Proceedings of Machine Learning Research vol 145:1–21, 2022 3rd Annual Conference on Mathematical and Scientific Machine Learning

Structure-preserving Sparse Identification of Nonlinear Dynamics for Data-driven Modeling

Kookjin Lee KOOKJIN.LEE@ASU.EDU
School of Computing and Augmented Intelligence, Arizona State University

Nathaniel Trask NATRASK@SANDIA.GOV
Center for Computing Research, Sandia National Laboratories

Panos Stinis PANAGIOTIS.STINIS@PNNL.GOV
Pacific Northwest National Laboratory

Editors: Bin Dong, Qianxiao Li, Lei Wang, Zhi-Qin John Xu

Abstract

Discovery of dynamical systems from data forms the foundation for data-driven modeling and recently, structure-preserving geometric perspectives have been shown to provide improved forecasting, stability, and physical realizability guarantees. We present here a unification of the Sparse Identification of Nonlinear Dynamics (SINDy) formalism with neural ordinary differential equations. The resulting framework allows learning of both "black-box" dynamics and learning of structure preserving bracket formalisms for both reversible and irreversible dynamics. We present a suite of benchmarks demonstrating effectiveness and structure preservation, including for chaotic systems.

Keywords: System Identification, Structure preservation, GENERIC, neural ordinary differential equations

1. Introduction

A number of scientific machine learning (ML) tasks seek to discover a dynamical system whose solution is consistent with data (e.g. constitutive modeling Patel et al. (2020); Karapiperis et al. (2021); Ghnatios et al. (2019); Masi et al. (2021), reduced-order modeling Chen et al. (2021); Lee and Carlberg (2021); Wan et al. (2018), physics-informed machine learning Karniadakis et al. (2021); Wu et al. (2018), and surrogates for performing optimal control Alexopoulos et al. (2020)). A major challenge for this class of problems is the preservation of both numerical stability and physical realizability when performing out of distribution inference (i.e. extrapolation/forecasting). Unlike traditional ML for e.g. image/language processing tasks, engineering and science models pose strict requirements on physical quantities to guarantee properties such as conservation, thermodynamic consistency and well-posedness of resulting models Baker et al. (2019). These structural constraints translate to desirable mathematical properties for simulation, such as improved numerical stability and accuracy, particularly for chaotic systems Lee et al. (2021); Trask et al. (2020).

While so-called physics-informed ML (PIML) approaches seek to impose these properties by imposing soft physics constraints into the ML process, many applications require structure preservation to hold exactly; PIML requires empirical tuning of weighting parameters and physics properties hold only to within optimization error, which typically may be large Wang et al. (2020); Rohrhofer et al. (2021). Structure-preserving machine learning has emerged as a means of designing architectures such that physics constraints hold exactly by construction Lee et al. (2021); Trask et al.

(2020). By parameterizing relevant geometric or topological structures, researchers obtain more data-efficient hybrid physics/ML architectures with guaranteed mathematical properties.

In this work we consider geometric structure preservation for dynamical systems Hairer et al. (2006); Marsden and Ratiu (1995). Reversible bracket formalisms (e.g. Hamiltonian/Lagrangian mechanics) have been shown effective for learning reversible dynamics Toth et al. (2019); Cranmer et al. (2020); Lutter et al. (2018); Chen et al. (2019); Jin et al. (2020); Tong et al. (2021); Zhong et al. (2021); Chen and Tao (2021); Bertalan et al. (2019), while dissipative metric bracket extensions provide generalizations to irreversible dynamics in the metriplectic formalism Lee et al. (2021); Desai et al. (2021); Hernández et al. (2021); Yu et al. (2020); Zhong et al. (2020). Accounting for dissipation in this manner provides important thermodynamic consistency guarantees: namely, discrete versions of the first and second law of thermodynamics along with a fluctuation-dissipation theorem for thermal systems at microscales. Training these models is however a challenge and leads to discovery of non-interpretable *Casimirs* (generalized entropy and energy functionals describing the system).

For $\boldsymbol{x}(t) \in \mathbb{R}^n$ denoting the state of a system at time t, $\boldsymbol{f}$ denoting a velocity with potentially exploitable mathematical structure, and Θ denoting trainable parameters, we consider learning of autonomous dynamics of the form

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}_\Theta(\boldsymbol{x}).$$

We synthesize the Sparse Identification of Nonlinear Dynamics (SINDy) formalism Brunton et al. (2016) with neural ordinary differential equations (NODEs) Chen et al. (2018) to obtain a framework for learning dynamics. This sparse dictionary-learning formalism allows learning of either interpretable $\boldsymbol{f}$ when learning "black-box" ODEs, or for learning more complicated bracket dynamics for $\boldsymbol{f}$ which describe structure-preserving reversible and irreversible systems. Our main technical contributions include:

- a novel extension of SINDy into NODE settings, including a training strategy with L1 weight decay and pruning,
- the first application of SINDy with structure-preservation, including: black-box, Hamiltonian, GENERIC, and port-Hamiltonian formulations, and
- empirical demonstration on the effectiveness of the proposed algorithm for a wide array of dynamics spanning: reversible, irreversible, and chaotic systems.

2. Sparse Identification of Nonlinear Dynamics (SINDy)

The Sparse Identification of Nonlinear Dynamics (SINDy) method aims to identify the dynamics of interest using a sparse set of dictionaries such that

$$\boldsymbol{f}_\Theta(\boldsymbol{x}) = (\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T}\Xi)^\mathsf{T},$$

where $\Xi \in \mathbb{R}^{p\times n}$ is the coefficient matrix and $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{1\times p}$ denotes a "library" vector consisting of candidate functions, e.g., constant, polynomials, and so on.

From measurements of states (and derivatives), we can construct a linear system of equations:

$$\dot{X} = \Phi(X)\Xi$$

where $X \in \mathbb{R}^{m\times n}$ and $\dot{X} \in \mathbb{R}^{m\times n}$ are collections of the states $\boldsymbol{x}(t)$ and the (numerically approximated) derivatives $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$ sampled at time indices $\{t_1, \ldots, t_m\}$ such that

$$[X]_{ij} = x_j(t_i), \quad \text{and} \quad [\dot{X}]_{ij} = \frac{\mathrm{d}x_j}{\mathrm{d}t}(t_i).$$

A library, $\Phi(X) \in \mathbb{R}^{m\times p}$, consists of candidate nonlinear functions, e.g., constant, polynomial, and trigonometric terms:

$$\Phi(X) = \begin{bmatrix} \mathbb{1} & X & \mathbb{P}_2(X) & \cdots & \cos(X) & \cdots \end{bmatrix},$$

where $\mathbb{P}_2$ is a function of the quadratic nonlinearities such that the ith row of $\mathbb{P}_2(X)$ is defined as, for example, with $n = 3$,

$$[\mathbb{P}_2(X)]_i^{\mathsf{T}} = [x_1^2(t_i), x_1(t_i)x_2(t_i), x_1(t_i)x_3(t_i), x_2^2(t_i), x_2(t_i)x_3(t_i), x_3^2(t_i)].$$

Lastly, $\Xi \in \mathbb{R}^{p\times n}$ denotes the collection of the unknown coefficients $\Xi = [\xi_1, \ldots, \xi_n]$, where $\xi_i \in \mathbb{R}^p$, $i = 1, \ldots, n$ are sparse vectors.

Typically, only X is available and $\dot{X}$ is approximated numerically. Thus, in SINDy, $\dot{X}$ is considered to be noisy, which leads to a new formulation:

$$\dot{X} = \Phi(X)\Xi + \eta Z,$$

where Z is a matrix of independent identically distributed unit normal entries and η is noise magnitude. To compute the sparse solution Ξ, SINDy employs either the least absolute shrinkage and selection operator (LASSO) Tibshirani (1996) or sequential threshold least-squares method Brunton et al. (2016). LASSO is an L1-regularized regression method and the sequential threshold least-squares method is an iterative algorithm which repetitively zeroes out entries of Ξ and solves least-squares problems with remaining entries of Ξ.

3. Neural SINDy

The original SINDy requires the derivatives of the state variable either empirically measured or numerically computed. To avoid such requirement, we propose to use the training algorithm introduced in neural ordinary differential equations (NODEs) with L1 weight decay. In addition, we propose the magnitude-based pruning strategy to retain only the terms with significant contributions.

3.1. Neural Ordinary Differential Equations

NODEs Chen et al. (2018) are a family of deep neural network models that parameterize the time-continuous dynamics of hidden states $\boldsymbol{x}(t)$ using a system of ODEs:

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{f}_\Theta(\boldsymbol{x}),$$

where $\boldsymbol{x}(t)$ is a time-continuous representation of a state, $\boldsymbol{f}_\Theta$ is a parameterized (trainable) velocity function, and Θ is a set of neural network weights and biases. NODEs allows a memory-efficient training via the adjoint-sensitiviy method Pontryagin (1987), where the same idea of applying the adjoint method to inverse problems of imaging can be found in Oberai et al. (2003).

