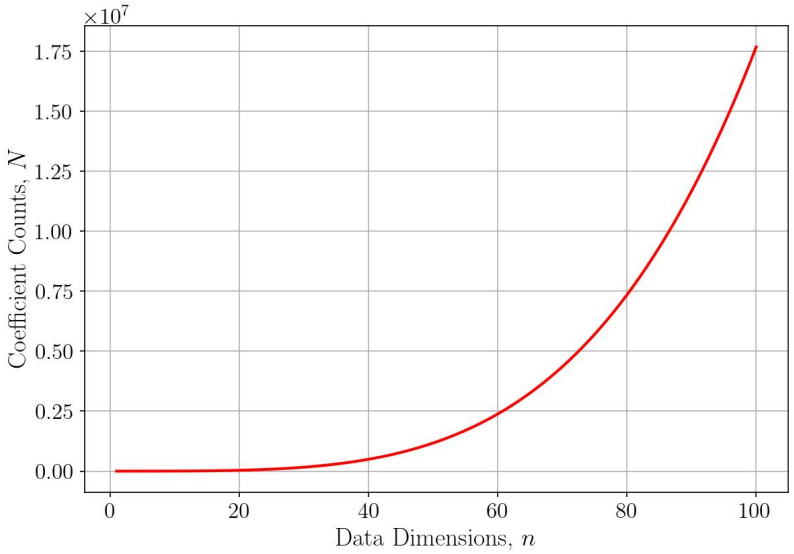

Fig. 1 The number of coefficients in the matrix $\boldsymbol{\Xi}$ blows up with increasing data dimensions. The highest polynomial of order p is fixed to 3 respectively.

3 Accelerating SINDy with combinatorial operation neural networks (CombOpNets)

In order to reduce the memory consumption of SINDy, Gelß et al [42] in 2019 proposed using tensor-train to represent the matrices $\boldsymbol{\Theta}$ and $\boldsymbol{\Xi}$. With this representation, one can establish a one-to-one correspondence between tensor-train and matrix representations. To reduce the computational cost, the inversion of the tensor train is then approximated by pseudoinversion (i.e., Moore-Penrose inverse) based on singular value decomposition (SVD).

Although we seek a similar goal to develop a compact representation of $\boldsymbol{\Xi}$, we further inspect the fundamental assumption of SINDy: the dynamics of a nonlinear system are governed by only a few dominant terms. In other words, $\boldsymbol{\Xi}$ is sparse with the majority of elements being zero. This indicates the redundancy of the matrix representation of $\boldsymbol{\Xi}$ and inspires us to further reduce the number of weights (coefficients) by using this property.

We achieve this goal by leveraging the structure of feedforward neural networks (NNs). More specifically, we realize the right-hand side (RHS) of a differential equation through the forward propagation of NNs. The rationale behind this approach is that the RHS normally takes the superposition form of nonlinear terms composed by the multiplication of univariate functions. For example, consider a nonlinear term represented by a p-th order monomial

$$(x_1)^{i_1}(x_2)^{i_2}\cdots(x_m)^{i_m}, \tag{12}$$

where $i_1+i_2+\cdots+i_m=p$. Such a monomial can be viewed as a chain of multiplication of first-order monomials, which can be accomplished through the forward propagation of a feedforward path of a NN. As shown in Fig.2, we can break the monomial $x_1x_k\ldots x_2$ into the multiplication of p univariate first-order monomials (e.g., $x_1\cdot x_k\ldots\cdot x_2$). Each first-order monomial then becomes a neuron (highlighted) in the feedforward network. The forward propagation along the path connected to such neurons will reconstruct the p-th order polynomial (we neglect the weights therein for the ease of discussion).

By further adding the shaded neurons representing additional variables and a constant 1 to each layer, we create a new NN structure. In this NN, each forward path generates a monomial and the associated weights. By aggregating these monomials at the output layer, we obtain an output containing

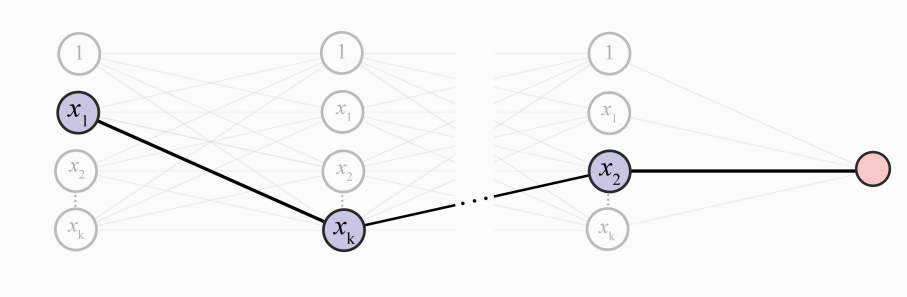

Fig. 2 Neural-network implementation (highlighted path) of the multiplication chain of Eq. (12). The highlighted path yields a n-th order polynomial $x_1 x_k \cdots x_2$, whose weight is the product of individual weights on the path. By adding the additional state variables and a constant one (shaded neurons), the forward propagation can aggregate all possible combinations of the multivariate polynomials up to p-th order at the output.

all possible combinations of such monomials up to the p-th order. Therefore, we name this new network as the Combinatorial Operation Neural Network, or CombOpNet.

However, it is important to note that the coefficients of the resulting multivariate polynomials are not fully independent, as each link/synapsis in CombOpNet can be shared by multiple paths. Nevertheless, since only a few dominant terms are necessary to describe the dynamics, only a small number of active paths are needed to faithfully represent a system's dynamics. In Appendix A.1, we demonstrate that achieving this with CombOpNet is not a demanding task.

3.1 Structure of CombOpNet

We now present and generalize the structure of CombOpNet, (see Fig.3). Besides first-order monomials, we can assign a candidate function (e.g., constant, polynomial, trigonometric functions) to each neuron, thus giving

$$\Theta_i(\zeta) \in \{1, \zeta, \sin(\zeta), \sin(2\zeta), \ldots, \cos(\zeta), \cos(2\zeta), \ldots\} \tag{13}$$

where ζ represents the input data of an arbitrary dimension. We define the candidate functions of the j-th layer as

$$\boldsymbol{\Theta}^{[j]} = [\Theta_1^{[j]}, \Theta_2^{[j]}, \ldots, \Theta_k^{[j]}]^T. \tag{14}$$

Therefore, the output of CombOpNet takes the form

$$\mathscr{N}(\boldsymbol{x}) = \hat{y}(\boldsymbol{x}) = \mathbf{1}_{k\times 1} \cdot \boldsymbol{\Theta}^{[p]}(\boldsymbol{x}) \odot \boldsymbol{W}^{[p-1]} \cdots \boldsymbol{W}^{[2]} \cdot \boldsymbol{\Theta}^{[2]}(\boldsymbol{x}) \odot \boldsymbol{W}^{[1]} \cdot \boldsymbol{\Theta}^{[1]}(\boldsymbol{x}) \tag{15}$$

where $\mathbf{1}_{k\times 1}$ is a k-by-1 identity matrix (output dense layer), $\boldsymbol{W}^{[j]}$ is the weight matrix of the j-th layer, $\cdot$ represents matrix multiplication, and $\odot$ denotes Hadamard (elementwise) product.

Distinct from a multi-layer perceptron (MLP), CombOpNet does not compose the linear combination of the output of the previous layer to nonlinear activation functions. Instead, it multiplies the linear combination of the $(j-1)$-th layer's outputs with candidate functions of the j-th layer, whose input now comes from the original input data. In the course of forward propagation herein, the candidate functions of different layers are multiplied together, resulting in all possible combinations of these candidate functions being summed at the output. This way, we can significantly reduce the number of training parameters in SINDy (see Section 3.4).