In the forward pass, given the initial condition $\boldsymbol{x}(0)$, a hidden state at any time index can be obtained by solving the initial value problem (IVP). A black-box time integrator can be employed to compute the hidden states with the desired accuracy:

$$\tilde{\boldsymbol{x}}(t_1), \ldots, \tilde{\boldsymbol{x}}(t_m) = \text{ODESolve}(\boldsymbol{x}(0), \boldsymbol{f}_\Theta, t_1, \ldots, t_m).$$

The model parameters Θ are then updated with an optimizer, which minimizes a loss function measuring the difference between the output and the target variables.

Dictionary-based parameterization As in the original SINDy formulation, we assume that there is a set of candidate nonlinear functions to represent the dynamics

$$\boldsymbol{f}_\Theta(\boldsymbol{x}) = (\boldsymbol{\phi}(\boldsymbol{x})^\mathsf{T} \Xi)^\mathsf{T}, \tag{1}$$

where, again, $\Xi \in \mathbb{R}^{p \times n}$ is a trainable coefficient matrix and $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{p \times 1}$ denotes a library vector. In this setting, the learnable parameters of NODE become $\Theta = \Xi$.

As presented in Sahoo et al. (2018), this parameterization can be seen as a multi-layer feed-forward neural network with hard-coded weights and particular nonlinear activations such as multiplication (e.g., $z_1 z_2$), powers (e.g., z_1^2), and so on. As a potential extension, those hard-coded weights can be modeled as learnable weights following Sahoo et al. (2018).

Sparsity inducing loss: L1 weight decay (Lasso) To induce sparsity in Ξ we add an L1 weight decay to the main loss objective for the mean absolute error:

$$\mathcal{L} = \frac{1}{n_{\text{train}}} \sum_{\ell=1}^{n_{\text{train}}} \sum_{i=1}^{m} \left\| \boldsymbol{x}^{(\ell)}(t_i) - \tilde{\boldsymbol{x}}^{(\ell)}(t_i) \right\|_1 + \lambda \left\| \Xi \right\|_1, \tag{2}$$

where n_{train} denotes the number of training data instances (trajectories), and $\|X\|_1 = \sum_{l,k} |[X]_{lk}|$.

Pruning Taking linear combinations of candidate functions in Eq. (1) admits implementation as a linear layer, specified by Ξ, which does not have biases or nonlinear activation. To further sparsify the coefficient matrix Ξ, we employ the magnitude-based pruning strategy:

$$[\Xi]_{lk} = 0 \quad \text{if} \quad |[\Xi]_{lk}| < \tau \tag{3}$$

We find that pruning is essential to find the sparse representation of Ξ and will provide empirical evidence obtained from the numerical experiments. In next Section, we detail how we use the pruning strategy during the training process.

Discussion Thanks to the computational formalism provided by NODEs, the proposed method exhibits several advantages over SINDy; neural SINDy is effective in identifying dynamics with measurements 1) collected at large time steps, 2) collected at irregular time steps, and 3) collected with (additive) noise. The capability of handling samples collected large and/or irregular time steps is naturally incorporated in the continuously-defined dynamics model as shown in Chen et al. (2018). Also, the proposed method is expected to be robust to additive noise due to its loss formulation (Eq. (2)), which is equivalent to finding maximum likelihood estimator (MLE) with Laplacian prior (Gaussian prior if mean squared error is considered). We provide experimental results in Section 6.3.

3.2. Training

We employ mini-batching to train the proposed neural network architecture. For each training step, we randomly sample n_{batch} trajectories from the training set and then randomly sample initial points from the selected n_{batch} trajectories to assemble n_{batch} sub-sequences of length ℓ_{batch}. We solve IVPs with the sample initial points, measure the loss (Eq. (2)), and update the model parameters Θ via Adamax. After the update, we prune the model parameters using the magnitude-based pruning as shown in (3). Algorithm 1 summarizes the training procedure.

4. Neural SINDy for structure-preserving parameterization

4.1. GENERIC structure-preserving parameterization

In the following, we present neural SINDy for a structure-preserving reversible and irreversible dynamics modeling approach. An alternative formalism for irreversible dynamics based on port-Hamiltonians is also presented in the next section.

GENERIC formalism We begin by reviewing the general equation for the nonequilibrium reversible–irreversible coupling (GENERIC) formalism.

In GENERIC, the evolution of an observable $A(\boldsymbol{x})$ is assumed to evolve under the gradient flow

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \{A, E\} + [A, S]$$

where E and S denote generalized energy and entropy, and $\{\cdot,\cdot\}$ and $[\cdot,\cdot]$ denote a Poisson bracket and an irreversible metric bracket. The Poisson bracket is given in terms of a skew-symmetric Poisson matrix L and the irreversible bracket is given in terms of a symmetric positive semi-definite friction matrix M,

$$\{A, B\} = \frac{\partial A}{\partial \boldsymbol{x}} L \frac{\partial B}{\partial \boldsymbol{x}}, \quad \text{and} \quad [A, B] = \frac{\partial A}{\partial \boldsymbol{x}} M \frac{\partial B}{\partial \boldsymbol{x}}.$$

Lastly, the GENERIC formalism requires two degeneracy conditions defined as

$$L\frac{\partial S}{\partial \boldsymbol{x}} = 0, \quad \text{and} \quad M\frac{\partial E}{\partial \boldsymbol{x}} = 0.$$

With the state variables $\boldsymbol{x} = [q, p, S]^{\mathsf{T}}$, the GENERIC formalism defines the evolution of $\boldsymbol{x}$ as

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = L\frac{\partial E}{\partial \boldsymbol{x}} + M\frac{\partial S}{\partial \boldsymbol{x}}. \tag{4}$$

Algorithm 1: Neural SINDy training

```
Initialize $\Theta$
for $(i = 0;\ i < n_{\max};\ i = i + 1)$ do
    Sample $n_{\text{batch}}$ trajectories randomly from $\mathcal{D}_{\text{train}}$
    Sample initial points randomly from the sampled trajectories: $\boldsymbol{x}^r_{s(r)}$, $s(r) \in [0, \ldots, m - \ell_{\text{batch}} - 1]$ for $r = 1, \ldots, n_{\text{batch}}$
    $\tilde{\boldsymbol{x}}(t_1), \ldots, \tilde{\boldsymbol{x}}(t_m)$=ODESolve($\boldsymbol{x}^r_{s(r)}, \boldsymbol{f}_\Theta, t_1, \ldots, t_m$), for $r = 1, \ldots, n_{\text{batch}}$
    Compute the loss $\mathcal{L}$ (Eq. (1))
    Update $\Theta$ via SGD
    Prune $\Theta$ based on the magnitude (Eq. (3))
end
```

Parameterization for GENERIC Here, we review the parameterization technique, developed in Oettinger (2014) and further extended into deep learning settings in Lee et al. (2021). Compared to the settings in Lee et al. (2021), the parameterization technique we present in this paper is one specific instance of possible parameterizations; we assume that L is known and S is an observable state variable. First, we parameterize the total energy such as

$$E_{\theta_1}(q,p,S) = (\boldsymbol{\phi}(q,p,S)^{\mathsf{T}}\Xi)^{\mathsf{T}},$$

with $\theta_1 = \Xi$. Then we parameterize the symmetric irreversible dynamics via the bracket

$$[A,B]_{\theta_2} = \zeta_{\alpha\beta,\mu\nu}\frac{\partial A}{\partial x_\alpha}\frac{\partial E_{\theta_1}}{\partial x_\beta}\frac{\partial B}{\partial x_\mu}\frac{\partial E_{\theta_1}}{\partial x_\nu},$$

where

$$\zeta_{\alpha\beta,\mu\nu} = \Lambda^m_{\alpha\beta}D_{mn}\Lambda^n_{\mu\nu}.$$

The matrices, Λ and D, are skew-symmetric and symmetric positive semi-definite matrices, respectively, such that

$$\Lambda^m_{\alpha\beta} = -\Lambda^m_{\beta\alpha}, \quad \text{and} \quad D_{mn} = D_{nm},$$

where the skew-symmetry and the symmetric positive semi-definiteness can be achieved by the parameterization tricks

$$\Lambda = \frac{1}{2}(\tilde{\Lambda} - \tilde{\Lambda}^{\mathsf{T}}), \quad \text{and} \qquad D = \tilde{D}\tilde{D}^{\mathsf{T}},$$

where $\tilde{\Lambda}$ and $\tilde{D}$ are matrices with learnable entries and, thus, $\theta_2 = [\tilde{\Lambda}, \tilde{D}]$.

With this parameterization, the irreversible part of the dynamics is given by

$$[\boldsymbol{x}, S]_{\theta_2} = M_{\theta_2}\frac{\partial S}{\partial \boldsymbol{x}} = \zeta_{\alpha\beta,\mu\nu}\frac{\partial E_{\theta_1}}{\partial x_\beta}\frac{\partial S}{\partial x_\mu}\frac{\partial E_{\theta_1}}{\partial x_\nu}$$

and, as a result, $\boldsymbol{f}_\Theta$ is now defined as

$$\boldsymbol{f}_\Theta = \{\boldsymbol{x}, E_{\theta_1}\} + [\boldsymbol{x}, S] = L\frac{\partial E_{\theta_1}}{\partial \boldsymbol{x}} + M_{\theta_2}\frac{\partial S}{\partial \boldsymbol{x}},$$

with $\Theta = [\theta_1, \theta_2]$.