3.2 Parallel CombOpNets for sparse system identification

To integrate the CombOpNet within the framework of SINDy, we parallelly stack multiple CombopNets, allocating one for each ODE, as shown in Fig.4. In other words, the result of the i-th neuron in the output layer is predicted by

$$\hat{y}_i = \mathscr{N}_i(\boldsymbol{x}),\ i = 1, 2, \cdots, n. \tag{16}$$

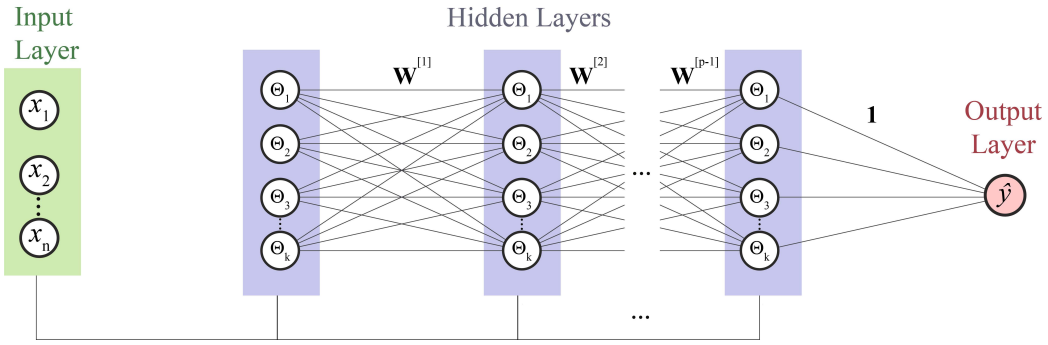

Fig. 3 The structure of Combinatorial Operation Neural Networks (CombOpsNets). The network has no bias and the composition of nonlinear activation functions. All neurons are univariate functions. The input is parallelly fed into the candidate functions of each layer.

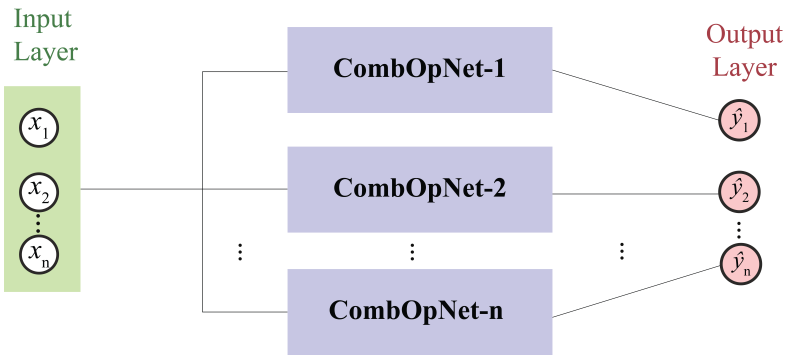

Fig. 4 Stacking multiple CombOpNets for sparse identification of nonlinear dynamics (SINDy), each of which is associated with one output dimension.

This way, we decouple the networks for different output dimensions, avoiding their cross-coupling. We define the loss function in the form

$$L = \sum_{i=1}^{m} \frac{1}{m} ||\dot{\boldsymbol{x}}^i - \mathscr{N}(\boldsymbol{x}^i)|| + R(\boldsymbol{W}) \tag{17}$$

where $\boldsymbol{W}$ represents all weight matrices, m stands for the number of samples, R denotes a sparse-promoting operator, and λ is a scaling factor.

3.3 Comparison of coefficient relations: matrix formulation vs. CombOpNet formulation

How are the coefficients related between the matrix-based formulation and CombOpNet formulation? It turns out that every CombOpNet has a corresponding matrix form of SINDy. In other words, the weights of a CombOpNet can be used to reconstruct the weights (coefficients) in the matrix formulation of SINDy. During this reconstruction, the multiplication of weights along a feedforward path corresponds to the coefficient of a composite term. As there can have multiple paths leading to the same composite term, the weights resulting from these paths need to be summed together. For example, the total weight of a composite candidate function $\Theta_{i_1}\Theta_{i_2}\dots\Theta_{i_p}$ is represented by

$$\mathscr{W}_j = \sum_{cyc} W^{[1]}_{i_1 i_2} W^{[2]}_{i_2 i_3} \cdots W^{[p-1]}_{i_{p-1} i_p}, \tag{18}$$

where $\sum_{cyc}$ represents the cyclic sum of indexes $i_1, i_2, \dots, i_n$ and the index j is defined as $j = \sum_{k=1}^{n} i_k z_k$, with z_k being a prime number in ascending order with k. More specifically, we define $\boldsymbol{z} = [2, 3, 5, \dots]$.

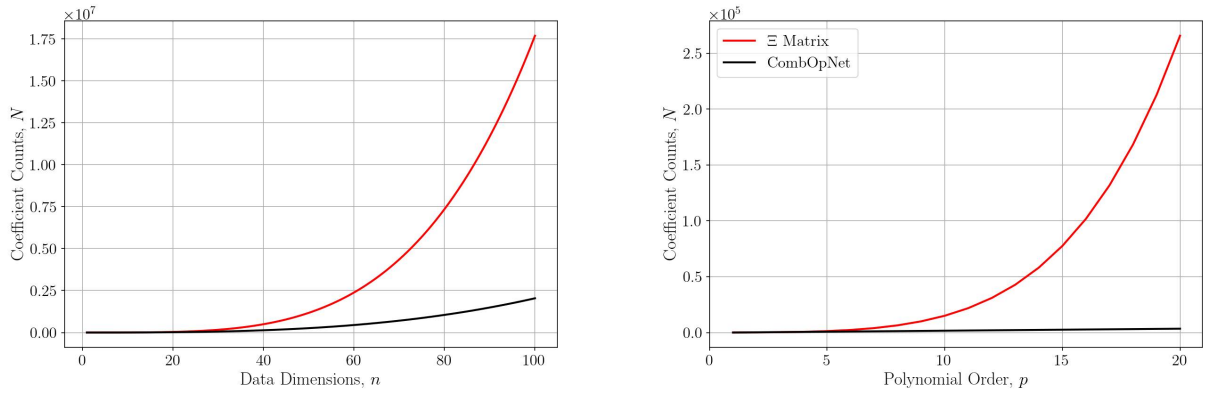

Fig. 5 Comparison of Coefficient Counts: Ξ Matrix Representation in SINDy vs. CombOpNet. (a) Data dimensions n vs. number of coefficients N, for polynomial order $p = 3$. (b) Polynomial order p vs. number of coefficients N, for data dimensions $n = 10$.

This is to ensure the uniqueness of j. At the code implementation level, j can be represented by a sorted string composed of characters corresponding to the indexes $i_1, i_2, \ldots, i_p$.

Eq. (18) indicates that the CombOpNet constructs a nonlinear mapping between its weights and the weights (coefficients) by the matrix formulation of SINDy. With this mapping, we can significantly reduce the number of weights for training, thus improving the algorithm's efficiency by several orders of magnitude.

3.4 Weights reduction by CombOpNets

In the discussion that follows next, we restrict ourselves to monomial candidate functions again. We assume that the dimension of the state variables is n, and the highest polynomial order is p. In addition, we assume the same number of neurons per layer for simplicity. The number of neurons (including the candidate function $\Theta = 1$) per layer becomes $n+1$. Therefore, the total number of trainable weights in CombOpNet reads

$$N = n(n+1)^2(p-1) \approx \mathscr{O}(n^3 p), \tag{19}$$

where $(n+1)^2$ represents the number of weights per layer, and $(p-1)$ is the number of trainable weight matrices. As N (cubically) increases with the number of state variables and is proportional to the highest order of monomials (see, Eq. (19)), it is evident that CombOpNets can significantly reduce the number of weights (coefficients) that need to be determined by an optimization algorithm compared to the matrix representation of $\boldsymbol{\Xi}$.

4 Training scheme

To yield a parsimonious model from the available data, we adopt supervised learning to minimize the loss function of Eq. (17). The training dataset is generated by uniformly sampling N points along a trajectory $\boldsymbol{x}(t)$, with $0 \leq t \leq T$. The time derivatives are computed using the vector field of a given system of ODEs (although they could be estimated by employing finite differences). The trajectory typically originates from an initial condition near an equilibrium point. Such data can be readily collected experimentally. The validation dataset is generated by extending the simulation from the termination point of the training dataset.

Algorithm 1 Training with hard thresholded pruning.

```
Initialization: Epoch, n, δ.            ▷ n denotes the dimensions, and δ represents the threshold.
Train the first m epochs without threshold.
while Epoch ≤ N do
    masks ← generate_masks(W, δ)        ▷ If a weight is less than δ, mask it to zero.
    W ← pruning_weights(W, masks)
    pred ← model(inputs)                ▷ Forward propagation
    loss ← loss_fn(outputs, pred)
    gradients ← calculate_gradients(loss, W)
    masked_gradients ← get_masked_gradients(gradients, masks)
    optimizer.apply_gradients(masked_gradients)   ▷ Backward propagation
    Epoch ← Epoch +1                    ▷ Convergence check
    for i ← 1 to n do
        if is_subnet_trainable(i) then
            if is_subnet_converged(i) then
                freeze_subnet(i)
            end if
        end if
    end for
end while
```

To ensure robust training, we shuffle the training set, thus randomizing the order of data samples. Subsequently, we divide the training set into multiple mini-batches, with a batch size of $2^7 = 128$. We employ the first-order optimizer, Adam [45], to train the model. The learning rate is set to 0.2 for the first 1000 epochs, then reduced to 0.02 from epoch 1001 to 5000, and further reduced to 0.002 after epoch 5000. The training will abort when the value of the loss function is less than 10^{-10}.

To promote the sparsity of training parameters, we adopt the sequential threshold least squares (STLSQ) method [5], which will prune the weights less than a fixed threshold to zero. However, as the trainable variables in ML libraries such as Tensorflow are immutable data structures, individually zeroing the weights below a threshold during training is not permitted. Therefore, the original STLSQ has to be adapted in order to utilize the power of such libraries.

This leads us to develop a variant of STLSQ, inspired by the pruning method for Deep Neural Networks [46]. During training, we apply a binary mask to the weight matrix of each layer. This mask, having the same shape as the layer's weight matrix, zeros out the weights below a certain threshold. This zeroing is achieved through element-wise multiplication with the binary mask. During backpropagation, we similarly multiply the gradient element-wise with the binary mask. This ensures that gradients associated with masked weights are eliminated, preventing their participation in the weight update process. The pseudo code of the training algorithm with such a pruning scheme that we developed is listed in Algorithm 1. It is important to note that we pretrain the network for a few epochs without pruning to avoid accidentally removing dominant terms before their weights exceed the threshold.

Additionally, to accelerate training, we sequentially evaluate the loss value of each network in the stack. If a subnetwork's loss value is below a certain threshold, we stop its training by freezing its trainable weights.

After training, the weights of a CombOpNet need to be transformed to determine the weights of the composite terms in the parsimonious model using Eq. (18).

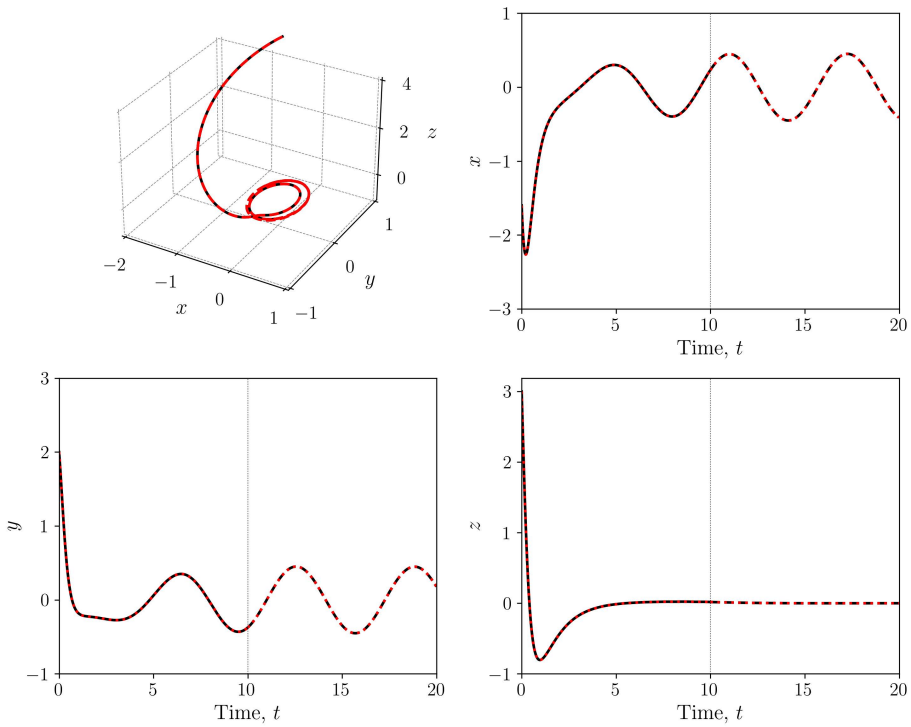

Fig. 6 Evolution of a trajectory in the mean field model. Top left: 3D trajectory. Top right: x evolution. Bottom left: y evolution. Bottom right: z evolution. Training data ($0 < t < 10$) and validation data ($10 < t < 20$). Black lines represent ground truth; red lines denote CombOpNet predictions.

5 Examples

In this section, we demonstrate the ability of CombOpNet to identify the dynamics of several nonlinear dynamical systems, ranging from dimension 3 to dimension 15. We adopt the training scheme discussed in Section 4.

5.1 Mean field model

As our first example, we consider the mean field model of [47] (see, also [5]) that was developed by the proper orthogonal decomposition (POD) [48] within the context of laminar flows around a circular cylinder. By choosing the two most energetic POD modes that represent steady-state vortex shedding, and coupling them with a shift mode that links the transient dynamics to the vortex shedding, one obtains the model

$$\begin{aligned} \frac{dx}{dt} &= \mu x - \omega y - Axz, \\ \frac{dy}{dt} &= \omega x + \mu y - Ayz, \\ \frac{dz}{dt} &= -\lambda(z - x^2 - y^2), \end{aligned} \tag{20}$$

where x and y represent the two POD-mode states, and z denotes the shift-mode state. The parameter μ affects the dynamics of the system and it connects with the Reynolds number [5, 47]. It should be noted that the above system has a quadratic nonlinearity whereas the parameters to be estimated are μ, A, ω and λ.

To do so, we select 2000 sampling points with sampling length of $T = 10$ and time-step size of 0.005. The initial condition $[2\,0\,1]^T$ is employed for training the network. For the NN configuration,

Table 1 True vs estimated parameters in the mean field model. The parameter A in x and y dimensions is trained separately. The one with the largest error is presented.

Parameter	True	Estimated
μ	0.2	0.199999973
ω	1.0	0.999999970
A	1.0	1.000000030
λ	1.0	0.999999970

we consider three parallel CombOpNets with 2 layers, and 4 neurons per layer. The final loss value is 1.41059×10^{-14}. Our numerical results using CombOpNet are shown in Fig.6. The top left panel showcases the solution in phase space, whereas the top right as well as bottom left and right panels depict the temporal evolution of the state variables (see axes labels). Moreover, the solid black lines correspond to the ground truth whereas the red lines to the predictions stemming from CombOpNet. It can be discerned from the panels that the CombOpNet accurately predicts the temporal evolution of the model which in fact asymptotes to a stable periodic orbit (also called periodic vortex shedding [47]). Moreover, CombOpNet predicts correctly the quadratic order of the polynomial associated with the nonlinearity as well as the correct parameter values are shown in Table 1, where the estimated values agree with the exact values within 10^{-8} accuracy.

5.2 Shimizu-Morioka model

In the region of high Reynolds numbers, the so-called Shimizu-Morioka model

$$\begin{aligned} \dot{x} &= Ay, \\ \dot{y} &= Ax - ay - xz, \\ \dot{z} &= -bz + Ax^2, \end{aligned} \tag{21}$$

shows a similar period-doubling bifurcation as in the Lorenz system [49]. We consider the above model that features a quadratic nonlinearity as a test example of CombOpNet where 50000 sampling points with sampling length of $T = 500$ and time-step size of 0.01 were selected upon utilizing a randomly generated initial condition $[0.21166498 \; -0.44449307 \; 0.19439383]^T$. Similar to the mean field model, we consider three parallel networks with 2 layers, and 4 neurons per layer.