Experiments An example problem (with their reference mathematical formulations) considered is a damped nonlinear oscillator from Shang and Öttinger (2020): the ground truth equation can be written in the GENERIC formalism (Eq. (4)) with the following components:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad M = 0.04\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -p \\ 0 & -p & p^2 \end{bmatrix},$$

$$\text{and} \quad E(q,p,S) = \frac{p^2}{2} - 3\cos(q) + S.$$

Alternatively, the equation can be written as

$$\begin{aligned} \dot{q} &= p, \\ \dot{p} &= -3\sin(q) - 0.04p, \\ \dot{S} &= 0.04p^2. \end{aligned}$$

In the experiment, we assume that we have knowledge on L and that the measurements on S are available. That is, the nSINDy method with the GENERIC structure preservation seeks the unknown M and E. In the following, we denote this method by nSINDY - GNN (GENERIC Neural Network). For generating data, we base our implementation on the code from Shang and Öttinger (2020). We generate 800 training trajectories, 160 validation trajectories, and 160 test trajectories with $\Delta t = 0.001$ and the simulation time $[0, 5.12]$; each trajectory starts with varying initial conditions. We use a dictionary consisting of polynomials and trigonometric functions:

$$\mathcal{P} = \{\mathcal{P}_3(p, q, S), \cos(q), \sin(q), \cos(p), \sin(p), \cos(S), \sin(S)\},$$

where $\mathcal{P}_3(q,p,S) = \{1, q, q^2, q^3, q^2p, q^2S, qp, qp^2, qpS, qS, qS^2, p, p^2, p^3, p^2S, pS, pS^2, S, S^2, S^3\}$.

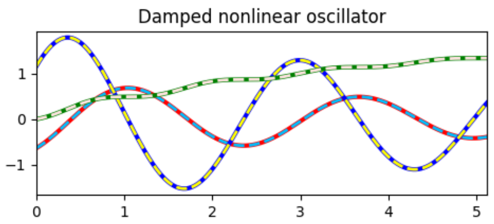

Figure 1: [GENERIC structure preservation] Examples of reference trajectories (solid lines) and computed trajectories (dashed lines) from learned dynamics.

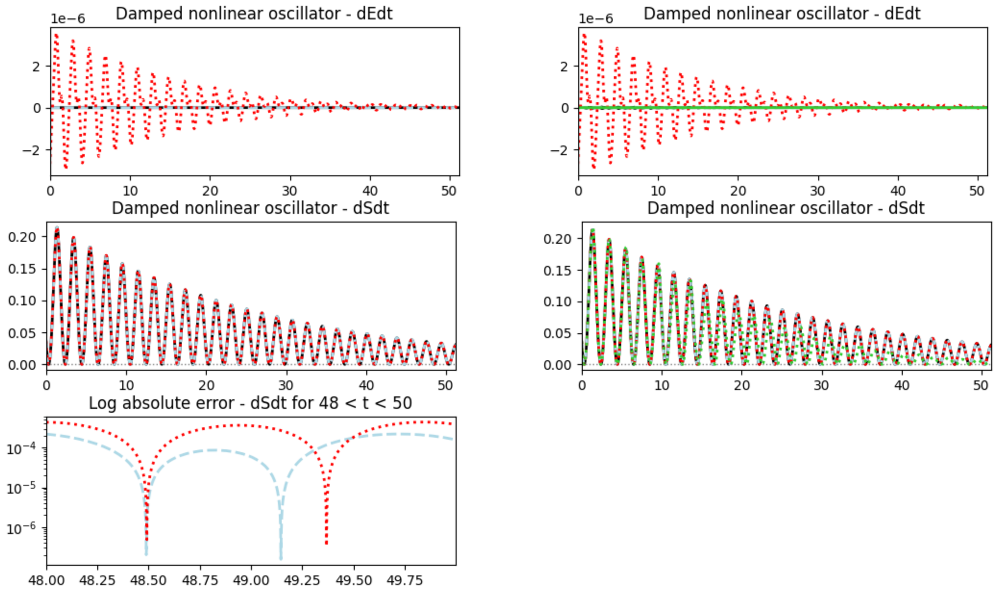

Figure 2: [GENERIC structure preservation] $\frac{\mathrm{d}E}{\mathrm{d}t}$ and $\frac{\mathrm{d}S}{\mathrm{d}t}$ of the reference system (black solid line). (Left) the system identified by nSINDy (red dotted line), and the system identified by nSINDy - GNN (lightblue dashed line). (Right) The systems identified by nSINDy, nSINDy-GNN and, nSINDy - NN (green).

We implement the proposed method in PYTHON 3.7.2 and PYTORCH 1.9.0 Paszke et al. (2019). In all training, we use Adamax Kingma and Ba (2015) with an initial learning rate 0.01 and use

Parameterization	Ground truth	Identified
nSINDy	$\dot{q} = p$ $\dot{p} = -3\sin(q) - 0.04p$ $\dot{S} = 0.04p^2$	$\dot{q} = 0.999622p$ $\dot{p} = -2.8645\sin(q) - 0.04001p - 0.1340q + 0.0190q^3$ $\dot{S} = 0.040081p^2$
nSINDy - GNN	$E = 0.5p^2 - 3\cos(q) + S$	$E_{\theta_1} = 0.500144p^2 - 2.999785\cos(q) + 0.998337S$

Table 1: [GENERIC structure preservation] The equation names, the ground truth equations, and the identified equations by using nSINDy (non structure preserving parameterization) and nSINDy - GNN (GENERIC structure preserving parameterization).

exponential learning rate decay with the multiplicative factor, 0.9987. In the following experiments, we set the penalty weight as $\lambda = 10^{-4}$ and the pruning threshold as $\tau = 10^{-6}$. For ODESolve, we use the Dormand–Prince method (dopri5) Dormand and Prince (1980) with relative tolerance 10^{-7} and absolute tolerance 10^{-9} unless otherwise specified. Finally, $n_{\max} = 300$. All experiments are performed on MACBOOK PRO with M1 CPU and 16 GB memory.

Table 1 reports the coefficients of the identified systems when nSINDy and nSINDy - GNN are used. With $n_{\max} = 300$, nSINDy fails to correctly identify the system, whereas nSINDy - GNN identifies the exact terms correctly and computes the coefficients accurately. Figure 1 depicts the ground truth trajectory and the trajectory computed from the learned dynamics function (nSINDy - GNN). Figure 2 (left column) shows the advantages of using GENERIC structure preservation via plots of $\frac{\mathrm{d}E}{\mathrm{d}t}$ and $\frac{\mathrm{d}S}{\mathrm{d}t}$. The difference between structure-preserving and non-structure-preserving parameterization is shown dramatically in the plot of $\frac{\mathrm{d}E}{\mathrm{d}t}$; the structure-preserving one produces $\frac{\mathrm{d}E}{\mathrm{d}t} = 0$, whereas the non-structure-preserving one produces fluctuating values of $\frac{\mathrm{d}E}{\mathrm{d}t}$. We also test the black-box neural network parameterization of E with an MLP: a fully-connected neural networks with 4 layers and 25 neurons in each layer, which we denote by nSINDy-NN. Figure 2 (right column) shows that the structure-preservation is being enforced by design (i.e., $\frac{\mathrm{d}E}{\mathrm{d}t} = 0$ and $\frac{\mathrm{d}S}{\mathrm{d}t} \geq 0$). However, nSINDy - NN is shown to be less effective in extrapolation compared to dictionary-based parameterization; training simulation time is 5.12 seconds while the test simulation time is 51.20 seconds. This observation is consistent with the results that will be presented in the later section, where we compare dictionary-based parameterization and black-box parameterization. Dictionary-based parameterization generalizes better as it is equivalent to imposing stronger inductive biases (i.e., dictionaries) to the model. We also report the relative training time of three different approaches, nSINDy, nSINDy-GNN, and nSINDy-NN; training nSINDy-GNN and nSINDy-NN require $\times 15$ and $\times 80$ more seconds than training nSINDy. As have seen in the previous work (e.g., Greydanus et al. (2019)), this overhead mostly comes from the automatic differentiation.