Upon training CombOpNet for 10 epochs with 8.67×10^{-15} final loss function value and hard threshold of 0.1, the predicted parameters are shown in Table 2. A perfect agreement can be discerned from the table between predicted and true parameter values. Similar to the top left panel of Fig.6, the left and right panels of Fig.7 present the phase space trajectories of the model corresponding to the ground truth and CombOpNet predictions, respectively. It should be noted that the predicted trajectories experience the same behavior as the ground truth's ones. This observation is explored further in the panels of Fig.8, where the temporal evolution of the state variables is shown. Indeed, in the panels a direct comparison between ground truth (solid black line) and predictions (solid red line) is performed. Past $t = 500$, we perform a validation where CombOpNet predicts quite accurately the underlying dynamics of Eq. (21).

5.3 Triadic MHD model

Next, we consider a more complicated triad-interaction model in magnetohydrodynamics (MHD), which is an example in the PySINDy library [50] (originally proposed by Carbone and Veltri [51] in 1992). The model was developed from an incompressible 2D magnetofluid model subject to the external magnetic field. By truncating a Fourier expansion of the field variables, Carbone and Veltri derived 6-dimensional

Table 2 True vs estimated parameters in the Shimizu-Morioka Model. A in each dimension is trained separately. The one with the maximized error is presented.

Parameter	True	Estimated
a	0.81	0.80999973
b	0.375	0.37487731
A	1.0	0.99999997

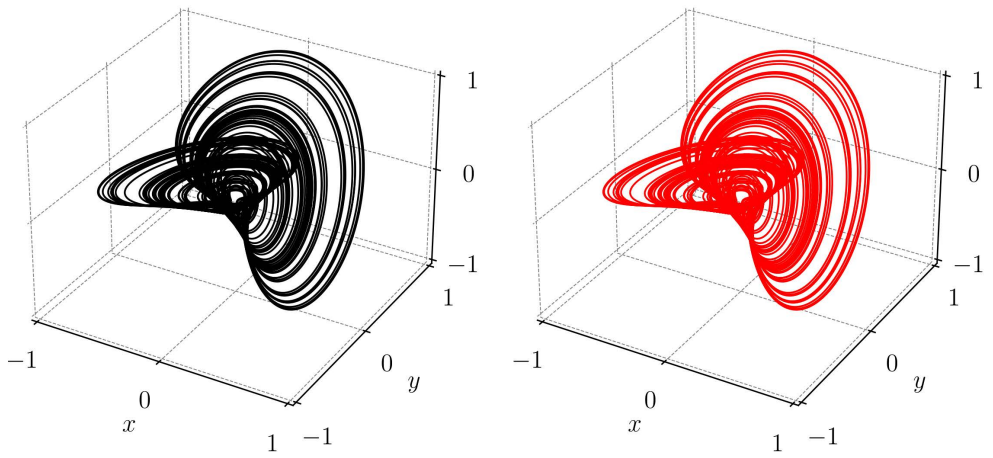

Fig. 7 Evolution of phase trajectories of the Shimizu-Morioka model. Left panel (black): ground truth. Right panel (red): trajectory of the identified model.

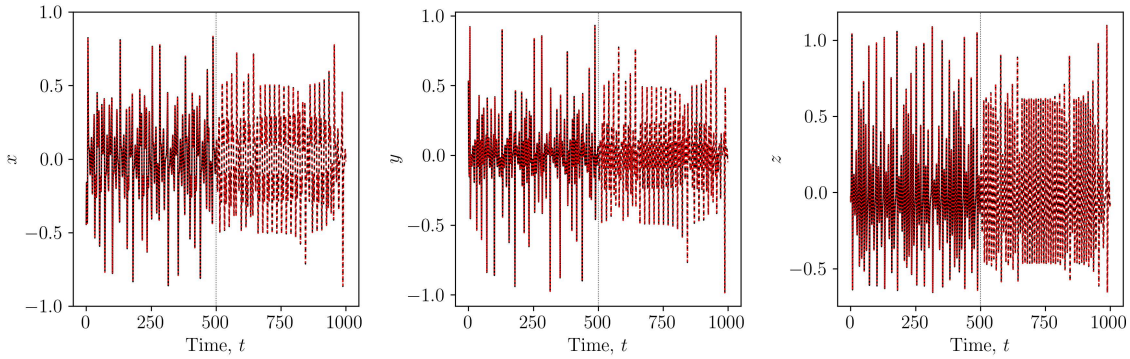

Fig. 8 The time evolution of state variables of the Shimizu-Morioka model. Left: x, middle: y, right: z. The vertical dash line at $t = 500$ separates the training (solid line) and validation (dash line) data. Again, the red represents the trajectory of the estimated model, while the black denotes the ground truth.

system of nonlinear ODEs from the original MHD equation that reads:

$$
\begin{aligned}
\dot{x}_0 &= -2\nu x_0 + 4\left(x_1x_2 - x_4x_5\right), \\
\dot{x}_1 &= -5\nu x_1 - 7\left(x_0x_2 - x_3x_5\right), \\
\dot{x}_2 &= -9\nu x_2 + 3\left(x_0x_1 - x_3x_4\right), \\
\dot{x}_3 &= -2\mu x_3 + 2\left(x_4x_1 - x_2x_3\right), \\
\dot{x}_4 &= -5\mu x_4 + \sigma x_5 + 5\left(x_2x_3 - x_0x_5\right), \\
\dot{x}_5 &= -9\mu x_5 + \sigma x_4 + 9\left(x_3x_0 - x_1x_3\right),
\end{aligned}
\tag{22}
$$

Table 3 True vs estimated parameters in the MHD model. The constants also match precisely. The parameters in each dimension are trained independently. Only the parameters with the largest errors are presented in the table.