4.2. Hamiltonian structure-preserving parameterization

As a special case of the GENERIC formalism, we consider the Hamiltonian structure-preserving parameterization technique, which is originally proposed in Hamiltonian neural networks (HNNs) Greydanus et al. (2019): parameterizing the Hamiltonian function $\mathcal{H}(q,p)$ as $\mathcal{H}_\Theta(q,p)$ such that

$$\mathcal{H}_\Theta = (\boldsymbol{\phi}(q,p)^{\mathsf{T}}\Xi)^{\mathsf{T}}. \tag{5}$$

With the above parameterization and $\boldsymbol{x} = [q,p]^{\mathsf{T}}$, the dynamics can be modeled as

$$\begin{bmatrix} \frac{\mathrm{d}q}{\mathrm{d}t} \\ \frac{\mathrm{d}p}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{H}_\Theta}{\partial p} \\ -\frac{\partial \mathcal{H}_\Theta}{\partial q} \end{bmatrix},$$

i.e., $\boldsymbol{f}_\Theta$ is now defined as $\boldsymbol{f}_\Theta = [-\frac{\partial \mathcal{H}_\Theta}{\partial p}, \frac{\partial \mathcal{H}_\Theta}{\partial q}]^\mathsf{T}$. Then the loss objective can be written as

$$\left\| \frac{\partial \mathcal{H}_\Theta}{\partial p} - \frac{\mathrm{d}q}{\mathrm{d}t} \right\|_1 + \left\| \frac{\partial \mathcal{H}_\Theta}{\partial q} + \frac{\mathrm{d}p}{\mathrm{d}t} \right\|_1 .$$

As noted in Lee et al. (2021), with canonical coordinates $\boldsymbol{x}{=}[q,p]^\mathsf{T}$, and canonical Poisson matrix $L{=}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$, and $M = \mathbf{0}$, the GENERIC formalism in Eq. (4) recovers Hamiltonian dynamics.

Experiments We examine the performance of the proposed method with a list of example dynamic systems (see Table 2).

For generating data, we base our implementation on the code from Greydanus et al. (2019). We generate 800 training trajectories, 160 validation trajectories, and 160 test trajectories for varying initial conditions. For the mass spring and the pendulum problem, we use $\Delta t = 0.1$ and $\Delta t = 1/15$ and the simulation time is set as $[0, 3]$ and $[0, 9]$, respectively.

Eq. name	Parameterization	Ground truth	Identified
Ideal mass-spring	nSINDy	$\dot{q} = p$ $\dot{p} = -q$	$\dot{q} = 0.999989p$ $\dot{p} = -1.000030q$
	nSINDy - HNN	$\mathcal{H}(q,p) = 0.5q^2 + 0.5p^2$	$\mathcal{H}(q,p) = 0.499991q^2 + 0.500033p^2$
Ideal pendulum	nSINDy	$\dot{q} = p$ $\dot{p} = -6\sin(q)$	$\dot{q} = 1.000018p + 0.034095q - 0.004796q^3$ $\dot{p} = -5.890695\sin(q) - 0.107052q$
	nSINDy - HNN	$\mathcal{H}(q,p) = 6 - 6\cos(q) + 0.5p^2$	$\mathcal{H}(q,p) = 6 - 5.999838\cos(q) + 0.500014p^2$

Table 2: [Hamiltonian structure preservation] The equation names, the ground truth equations, and the identified equations by using nSINDy (non structure preserving parameterization) and nSINDy - HNN (Hamiltonian structure preserving parameterization).

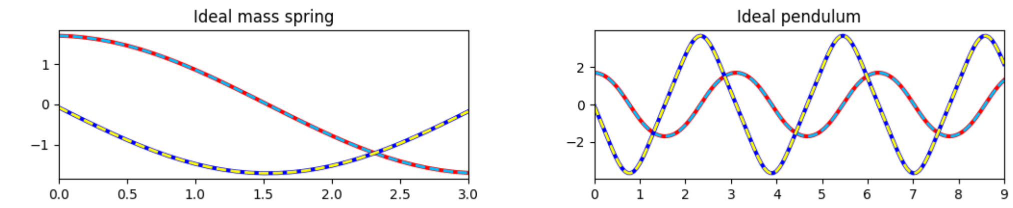

Figure 3: [Hamiltonian structure preservation]Examples of reference trajectories (solid lines) and computed trajectories (dashed lines) from learned dynamics.

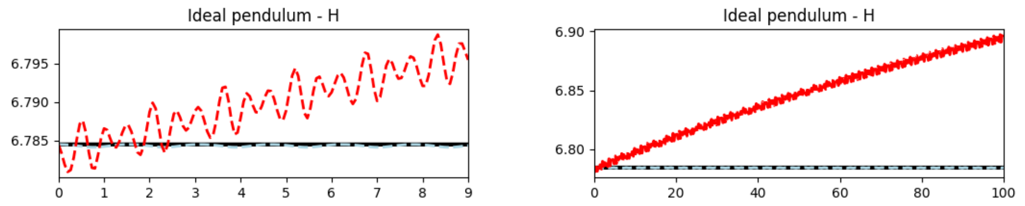

Figure 4: [Hamiltonian structure preservation] The Hamiltonian function measured by the trajectories computed with nSINDy (red dashed line) and nSINDy-HNN (blue dashed line). Figure below depicts the Hamiltonian functions measured by the trajectories computed for extended numerical simulation time, i.e., 100 seconds (the training simulation time is 9 seconds).

Table 2 shows the ground truth and identified equations by using neural SINDy. For the mass spring problem, we use $\mathcal{P}_3(q,p)$ and for the pendulum problem we use a dictionary consisting of polynomials and trigonometric functions:

$$\mathcal{P} = \{\mathcal{P}_3, \cos(q), \sin(q), \cos(p), \sin(p)\}.$$

Again, we use the same experimental settings described above for the GENERIC parameterization. In Table 2, we presented results obtained by two different parameterization techniques: 1) the "plain" dictionary approach (Eq. (1)) denoted by (nSINDy) and 2) the Hamiltonian approach (Eq (5)) denoted by (nSINDy - HNN). Figure 3 depicts examples of reference trajectories and trajectories computed from identified dynamics, where the trajectories are chosen from the test set. Figure 4 depicts how the energy is being conserved in the dynamics learned with nSINDy-HNN, whereas the plain nSINDy fails to conserve the energy.

5. Port-Hamiltonian structure preservation

Here, we consider the dictionary-based parameterization for the port-Hamiltonian systems. We follow the port-Hamilonian neural network formulation presented in Desai et al. (2021), which is written as:

$$\begin{bmatrix} \frac{\mathrm{d}q}{\mathrm{d}t} \\ \frac{\mathrm{d}p}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathcal{H}_\Theta}{\partial p} \\ \frac{\partial \mathcal{H}_\Theta}{\partial q} \end{bmatrix} + \begin{bmatrix} 0 \\ N\frac{\partial \mathcal{H}_\Theta}{\partial p} \end{bmatrix} + \begin{bmatrix} 0 \\ F(t) \end{bmatrix},$$

where $\mathcal{H}_\Theta$ denotes the parameterized Hamiltonian function, N denotes the nonzero coefficient for a damping term, and $F(t)$ denotes a time-dependent forcing. We consider the dictionary-type parameterization:

$$\mathcal{H}_\Theta = (\boldsymbol{\phi}(q,p)^{\mathsf{T}}\Xi)^{\mathsf{T}},$$

set N to be a trainable coefficient, and assume that $F(t)$ is known.

As in Desai et al. (2021), we consider the Duffing equation, where the Hamiltonian function is defined as

$$\mathcal{H}(q,p) = \frac{p^2}{2m} + \alpha\frac{q^2}{2} + \beta\frac{q^4}{4}.$$

Then the Duffing equation considering the damping term and the forcing can be written as:

$$\ddot{q} = -\alpha q - \beta q^3 + \gamma \sin(\omega t) - \delta \dot{q}.$$

In the following experiment, we consider

$$\mathcal{P}(4) = \{1, q, p, q^2, qp, p^2, q^3, q^2pqp^2, p^3, q^4, q^3p, q^2p^2, qp^3, p^4\}$$

for parameterizing the Hamiltonian function.

Nonchaotic case: $\alpha = -1$, $\beta = 1$, $\gamma = 0.3$, $\delta = 0.3$, and $\omega = 1.2$. In the experiment, we assume that we know the values of γ and ω and identify the Hamiltonian function, which can be written as

$$\mathcal{H}(q,p) = 0.5p^2 - 0.5q^2 + 0.25q^4.$$

The identified Hamiltonian function is

$$\mathcal{H}_\Theta(q,p) = 0.50015p^2 - 0.50004q^2 + 0.25010q^4$$

and the identified value of $\delta = 0.3002$. Figure 5 depicts the reference trajectories and the trajectories computed from the learned dynamics model (left) and the reference Hamiltonian function and the learned Hamiltonian function (right).

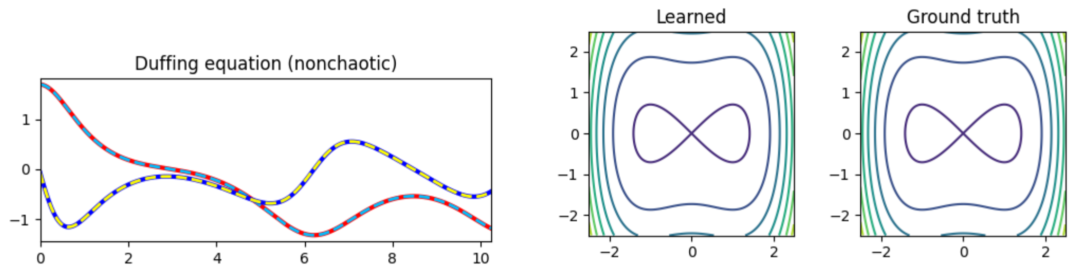

Figure 5: [Duffing equation] Left: Examples of reference trajectories (solid lines) and computed trajectories (dashed lines) from learned dynamics. Right: the learned and the ground truth Hamiltonian function evaluated on the uniform grid of (q, p).