Parameter	True	Estimated
ν	0.0	0.00000000
μ	0.0	0.0000000000
σ	3.0	3.000000000

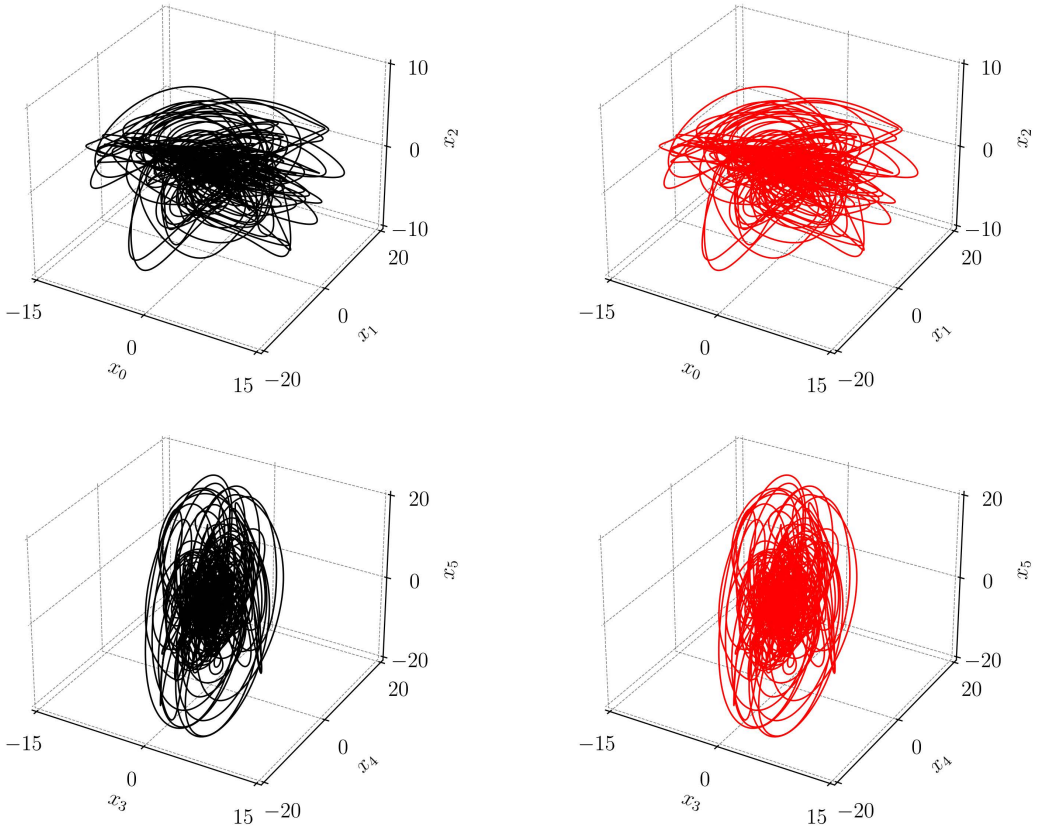

Fig. 9 Phase trajectories projected in 3D spaces. Top panel (velocity field): x_0-x_2. Bottom panel (magnetic field): x_3-x_5. Left panel: ground truth. Right panel: trajectory predicted by the identified model.

where x_0, x_1, and x_2 correspond to the Fourier amplitudes of a triad of wave vectors of the velocity field, and x_3, x_4, and x_5 represent the corresponding amplitudes of the magnetic field (see [51]). In addition, the parameter σ represents external forcing, while ν and μ denote viscosity and magnetic diffusivity, respectively.

We sample a trajectory of the system originating from a randomly generated initial condition. The simulation time is $T = 50$, and we collect 50,000 discrete points with a uniform time step of 0.001. We use six parallel CombOpNets, each with 2 layers and 7 neurons per layer.

To train the parallel networks, we adopt a hard threshold of 0.15. With 5 epochs of pretraining (no pruning), the networks converge with a total loss value of 2.05×10^{-13}. This is partly contributed by the relatively large sampling numbers. With the mini-batch size of 128 (see Sec. 4), 319 stochastic gradient descents are applied in a single epoch. During training, the constants are treated as unknowns and are identified as well. Their values are identical to the exact values up to 10^{-9}. This perfect match is observed in those parameters tabulated in Table 3. The excellent agreement between the ground truth

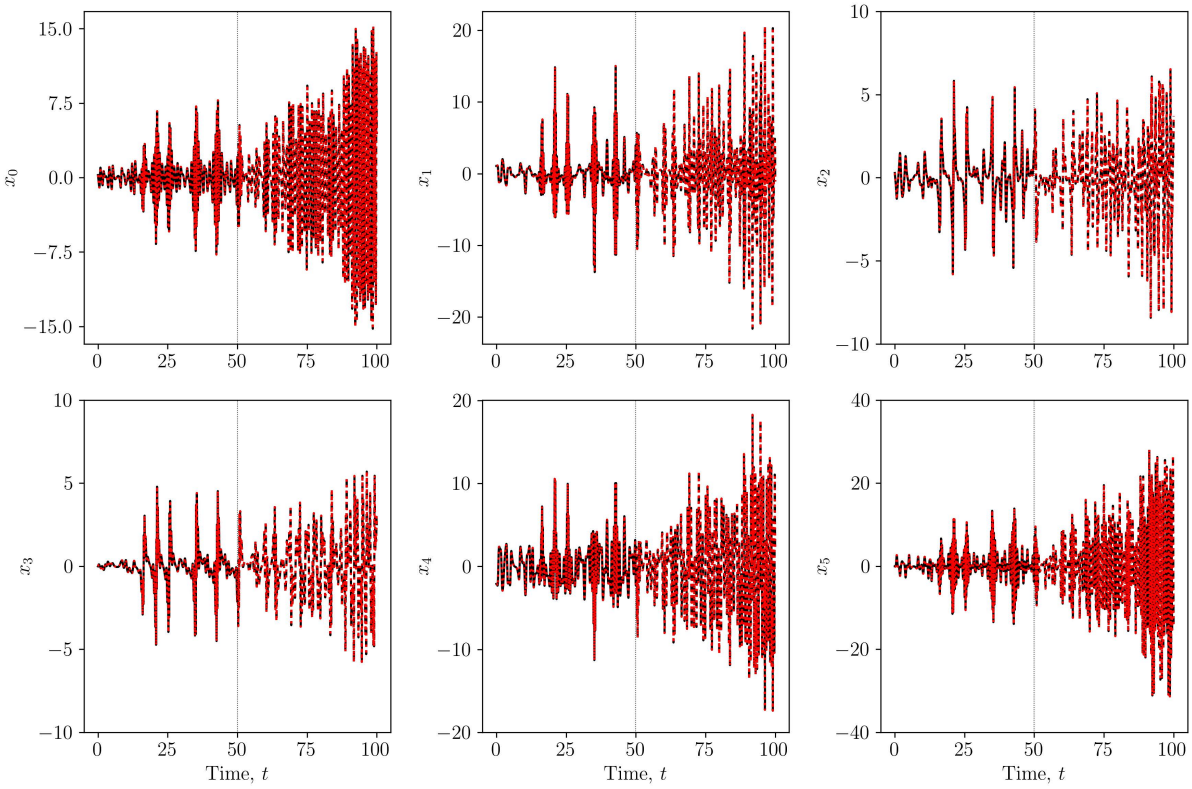

Fig. 10 Time response of state variables x_0-x_5 in the MHD model. The black curve represents the ground truth, while the red curve shows the estimated trajectory from the identified system. The solid line indicates the training set, and the dashed line represents the validation set.

(black solid line) and prediction (red solid line) is illustrated in Fig.9, where the top panel shows the evolution of state variables of the velocity field, and the bottom panel illustrates that of the magnetic field. In Fig.10, we further extend the time span to validate the prediction of the identified system. It can be seen that long-term accurate prediction can be achieved by the parallel CombOpNets.

5.4 Lorenz-96 with $n = 15$

Having studied the four examples above, we are now ready to apply the parallel CombOpNets to the Lorenz-96 [52] system with higher dimensions. The Lorenz-96 system is a one-dimensional atmospheric model that divides the earth latitude circle into n sectors. The dynamics of the j-th sector is characterized by

$$\dot{x}_j = x_{j-1}(Ax_{j+1} - Bx_{j-2}) - Cx_j + F, \quad j = 1, 2, \ldots, n, \tag{23}$$

where $x_{-1} = x_{n-1}$ and $x_0 = x_n$, $x_{n+1} = x_1$. Herein, the state variable x_j represents the temperature of the j-th sector. The linear terms Cx_j and F denote internal dissipation and external constant forcing (e.g. solar heating), respectively. The quadratic terms simulate advection between two adjacent sectors.

We choose $n = 15$ so that each sector covers 24 degrees of latitude. As our goal is to validate the sparse identification algorithm with CombOpNets, we simply use a random initial condition near the origin. We sample 50,000 points again with a uniform time step 0.001. In other words, the time span of data is $T = 50$. Each CombOpNet in the parallel networks is configured to possess 2 layers and 16 neurons per layer. We adopt a hard threshold value of 0.09. We train the parallel networks for 30 epochs, with 5 epochs of pretraining. The final loss value stops at 5.34×10^{-9}.

The identified nonlinear system has identical terms to the original one. The estimated parameters

Table 4 True vs estimated parameters in the Lorenz-96 model. Note that the parameters in each dimension are trained independently. The parameters with the largest errors are presented in the table.

Parameter	True	Estimated
A	1.0	0.99999949
B	1.0	1.00000006
C	1.0	1.0000052
F	8.0	7.99995995

are tabulated in Table 4, precisely aligning with the exact values. Fig.11 presents the phase trajectories projected to two-dimensional subplanes formed by two adjacent states, where the solid blue curve represents the trajectory of the original system, while the red dash curve denotes the evolution of states of the identified system. Perfect agreement between them is observed. Again, to further examine the ability of extrapolation of CombOpNets, we use the states at the final time in the training set as the initial condition for the validation set. We rerun numerical simulation for the same duration as the training set, and compare the results produced by the identified model by CombOpNets, as shown in Fig.12. It can be seen that the identified model can also nicely match the trajectory of the original system at least for the same duration, indicating the great accuracy of the identified model.