Chaotic case $\alpha = -1$, $\beta = 1$, $\gamma = 0.1$, $\delta = 0.39$, and $\omega = 1.4$. The ground truth and identified Hamiltonian functions are

$$\mathcal{H}(q,p) = 0.5p^2 - 0.5q^2 + 0.25q^4,$$
$$\mathcal{H}_\Theta(q,p) = 0.50007p^2 - 0.50053q^2 + 0.25017q^4$$

and the identified value of $\delta = 0.1001$. As observed in the nonchaotic case, the identified system matches well with the reference system.

6. Additional neural SINDy experiments

Finally, we report the results of applying neural SINDy to various dynamical systems without structure-preservation. We consider the same experimental settings that are used in the above experiments: Adamax with the initial learning rate 0.01, exponential learning rate decay with a factor 0.9987, L1-penalty weight 10^{-4}, pruning threshold[1] 10^{-6}, dopri5 for the ODE integrator with 10^{-7} and 10^{-9} for relative and absolute tolerances. We employ a set of polynomials, $\mathcal{P}_{n,d}$, where n is the number of variables and d is the maximal total degree of polynomials in the set.

6.1. Experiment set

We examine the performance of the proposed method for a number of dynamical systems (ones in Table 3 and the Lorenz 96 model). To generate data, we base our implementation on the code from Liu et al. (2020): generating 1600 training trajectories, 320 validating trajectories, and 320 test trajectories, for varying initial conditions. For Lorenz 96, we generate 128 training/validating trajectories and 16 testing trajectories. For the first four example problems, we use time step size $\Delta t = 0.01$ and total simulation time $t \in [0, 51.20]$. For Lorenz 63 and 96, we use $\Delta t = 0.0005$ and 0.01, and $t \in [0, 2.56]$ and $[0, 30.72]$. For training, we use $n_{\max} = 500$, $n_{\max} = 2000$, and $n_{\max} = 250$ for the first four example problems, Lorenz 63, and Lorenz 96, respectively.

1. Based on the result of a parameter sweep for pruning threshold, $[10^{-6}, 10^{-2}]$, we recommend threshold $\leq 10^{-5}$.

Eq. name	Ground truth	Identified
Hyperbolic	$\dot{x} = -0.05x$ $\dot{y} = x^2 - y$	$\dot{x} = -0.050006x$ $\dot{y} = 1.000063x^2 - 1.000063y$
Cubic oscillator	$\dot{x} = -0.1x^3 + 2y^3$ $\dot{y} = -2x^3 - 0.1y^3$	$\dot{x} = -0.100075x^3 + 2.000019y^3$ $\dot{y} = -1.999906x^3 - 0.099915y^3$
Van der Pol	$\dot{x} = y$ $\dot{y} = -x + 2y - 2x^2y$	$\dot{x} = 1.000015y$ $\dot{y} = -1.000043x + 2.000271y - 1.999807x^2y$
Hopf bifurcation	$\dot{\mu} = 0$ $\dot{x} = \mu x + y - x^3 - xy^2$ $\dot{y} = \mu y - x - yx^2 - y^3$	$\dot{\mu} = 0$ $\dot{x} = 0.999237\mu x + 1.000174y - 0.999523x^3 - 0.999582xy^2$ $\dot{y} = 0.999515\mu y - 0.999821x - 0.999276yx^2 - 0.999409y^3$
Lorenz 63	$\dot{x} = -10x + 10y$ $\dot{y} = 28x - xz - y$ $\dot{z} = xy - \frac{8}{3}z$	$\dot{x} = -10.000108x + 9.999878y$ $\dot{y} = 27.999895x - 0.999998xz - 0.999883y$ $\dot{z} = 1.000017xy - 2.666697z$

Table 3: [nSINDy experiments] The equation names, the ground truth equations, and the identified equations by using nSINDy.

Table 3 shows the ground truth equations and equations identified by using neural SINDy (except Lorenz 96). We use $\mathcal{P}_{2,3}$ aad $\mathcal{P}_{3,3}$ for the first three example problems and the fourth example problem, and $\mathcal{P}_{3,2}$ and $\mathcal{P}_{6,2}$ for Lorenz 63 and Lorenz 96. During training, coefficients with small magnitude are pruned (i.e., setting them as 0) and are not presented in Table. We also report the learned coefficients without the pruning strategy in Appendix, which highlights that the pruning is required to zero out the coefficients of small magnitude.

Figure 6 depicts examples of reference trajectories and trajectories computed from identified dynamics, where the trajectories are chosen from the test set. The longer simulation indicates the extrapolation results. For Lorenz 63, the dynamics model is trained with the trajectories with the simulation time [0,2.56] and is tested on [0,15.36]. The figure shows that the learned model produce a trajectory that matches well up to $t = 12$.

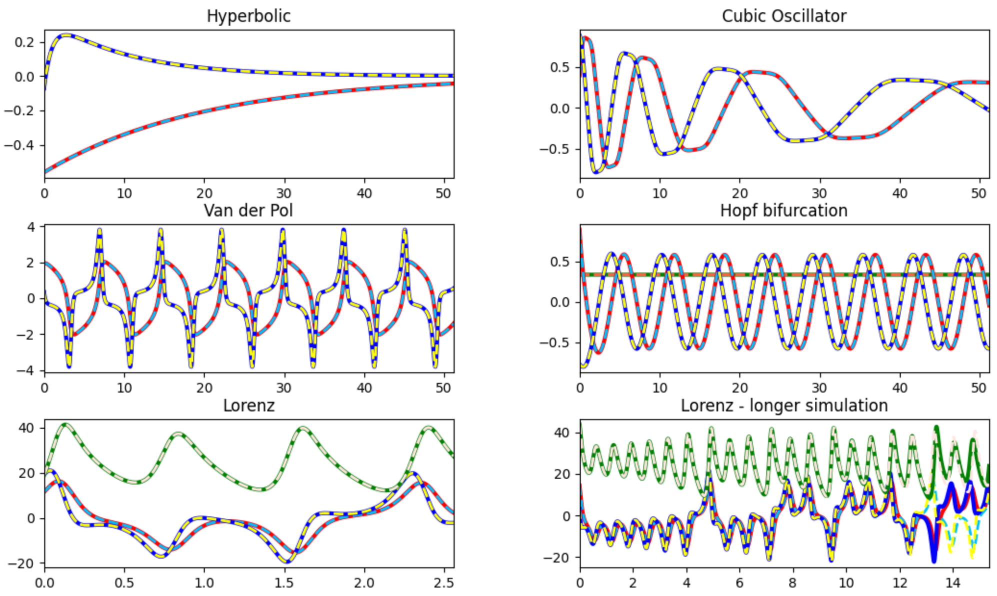

Figure 6: [nSINDy experiments] Examples of reference trajectories (solid lines) and computed trajectories (dashed lines) from learned dynamics. Longer simulation reports the extrapolation results (the training period [0,2.56] and the testing period [0, 15.36]).

Next we report the results of experiments with Lorenz 96. The reference Lorenz 96 equation is given by

$$\dot{x}_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F,$$

where $i = 1, \dots, 6$, $F = 8x_{-1} = x_5$, $x_0 = x_6$, and $x_1 = x7$. The initial conditions are sampled by adding perturbation noise (noise sampled from uniform distribution [-1,1]) to the base initial condition $[1, 8, 8, 8, 8, 8]$.

The dynamical system identified by neural SINDy is

$$\begin{aligned}
x_1 &= 0.9995x_2x_6 - 0.9997x_5x_6 - 1.0000x_1 + 8.002,\\
\dot{x}_2 &= 1.0001x_3x_1 - 1.0002x_6x_1 - 0.9999x_2 + 8.0014,\\
\dot{x}_3 &= 1.0003x_4x_2 - 0.9999x_1x_2 - 1.0008x_3 + 8.0005,\\
\dot{x}_4 &= 1.0005x_5x_3 - 1.0000x_2x_3 - 1.0000x_4 + 8.0000,\\
\dot{x}_5 &= 0.9998x_6x_4 - 1.0004x_3x_4 - 1.0011x_5 + 7.9992,\\
\dot{x}_6 &= 0.9992x_1x_5 - 0.9998x_4x_5 - 0.9992x_6 + 7.9999.
\end{aligned}$$

6.2. Comparison with MLP parameterization

Here, we also report that time-instantaneous MSEs computed by nSINDy are several orders of magnitude smaller ($10^6 \sim 10^8$ smaller) than those computed by the black-box NODEs.