6 Conclusions and future directions

The development of data-driven methods and their application to timely complex nonlinear systems have admittedly propelled the efforts forward in understanding complex nonlinear phenomena. Indeed, the Sparse Identification of Nonlinear Dynamics (SINDy) method has enjoyed success on that regard, and has sparked a great interest in extending and improving the method. In the present work, we attempt to accelerate SINDy by introducing the compact neural-network (NN) architecture that we name Combinatorial Operation Neural Network or CombOpNet. The present architecture represents nonlinear terms of a dynamical system as a chain of multiplication of univariate functions that themselves correspond to neurons of a multi-layer NN. We demonstrate that such a NN allows the construction of nonlinear terms in dynamical equations by virtue of forward propagations. We argue that CombOpNet uses far fewer weights, thus making it quite efficient when the dimension of a dynamical system is quite large. With that in mind, we performed a series of tests of CombOpNet, starting from dynamical systems of dimension 3, and gradually moving to ones with 6, and 15. In each case example, CombOpNet predicts quite accurately the dynamics of the pertinent models while demonstrating its efficiency over SINDy.

The present work opens up new directions of study that we aim to pursue. On the one hand, it was mentioned that path multiplicity allows for opening alternative paths in CombOpNet when one or more paths are diminished. This is true when the sparsity assumption is valid although issues arise when it is not. We thus plan to explore the existence of an upper limit of sparsity that the NN can feature. On the other hand, the striking feature of CombOpNet applied to high dimensional systems may open up the possibility of utilizing it to not only discrete lattice systems but also to PDEs as an alternative data-driven method. To achieve this goal, a new pruning method should be developed. As hard-threshold pruning can accidentally prune dominant terms before they exceed the threshold during gradient descent, careful tuning is required in order to adjust the threshold and select the epochs for pre-training, a task that may become quite challenging when scaling to higher dimensions. In addition, if the high-dimensional system exhibits distinct forms in different dimensions, this pruning scheme can become particularly tedious. In this case, a dimension-wise pruning scheme with different hyperparameters is planned to be considered and integrated in the CombOpNet framework. At last,

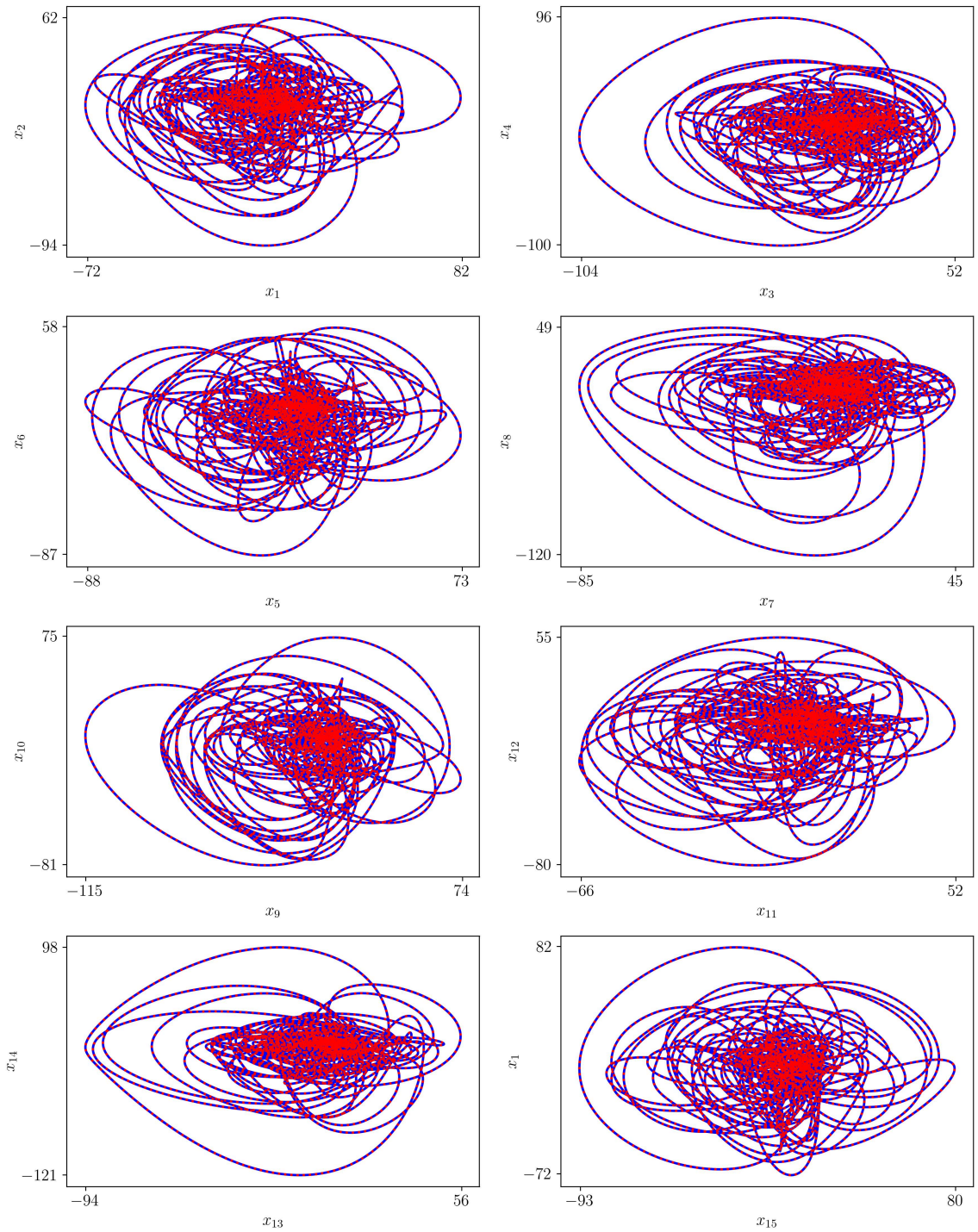

Fig. 11 Phase trajectories of two adjacent state variables in the Lorenz-96 system.

we would like to relay again the central thrust of our present work: CombOpNet is able to *accelerate* SINDy. Although we have not evaluated the performance of the network under the presence of noise at the moment, this important aspect is planned to be studied next. These are directions we plan to pursue in the future, and results will be reported in forthcoming works.