Figure 7 depicts time-instantaneous mean-squared errors measured between the ground truth trajectories and the trajectories computed from the learned dynamics functions, $\boldsymbol{f}_\Theta$. We consider the two models: the black-box NODE models and the proposed SINDy models. For the SINDy models, we use the same experimental settings considered in Experimental results Section. For the black-box NODE models, we consider feed-forward neural networks consisting of four layers with 100 neurons in each layer with the hyperbolic Tangent nonlinear activation (Tanh). For training the both models, we use the same optimizer, Adamax, and use the same initial learning rate 0.01 with the same decay strategy (i.e., exponential decay with the multiplicative factor 0.9987). Both models are trained over 500 epochs and 2000 epochs, respectively, for the first four benchmark problems and the last benchmark problem, respectively.

6.3. Comparisons with SINDy

Next, we demonstrate the results of numerical experiments, where we compare the performance of the proposed neural SINDy and the (plain) SINDy algorithm Brunton et al. (2016). We consider specific experimental settings, where the proposed neural SINDy is expected perform better than SINDy: 1) experiments with data collected with larger time step, Δt, 2) experiments with data collected at irregular time interval, and 3) experiments with noisy measurements.

For the numerical experiments of SINDy, we use the PYSINDY package Kaptanoglu et al. (2022). For neural SINDy, we consider the same experimental setting used in the previous experiments: $\mathcal{P}_{2,3}$. As a benchmark problem, we consider the cubic oscillator and take 80 training trajectories, which are originally generated by using the step size 0.01 and we choose $n_{\max} = 200$.

For the first set of experiments, we sample from the generated trajectories at a coarsened uniform time interval: $\Delta t = \{0.1, 0.3, 0.5\}$. Table 4 reports the identified systems with the large time steps for data collection and neural SINDy accurately identifies the correct dictionaries and the coefficients whereas SINDy gradually performs worse as the step size becomes larger.

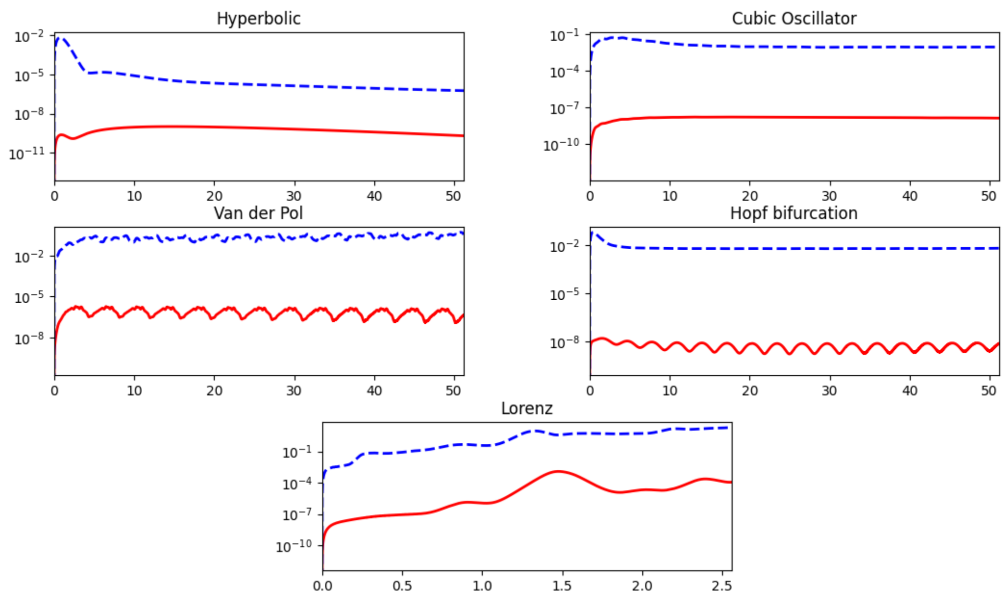

Figure 7: [nSINDy experiments] Time-instantaneous mean-squared errors. Two model are compared: the black-box NODE models (blue dashed lines) and the proposed neural SINDy model (red solid lines).

	SINDy	nSINDy
$\Delta t = 0.1$	$\dot{x} = -0.098x^3 + 1.995y^3$ $\dot{y} = -1.995x^3 - 0.099y^3$	$\dot{x} = -0.1005x^3 + 2.0001y^3$ $\dot{y} = -1.9996x^3 - 0.0995y^3$
$\Delta t = 0.3$	$\dot{x} = 0.012y - 0.084x^3 + 0.031x^2y - 0.048xy^2 + 1.920y^3$ $\dot{y} = -1.937x^3 - 0.046x^2y - 0.094xy^2 - 0.085y^3$	$\dot{x} = -0.1001x^3 + 1.9997y^3$ $\dot{y} = -2.0003x^3 - 0.0999y^3$
$\Delta t = 0.5$	$\dot{x} = 0.014x + 0.023y - 0.015y^2 - 0.106x^3 + 0.087x^2y - 0.120xy^2 + 1.824y^3$ $\dot{y} = 0.020y - 0.017xy - 1.853x^3 - 0.161x^2y - 0.241xy^2 - 0.118y^3$	$\dot{x} = -0.0999x^3 + 1.9995y^3$ $\dot{y} = -2.0005x^3 - 0.1001y^3$

Table 4: [SINDy and nSINDy experiments] Cubic oscillator, collected with larger time steps, Δt. The identified equations by using SINDy and nSINDy.

For the second set of experiments, we sample from the original training trajectories of length 5120 (step size: $\Delta t = 0.01$, 51.20 second) at irregular time indices. We use three randomly generated time indices with lengths that are proportional to the original length: $r = \{0.9, 0.1, 0.02\}$, which denote 90%, 10% and 2% of the original trajectories. Table 5 reports the experimental results. Again, the proposed neural SINDy outperforms SINDy in terms of accuracy.

Lastly, we assess the performance of the neural SINDy with noisy measurements. For this, we add an artificial noise with varying scaling factors, $\epsilon_{\text{noise}} = \{.05, .10, .20, .30\}$, i.e., 5%, 10%, 20%, 30% additive noise, to noise-free simulation data. That is, for each trajectory, the standard deviation of each variable over the entire simulation time is computed; we sample independent and identically distributed samples from the unit Normal distribution, which are then multiplied by scaling factors and added to the original trajectory. Note that for both SINDy and neural SINDy we assume that a correct initial condition is known.

Table 6 reports the equations identified by SINDy and neural SINDy. In high signal-to-noise-ratio (SNR) scenarios (ϵ_{noise} =5%∼10%), both SINDy and neural SINDy perform comparably although the coefficients identified by neural SINDy are slightly more accurate. The benefits of using neural SINDy are more pronounced in low SNR scenarios (ϵ_{noise} =20%∼30%); neural SINDy

	SINDy	nSINDy
$r = 0.9$	$\dot{x} = -0.112x^3 + 2.230y^3$ $\dot{y} = -2.223x^3 - 0.111y^3$	$\dot{x} = -0.0981x^3 + 1.9986y^3$ $\dot{y} = -2.0012x^3 - 0.1013y^3$
$r = 0.1$	$\dot{x} = -0.0121 + 0.031x + 0.290y - 0.010x^2 + 0.057xy + 0.184y^2 - 0.333x^3 - 0.157x^2y - 0.222xy^2 + 5.137y^3$ $\dot{y} = -0.0111 - 0.287x - 0.068y + 0.086x^2 + 0.039xy + 0.023y^2 - 5.110x^3 + 0.193x^2y + 0.187xy^2 - 0.129y^3$	$\dot{x} = -0.1006x^3 + 1.9993y^3$ $\dot{y} = -1.9981x^3 - 0.1004y^3$
$r = 0.02$	$\dot{x} = -0.0561 + 0.097x + 3.509y + 0.185x^2 + 1.060xy + 0.638y^2 - 0.794x^3 - 1.356x^2y - 1.946xy^2 + 12.659y^3$ $\dot{y} = 0.0831 - 3.526x - 0.282y - 1.026x^2 - 0.180xy - 0.337y^2 - 12.071x^3 - 0.467x^2y + 0.965xy^2 - 0.056y^3$	$\dot{x} = -0.0992x^3 + 1.9988y^3$ $\dot{y} = -2.0006x^3 - 0.0996y^3$

Table 5: [SINDy and nSINDy experiments] Cubic oscillator, collected at irregular time interval. The identified equations by using SINDy and nSINDy.

	SINDy	nSINDy
$\epsilon_{\text{noise}} = 0.05$ (5%)	$\dot{x} = -0.104x^3 + 0.015x^2y + 1.977y^3$ $\dot{y} = -1.979x^3 + 0.012xy^2 - 0.102y^3$	$\dot{x} = -0.1005x^3 + 2.0022y^3$ $\dot{y} = -1.9994x^3 + 0.0025x^2y - 0.1005y^3$
$\epsilon_{\text{noise}} = 0.10$ (10%)	$\dot{x} = -0.101x^3 + 0.045x^2y + -0.014xy^2 + 1.917y^3$ $\dot{y} = -0.014x + -1.887x^3 + -0.011x^2y + 0.023xy^2 + -0.101y^3$	$\dot{x} = -0.1024x^3 - 0.0024xy + 0.0048xy^2 + 2.0017y^3$ $\dot{y} = 0.0020x^2 - 1.9994x^3 - 0.0029xy + 0.0016y^2 - 0.0989y^3$
$\epsilon_{\text{noise}} = 0.20$ (20%)	$\dot{x} = 0.021y - 0.014y^2 + -0.095x^3 + 0.108x^2y - 0.025xy^2 + 1.663y^3$ $\dot{y} = -0.037x + 0.020y - 1.624x^3 + -0.068x^2y - 0.013xy^2 - 0.140y^3$	$\dot{x} = -0.1066x^3 - 0.0054xy + 0.0112xy^2 + 2.0070y^3$ $\dot{y} = -1.9993x^3 + 0.0087x^2y + 0.0024y^2 - 0.1006y^3$
$\epsilon_{\text{noise}} = 0.30$ (30%)	$\dot{x} = 0.042y + 0.026xy + -0.019y^2 - 0.075x^3 + 0.193x^2y - 0.043xy^2 + 1.353y^3$ $\dot{y} = -0.066x + 0.040y - 0.014xy - 1.296x^3 - 0.119x^2y - 0.068xy^2 - 0.180y^3$	$\dot{x} = -0.1084x^3 - 0.0075xy + 0.0139xy^2 + 2.0064y^3$ $\dot{y} = -0.0050x^2 - 1.9984x^3 + 0.0060x^2y - 0.0072xy + 0.0049y^2 - 0.0984y^3$

Table 6: [SINDy and nSINDy experiments] Cubic oscillator, noisy measurements. The identified equations by using SINDy and nSINDy.

has less errors in finding the correct dictionaries and the identified coefficients are more accurate than SINDy. As the loss term of neural SINDy minimizes the L1 error 2, minimizing the loss is equivalent to finding a maximum likelihood estimator (MLE) with zero mean Laplacian prior and, thus, neural SINDy is expected to be robust to Laplacian additive noise. Although not reported, we have confirmed this with experimental data: we have observed more accurate identification results than the ones in Table 6 when the zero-mean Laplacian noise is added. With the same L1 loss formulation, we observe that the neural SINDy is still robust to a different type of additive noise (i.e., Gaussian) to some extent as shown in Table 6.

7. Discussion

The proposed method shares the same limitations of the original SINDy method: successful identification requires inclusion of the correct dictionaries in the library. Potential alternatives are well-studied in the literature and include either adding an extra "black-box" neural network to compensate missing dictionaries or designing a neural network that can learn dictionaries from data (such as in Sahoo et al. (2018)).

The gradient-based parameter update and the magnitude-based pruning could potentially zero out unwanted coefficients in a special case: when the signs of the coefficients need to be changed in the later stage of the training. If the magnitude of the loss term becomes very small (and the gradient as well), the updated coefficients may satisfy the pruning condition shown in (3). We believe that this issue can be mitigated by adopting ideas of greedy algorithms (e.g., Boninsegna et al. (2018); Billings (2013)) to the proposed neural SINDy method.

Lastly, when adaptive step-size ODE solvers are used (e.g., dopri5), numerical underflow may occur. This can be mitigated by trying different initialization for the coefficients, or smaller batch length ℓ_{batch}. However, further study regarding robust initialization is required.

8. Related work

System identification In Schmidt and Lipson (2009), the proposed method uses a genetic algorithms to identify governing physical laws (Lagrangian or Hamiltonian) from measurements of real experiments. A seminal work on sparse regression methods for system identification has been proposed in Brunton et al. (2016). Then the sparse regression methods have been extended to various settings, e.g., for model predictive control Kaiser et al. (2018), and for identifying dynamics in latent space using autoencoders Hinton and Salakhutdinov (2006) and then learning parsimonious representations for the latent dynamics Champion et al. (2019). In Yang et al. (2020); Bhouri and Perdikaris (2022), Bayesian frameworks for identifying dynamical systems have been proposed. A dictionary-based NODE parameterization has considered and the parameters are inferred via maximum a posteriori estimation, which results in the probability distributions of each coefficients.Also, the sparse regression methods have been applied for identifying partial differential equations (PDEs) Rudy et al. (2017, 2019). For identifying PDEs, approaches such as physics-informed neural networks Raissi et al. (2019) and PDE-net Long et al. (2018) have been studied recently.

Structure preserving neural network Designing neural network architectures that exactly enforces important physical properties has been an important topic and studied extensively. Parameterization techniques that preserve physical structure include Hamiltonian neural networks Greydanus et al. (2019); Toth et al. (2019), Lagrangian neural networks Cranmer et al. (2020); Lutter et al. (2018), port-Hamiltonian neural networks Desai et al. (2021), and GENERIC neural networks Hernández et al. (2021); Lee et al. (2021). Neural network architectures that mimic the action of symplectic integrators have been proposed in Chen et al. (2019); Jin et al. (2020); Tong et al. (2021), and a training algorithm exploiting physical invariance for learning dynamical systems, e.g., time-reversal symmetric, has been studied in Huh et al. (2020).

9. Conclusion

We have proposed a simple and effective deep-learning-based training algorithm for a dictionary-based parameterization of nonlinear dynamics. The proposed algorithm is based on the training procedure introduced in neural ordinary differential equations (NODE) and employs L1-weight decay of model parameters and magnitude-based pruning strategy, inspired by sparse identification of nonlinear dynamics (SINDy). We have further extended the dictionary-based parameterization approach to structure-preserving parameterization techniques, such as Hamiltonian neural networks, GENERIC neural networks, and port-Hamiltonian networks. For a suite of benchmark problems, we have demonstrated that the proposed training algorithm is very effective in identifying the underlying dynamics from data with expected gains from imposing structure-preservation.

Acknowledgments

N. Trask and P. Stinis acknowledge funding under the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems (PhILMs) project funded by DOE Office of Science (Grant number DE-SC001924). N. Trask acknowledges funding from the DOE Early Career program. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear

Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

Kosmas Alexopoulos, Nikolaos Nikolakis, and George Chryssolouris. Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing. *International Journal of Computer Integrated Manufacturing*, 33(5):429–439, 2020.

Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

Tom Bertalan, Felix Dietrich, Igor Mezić, and Ioannis Kevrekidis. On learning hamiltonian systems from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12):121107, 2019.

Mohamed Aziz Bhouri and Paris Perdikaris. Gaussian processes meet neuralodes: a bayesian framework for learning the dynamics of partially observed systems from scarce and noisy data. *Philosophical Transactions of the Royal Society A*, 380(2229):20210201, 2022.

Stephen A Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of chemical physics*, 148(24):241723, 2018.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

Kathleen Champion, Bethany Lusch, Nathan Kutz, and Steven Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45): 22445–22451, 2019.

Renyi Chen and Molei Tao. Data-driven prediction of general hamiltonian dynamics via learning exactly-symplectic maps. *arXiv preprint arXiv:2103.05632*, 2021.

Ricky Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6572–6583, 2018.

Wenqian Chen, Qian Wang, Jan Hesthaven, and Chuhua Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of Computational Physics*, 2021.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2019.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop*, 2020.

Shaan Desai, Marios Mattheakis, David Sondak, Pavlos Protopapas, and Stephen Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *arXiv preprint arXiv:2107.08024*, 2021.

John R Dormand and Peter J Prince. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Chady Ghnatios, Iciar Alfaro, David González, Francisco Chinesta, and Elias Cueto. Data-driven generic modeling of poroviscoelastic materials. *Entropy*, 21(12):1165, 2019.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32:15379–15389, 2019.

Ernst Hairer, Marlis Hochbruck, Arieh Iserles, and Christian Lubich. Geometric numerical integration. *Oberwolfach Reports*, 3(1):805–882, 2006.

Quercus Hernández, Alberto Badías, David González, Francisco Chinesta, and Elías Cueto. Structure-preserving neural networks. *Journal of Computational Physics*, 426:109950, 2021.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

In Huh, Eunho Yang, Sung Ju Hwang, and Jinwoo Shin. Time-reversal symmetric ode network. *Advances in Neural Information Processing Systems*, 33, 2020.

Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. Sympnets: Intrinsic structure-preserving symplectic networks for identifying hamiltonian systems. *Neural Networks*, 132:166–179, 2020.

Eurika Kaiser, Nathan Kutz, and Steven Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219): 20180335, 2018.

Alan A. Kaptanoglu, Brian M. de Silva, Urban Fasel, Kadierdan Kaheman, Andy J. Goldschmidt, Jared Callaham, Charles B. Delahunt, Zachary G. Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton. Pysindy: A comprehensive python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, 2022. doi: 10.21105/joss.03994. URL https://doi.org/10.21105/joss.03994.

K Karapiperis, L Stainier, M Ortiz, and JE Andrade. Data-driven multiscale modeling in mechanics. *Journal of the Mechanics and Physics of Solids*, 147:104239, 2021.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kookjin Lee and Kevin T Carlberg. Deep conservation: A latent-dynamics model for exact satisfaction of physical conservation laws. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 277–285, 2021.