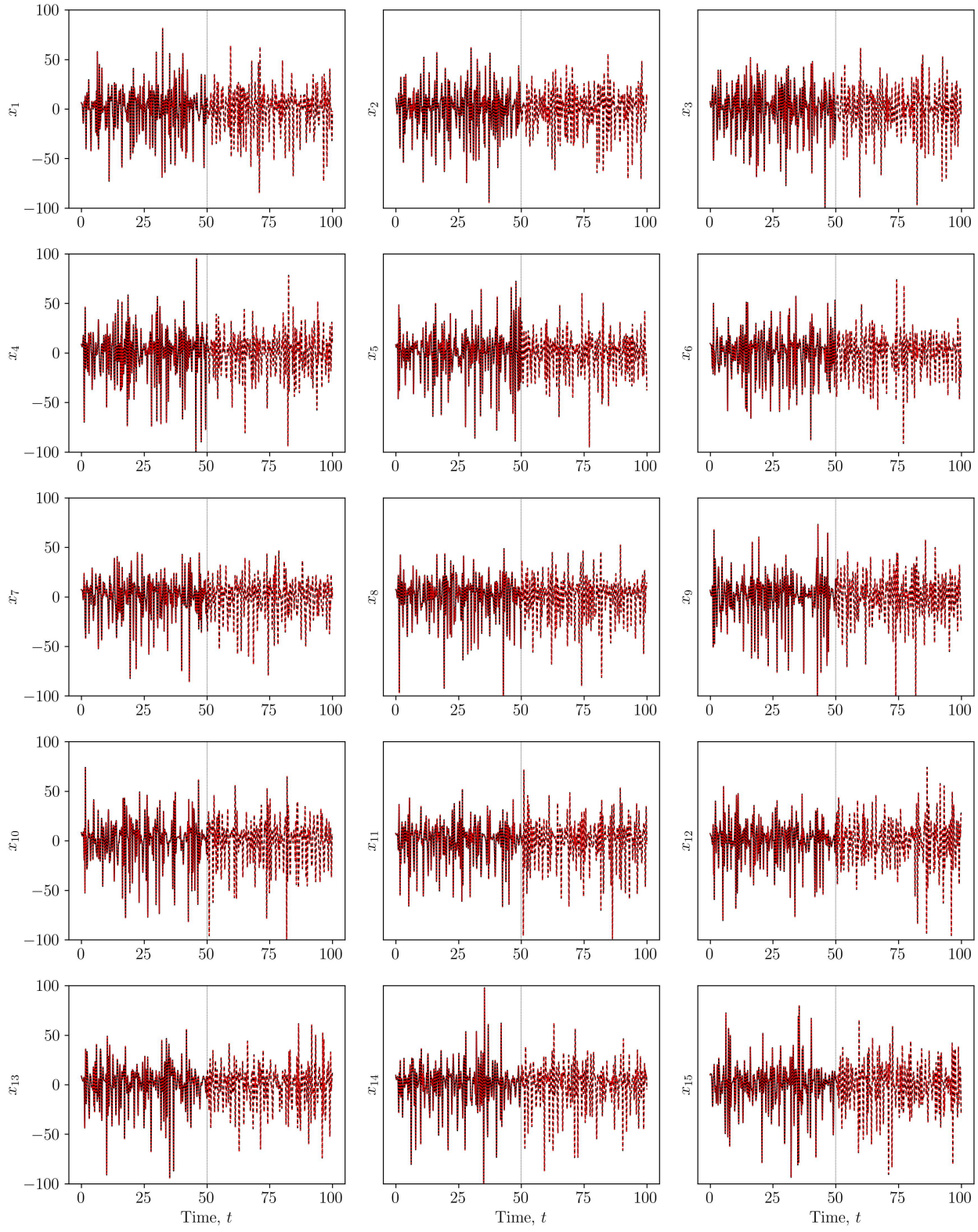

Fig. 12 Time response of state variables in the Lorenz-96 system.

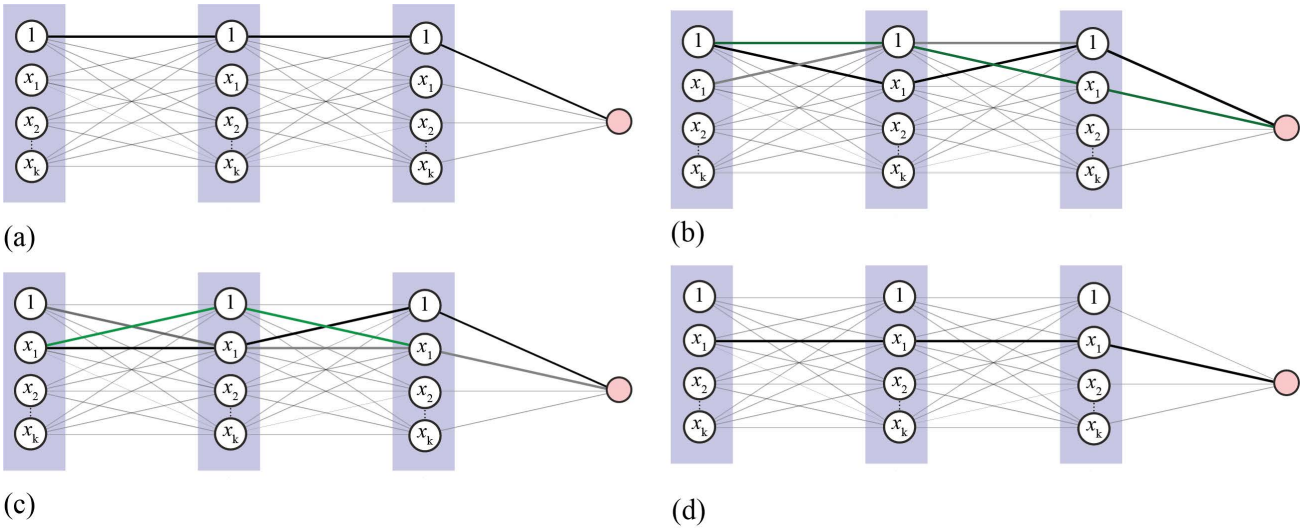

Fig. 13 Path multiplicity of CombOpNet. Multiple paths (highlighted) are associated with all terms except the constant term (a) and those with the highest orders (d). Path for (a) the constant term, (b) x_1, (c) x_1^2, and (d) x_1^3.

Acknowledgement

This material is based upon work supported by the Research, Scholarly & Creative Activities (RSCA) Program awarded by the Cal Poly division of Research, Economic Development & Graduate Education and Chrones Endowed Professorship (SX), and the US National Science Foundation under Grant No. DMS-2204782 (EGC). SX thanks Y.C. Lai and B.G. Anderson for useful discussions.

Appendix

A.1 Path multiplicity

Due to the scarcity of nonlinear terms, abundant weights in the CombOpNet will eventually diminish. When a weight becomes zero, all paths containing it will practically depreciate. One might naturally speculate that this could inadvertently eliminate the path of dominant terms. However, the structure of CombOpNet can accommodate this issue due to its path multiplicity. More specifically, more than one path in CombOpNet could lead to the same composite nonlinear term.

We demonstrate this feature through an example with a polynomial order of 3, in Fig.13. Except for the constant term in Fig.13(a) and the terms with the highest order in Fig.13(d) having unique paths, the remaining terms (e.g., the highlighted linear ones in Fig.13(b) and the highlighted quadratic ones in Fig.13(c)) all possess multiple paths. This path multiplicity allows for opening alternative paths when one or more paths are diminished. The path uniqueness of the highest-order terms can be remedied by adding one extra layer to increase the highest possible order by one. The reduction of the path containing all ones will not cause the paths of any other terms to diminish. In general, adding more layers will increase the number of available paths. We report however that a problem still arises when the sparsity assumption is no longer valid. The investigation into the upper limit of sparsity that the neural network can accommodate will be pursued in a forthcoming paper.

References

[1] Brunton, S.L. and Kutz, J.N. (2019), *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, Cambridge University Press.

[2] Kutz, J.N., Brunton, S.L., Brunton, B.W., and Proctor, J.L. (2016), *Dynamic Mode Decomposition*, Society for Industrial and Applied Mathematics.

[3] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021), Physics-informed machine learning, *Nature Reviews Physics*, **3**, 422-440.

[4] Goodfellow, I., Bengio, Y., and Courville, A. (2016), *Deep Learning*, MIT Press.

[5] Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016), Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proceedings of the National Academy of Sciences*, **113**, 3932-3937.

[6] Guruwacharya, N., Chakraborty, S., Saraswat, G., Bryce, R., Hansen, T.M., and Tonkoski, R. (2024), Data-driven modeling of grid-forming inverter dynamics using power hardware-in-the-loop experimentation, *IEEE Access*, **12**, 52267-52281.

[7] Katsoulakis, M.A. and Vilanova, P. (2020), Data-driven, variational model reduction of high-dimensional reaction networks, *Journal of Computational Physics*, **401**, 108997.

[8] Song, J., Tong, G., Chao, J., Chung, J., Zhang, M., Lin, W., Zhang, T., Bentler, P.M., and Zhu, W. (2023), Data driven pathway analysis and forecast of global warming and sea level rise, *Scientific Reports*, **13**, 5536.

[9] Beam, E., Potts, C., Poldrack, R.A., and Etkin, A. (2021), A data-driven framework for mapping domains of human neurobiology, *Nature Neuroscience*, **24**, 1733-1744.