Kookjin Lee, Nathaniel Trask, and Panos Stinis. Machine learning structure preserving brackets for forecasting irreversible processes. In *Advances in Neural Information Processing Systems*, 2021.

Yuying Liu, J Nathan Kutz, and Steven L Brunton. Hierarchical deep learning of multiscale differential equation time-steppers. *arXiv preprint arXiv:2008.09768*, 2020.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216. PMLR, 2018.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2018.

Jerrold E Marsden and Tudor S Ratiu. Introduction to mechanics and symmetry. *Physics Today*, 48 (12):65, 1995.

Filippo Masi, Ioannis Stefanou, Paolo Vannucci, and Victor Maffi-Berthier. Thermodynamics-based artificial neural networks for constitutive modeling. *Journal of the Mechanics and Physics of Solids*, 147:104277, 2021.

Assad A Oberai, Nachiket H Gokhale, and Gonzalo R Feijóo. Solution of inverse problems in elasticity imaging using the adjoint method. *Inverse problems*, 19(2):297, 2003.

Hans Christian Oettinger. Irreversible dynamics, Onsager–Casimir symmetry, and an application to turbulence. *Physical Review E*, 90(4):042121, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Ravi G Patel, Indu Manickam, Nathaniel A Trask, Mitchell A Wood, Myoungkyu Lee, Ignacio Tomas, and Eric C Cyr. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *arXiv preprint arXiv:2012.05343*, 2020.

Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Franz M Rohrhofer, Stefan Posch, and Bernhard C Geiger. On the pareto front of physics-informed neural networks. *arXiv preprint arXiv:2105.00862*, 2021.

Samuel Rudy, Alessandro Alla, Steven L Brunton, and J Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2): 643–660, 2019.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Xiaocheng Shang and Hans Christian Öttinger. Structure-preserving integrators for dissipative systems based on reversible–irreversible splitting. *Proceedings of the Royal Society A*, 476(2234): 20190446, 2020.

Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Yunjin Tong, Shiying Xiong, Xingzhe He, Guanghan Pan, and Bo Zhu. Symplectic neural networks in taylor series form for hamiltonian systems. *Journal of Computational Physics*, page 110325, 2021.

Peter Toth, Danilo J Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *International Conference on Learning Representations*, 2019.

Nathaniel Trask, Andy Huang, and Xiaozhe Hu. Enforcing exact physics in scientific machine learning: a data-driven exterior calculus on graphs. *arXiv preprint arXiv:2012.11799*, 2020.

Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5): e0197704, 2018.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020.

Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7): 074602, 2018.

Yibo Yang, Mohamed Aziz Bhouri, and Paris Perdikaris. Bayesian differential programming for robust systems identification under uncertainty. *Proceedings of the Royal Society A*, 476(2243): 20200290, 2020.

Haijun Yu, Xinyuan Tian, Qianxiao Li, et al. Onsagernet: Learning stable and interpretable dynamics using a generalized onsager principle. *arXiv preprint arXiv:2009.02327*, 2020.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative symoden: Encoding hamiltonian dynamics with dissipation and control into deep learning. *arXiv preprint arXiv:2002.08860*, 2020.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. In *Learning for Dynamics and Control*, pages 1218–1229. PMLR, 2021.

Appendix A. Training without pruning

Tables 7 and 8 report the coefficients learned by training without the pruning-strategy. Although the algorithm successfully finds the significant coefficients, it fails to zero out the coefficients of the small contributions.

	Hyperbolic		Cubic oscillator		Van der Pol	
	$\dot{x} = -0.05x$ $\dot{y} = 1x^2 - 1y$		$\dot{x} = -0.1x^3 + 2y^3$ $\dot{y} = -2x^3 - 0.1y^3$		$\dot{x} = 1y$ $\dot{y} = -1x + 2y - 2x^2y$	
	$\dot{x}$	$\dot{y}$	$\dot{x}$	$\dot{y}$	$\dot{x}$	$\dot{y}$
1	-4.0265e-04	-1.5801e-04	-1.1837e-04	-1.9252e-04	7.6617e-04	-4.6053e-04
x	-4.8321e-02	6.7640e-04	-3.8775e-04	2.1818e-05	-4.3242e-04	-9.9930e-01
y	2.2759e-05	-9.9818e-01	3.4231e-04	5.1497e-04	1.0002e+00	1.9997e+00
x^2	1.1598e-04	1.0021e+00	8.5114e-04	-1.0675e-04	-4.3595e-06	-2.6983e-04
xy	-9.0811e-04	1.3638e-03	3.9183e-04	-1.0255e-04	1.3413e-03	1.3681e-04
y^2	-9.9513e-04	1.2584e-03	4.7589e-04	-3.6600e-04	8.9859e-04	-5.9787e-04
x^3	2.3237e-03	-1.3569e-03	-9.9985e-02	-2.0003e+00	6.0941e-05	-5.1159e-05
x^2y	-7.3889e-04	1.1560e-03	-8.8203e-04	8.3778e-04	1.6378e-04	-2.0006e+00
xy^2	-3.7055e-04	3.2972e-04	-3.6603e-04	-6.3260e-04	3.8560e-04	3.5658e-04
y^3	-2.8758e-03	3.6145e-04	2.0003e+00	-1.0056e-01	1.3837e-04	6.8370e-05

Table 7: [Hyperbolic, cubic oscillator, and Van der Pol problems] The coefficients learned by nSINDy via training without the magnitude-based pruning strategy.

	Hopf bifurcation				Lorenz 63		
	$\dot{x} = 1x\mu + 1y - 1x^3 - 1xy^2$ $\dot{y} = 1y\mu - 1x - 1x^2y - 1y^3$ $\dot{\mu} = 0$				$\dot{x} = -10x + 10y$ $\dot{y} = 28x - 1xz - 1y$ $\dot{z} = 1xy - 8/3z$		
	$\dot{x}$	$\dot{y}$	$\dot{\mu}$		$\dot{x}$	$\dot{y}$	$\dot{z}$
1	-7.7859e-05	-5.4043e-05	7.5873e-05	1	5.1030e-07	-5.2767e-07	8.4725e-07
x	-9.0900e-05	-1.0005e+00	-2.7873e-04	x	5.6920e-01	1.1720e+01	-3.6492e-07
y	9.9948e-01	-1.6792e-04	-3.3130e-04	y	4.5274e+00	7.2215e+00	1.4872e-07
μ	-2.5933e-04	5.7727e-04	4.2798e-04	z	5.4791e-07	-3.4551e-06	-2.6812e+00
x^2	-3.8502e-04	7.2785e-04	-2.8368e-04	x^2	-1.7573e-06	1.7074e-06	1.4821e-02
xy	1.6046e-03	-4.7278e-05	1.6167e-03	xy	-1.4202e-06	1.6468e-06	9.6046e-01
$x\mu$	9.9672e-01	-3.5927e-04	-5.1250e-04	xz	-9.4857e-01	4.6996e-01	-3.5173e-06
y^2	3.8864e-04	3.1930e-04	-2.0894e-04	y^2	-1.5292e-06	8.3760e-06	1.6890e-02
$y\mu$	-4.5870e-04	9.9483e-01	-8.0084e-04	yz	4.5105e-01	-6.7203e-01	-8.0195e-06
μ^2	2.6971e-04	5.6016e-04	2.3888e-04	z^2	-8.3810e-05	1.1768e-04	2.0347e-03
x^3	-9.9727e-01	1.3899e-05	-1.2669e-03	x^3	-5.9848e-02	9.4702e-02	-8.1650e-07
x^2y	-2.1682e-03	-9.9460e-01	-5.3697e-04	x^2y	6.0315e-02	-8.9793e-02	-4.1083e-06
$x^2\mu$	-2.4824e-04	5.5955e-04	-2.7237e-04	x^2z	-5.6009e-06	5.7127e-06	-1.1296e-04
xy^2	-9.9607e-01	-1.2029e-03	-4.7030e-04	xy^2	-1.9930e-02	2.6786e-02	1.4347e-06
$xy\mu$	5.4106e-04	8.6669e-04	-8.0186e-04	xyz	1.1614e-05	-1.3253e-05	8.0456e-04
$x\mu^2$	-9.1186e-04	1.8879e-06	-1.3931e-04	xz^2	2.1789e-02	-3.3973e-02	-3.2658e-07
y^3	7.2948e-05	-9.9499e-01	-1.5385e-04	y^3	2.8419e-03	-3.5977e-03	-1.2999e-07
$y^2\mu$	3.1669e-04	-2.1393e-04	1.0944e-03	y^2z	4.1031e-07	-4.9989e-07	-3.6131e-04
$y\mu^2$	-4.0919e-04	-5.7416e-04	7.1154e-04	yz^2	-8.8740e-03	1.2924e-02	8.9630e-07
μ^3	3.7226e-04	1.8204e-04	9.8719e-05	z^3	2.7687e-06	-3.6289e-06	-6.3774e-05

Table 8: [Hopf bifurcation and Lorenz 63] The coefficients learned by nSINDy via training without the magnitude-based pruning strategy.