[10] Bongard, J. and Lipson, H. (2007), Automated reverse engineering of nonlinear dynamical systems, *Proceedings of the National Academy of Sciences*, **104**, 9943-9948.

[11] Schmidt, M. and Lipson, H. (2009), Distilling free-form natural laws from experimental data, *Science*, **324**, 81-85.

[12] Udrescu, S.M. and Tegmark, M. (2020), Ai feynman: A physics-inspired method for symbolic regression, *Science Advances*, **6**(16), eaay2631 (16 pages).

[13] Petersen, B.K., Landajuela, M., Mundhenk, T.N., Santiago, C.P., Kim, S.K., and Kim, J.T. (2021), Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients, *Proceeding of the International Conference on Learning Representations*, 1-26.

[14] Donoho, D.L. (2006), Compressed sensing, *IEEE Transactions on Information Theory*, **52**, 1289-1306.

[15] Baraniuk, R.G. (2007), Compressive sensing [lecture notes], *IEEE Journals & Manazine*, **24**, 118-121.

[16] Wang, W.X., Yang, R., Lai, Y.C., Vassilios, K., and Grebogi, C. (2011), Predicting catastrophes in nonlinear dynamical systems by compressive sensing, *Physical Review Letters*, **106**, 154101 (4 pages).

[17] Zhang, L. and Schaeffer, H. (2019), On the convergence of the SINDy algorithm, *Multiscale Modeling & Simulation*, **17**, 948-972.

[18] Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016), Sparse identification of nonlinear dynamics with control (sindyc), *IFAC-PapersOnLine*, **49**, 710-715.

[19] Rudy, S.H., Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016), Data-driven discovery of partial differential equations, *IFAC-PapersOnLine*, **49**, 710-715.

[20] Rudy, S., Alla, A., Brunton, S.L., and Kutz, J.N. (2019), Data-driven identification of parametric partial differential equations, *SIAM Journal on Applied Dynamical Systems*, **18**, 643-660.

[21] Messenger, D.A. and Bortz, D.M. (2021), Weak SINDy for partial differential equations, *Journal of Computational Physics*, **443**, 110525.

[22] Mangan, N.M., Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016), Inferring biological networks by sparse identification of nonlinear dynamics, *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, **2**, 52-63.

[23] Kaheman, K., Kutz, J.N., and Brunton, S.L. (2020), SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **476**, 20200279 (25 pages).

[24] Jacobs, M., Brunton, B.W., Brunton, S.L., Kutz, J.N., and Raut, R.V. (2023), HyperSINDy: Deep generative modeling of nonlinear stochastic governing equations.

[25] Hoffmann, M., Fröhner, C., and Noé, F. (2019), Reactive SINDy: Discovering governing reactions from concentration data, *The Journal of Chemical Physics*, **150**, 025101.

[26] Kaiser, E., Kutz, J.N., and Brunton, S.L. (2018), Sparse identification of nonlinear dynamics for model predictive control in the low-data limit, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **474**, 20180335 (25 pages).

[27] Zolman, N., Fasel, U., Kutz, J.N., and Brunton, S.L. (2024), RL: Interpretable and efficient model-based reinforcement learning, arXiv:2403.09110, ICS.

[28] Fukami, K., Murata, T., Zhang, K., and Fukagata, K. (2021), Sparse identification of nonlinear dynamics with low dimensionalized flow representations, *Computer Methods in Applied Mechanics and Engineering*, **926**, A10 (35 pages).
[29] Prokop, B. and Gelens, L. (2023), Data-driven discovery of oscillator models using SINDy: Towards the application on experimental data in biology, *bioRxiv*, pp.2023-08.
[30] Bramburger, J.J. and Kutz, J.N. (2020), Poincaré maps for multiscale physics discovery and nonlinear floquet theory, *Physica D: Nonlinear Phenomena*, **408**, 132479.
[31] Bramburger, J.J., Dylewsky, D., and Kutz, J.N. (2020), Sparse identification of slow timescale dynamics, *Physical Review E*, **102**, 022204.
[32] Saqlain, S., Zhu, W., Charalampidis, E.G., and Kevrekidis, P.G. (2023), Discovering governing equations in discrete systems using PINNs, *Communications in Nonlinear Science and Numerical Simulation*, **13**, 107498 (12 pages).
[33] Zheng, P., Askham, T., Brunton, S.L., Kutz, J.N., and Aravkin, A.Y. (2019), A unified framework for sparse relaxed regularized regression: SR3, *IEEE Access*, **7**, 1404-1423.
[34] Cortiella, A., Park, K.C., and Doostan, A. (2021), Sparse identification of nonlinear dynamical systems via reweighted l_1-regularized least squares. *Computer Methods in Applied Mechanics and Engineering*, **376**, 113620 (31 pages).
[35] Naozuka, G.T., Rocha, H.L., Silva, R.S., and Almeida, R.C. (2022), SINDy-SA framework: Enhancing nonlinear system identification with sensitivity analysis, *Nonlinear Dynamics*, **110**, 2589-2609.
[36] Cortiella, A., Park, K.C., and Doostan, A. (2022), A priori denoising strategies for sparse identification of nonlinear dynamical systems: A comparative study, *Journal of Computing and Information Science in Engineering*, **23**, 01004 (23 pages).
[37] Egan, K., Li, W., and Carvalho, R. (2024), Automatically discovering ordinary differential equations from data with sparsee regression, *Communications Physics*, **7**.
[38] Chen, Z., Liu, Y., and Sun, H. (2021), Physics-informed learning of governing equations from scarce data, *Nature Communications*, **12**, 6136 (13 pages).
[39] Kaheman, K., Brunton, S.L., and Kutz, J.N. (2022), Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data, *Machine Learning: Science and Technology*, **3**, 015031 (28 pages).
[40] Hirsh, S.M., Barajas-Solano, D.A., and Kutz, J.N. (2022), Sparsifying priors for bayesian uncertainty quantification in model discovery, *Royal Society Open Science*, **9**, 211823 (20 pages).
[41] Zhang, S. and Lin, G. (2018), Robust data-driven discovery of governing physical laws with error bars, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **474**.
[42] Gelß, P., Klus, S., Eisert, J., and Schütte, C. (2019), Multidimensional approximation of nonlinear dynamical systems, *Journal of Computational and Nonlinear Dynamics*, **14**, 061006 (15 pages).
[43] Champion, K., Lusch, B., Kutz, J.N., and Brunton, S.L. (2019), Data-driven discovery of coordinates and governing equations, *Proceedings of the National Academy of Sciences*, **116**, 22445-22451.
[44] Tibshirani, R. (1996), Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, **58**, 267-288.
[45] Kingma, D.P. and Ba, J. (2015), Adam: A method for stochastic optimization, *ICLR (Poster)*.
[46] Zhu, M. and Gupta, S. (2018), To prune, or not to prune: Exploring the efficacy of pruning for model compression, *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, OpenReview.net.
[47] Noack, B.R., Afanasiev, K., Morzyński, M., Tadmor, G., and Thiele, F. (2003), A hierarchy of low-dimensional models for the transient and post-transient cylinder wake, *Journal of Fluid Mechanics*, **497**, 335–363.
[48] Holmes, P., Lumley, J.L., and Berkooz, G. (1996), *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge Monographs on Mechanics, Cambridge University Press.
[49] Shimizu, T. and Moroika, N. (1980), On the bifurcation of a symmetric limit cycle to an asymmetric one in a simple model, *Physics Letters A*, **76**, 201-204.
[50] de Silva, B., Champion, K., Quade, M., Loiseau, J.C., Kutz, J., and Brunton, S. (2020), Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data, *Journal of Open Source Software*, **5**, 2104.
[51] Carbone, V. and Veltri, P. (1992), Relaxation processes in magnetohydrodynamics - a triad-interaction model, *Astronomy and Astrophysics*, **259**, 359-372.
[52] Lorenz, E. (1995), Predictability: a problem partly solved, *Seminar on Predictability, 4-8 September*, **1**, 1-18.