Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 015031 https://doi.org/10.1088/2632-2153/ac567a

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
30 October 2021

REVISED
9 January 2022

ACCEPTED FOR PUBLICATION
17 February 2022

PUBLISHED
8 March 2022

PAPER

Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data

Kadierdan Kaheman[1,*], **Steven L Brunton**[1] and **J Nathan Kutz**[2]

[1] Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, United States of America
[2] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: kadierk@uw.edu

Keywords: automatic differentiation, denoising, nonlinear dynamics, optimization, sparse identification, machine learning, discrepancy modeling

Abstract

The *sparse identification of nonlinear dynamics* (SINDy) is a regression framework for the discovery of parsimonious dynamic models and governing equations from time-series data. As with all system identification methods, noisy measurements compromise the accuracy and robustness of the model discovery procedure. In this work we develop a variant of the SINDy algorithm that integrates automatic differentiation and recent time-stepping constrained motivated by Rudy *et al* (2019 *J. Computat. Phys.* **396** 483–506) for simultaneously (1) denoising the data, (2) learning and parametrizing the noise probability distribution, and (3) identifying the underlying parsimonious dynamical system responsible for generating the time-series data. Thus within an integrated optimization framework, noise can be separated from signal, resulting in an architecture that is approximately twice as robust to noise as state-of-the-art methods, handling as much as 40% noise on a given time-series signal and explicitly parametrizing the noise probability distribution. We demonstrate this approach on several numerical examples, from Lotka-Volterra models to the spatio-temporal Lorenz 96 model. Further, we show the method can learn a diversity of probability distributions for the measurement noise, including Gaussian, uniform, Gamma, and Rayleigh distributions.

1. Introduction

The data-driven discovery of governing equations is an emerging field within the machine learning and artificial intelligence communities. From traditional model regression techniques [1–5] to sparse regression modeling [6–10], dynamic mode decomposition [11, 12], *neural networks* (NN) [13–19, 19, 32], genetic algorithm [20, 21] or even Koopman modeling techniques [22–24], a diversity of methods are emerging that transform time-series data (or spatio-temporal data) into representations of governing equations of motion [25–31]. The interpretability and generalizability of these discovered equations of motion are critical for understanding, designing, and controlling complex systems. As such, the *sparse identification of nonlinear dynamics* (SINDy) [7] framework provides a compelling regression framework since discovered models are interpretable and parsimonious by design. As with all system identification algorithms, noisy measurements compromise the accuracy and robustness of the model discovery procedure. Moreover, many optimization frameworks rely explicitly on the assumption of Gaussian noise, which is rarely true in the real world. Recently, Rudy *et al* [32] developed a novel optimization framework for separating signal and noise from noisy time-series data by identifying a deep NN model for the signal from numerical time-stepping constraints such as a Runge-Kutta. In this work, we build on this framework and leverage automatic differentiation [33] in the optimization procedure to simultaneously denoise data and identify sparse nonlinear models via SINDy. This new architecture yields significant improvements in model discovery, including superior separation of the signal from noise while simultaneously characterizing the noise distribution.

SINDy has emerged as a flexible and promising architecture for model discovery due to its inherent parsimonious representation of dynamics. The SINDy framework relies on sparse regression on a library of candidate model terms to select the fewest terms required to describe the observed dynamics [7]. Specifically, SINDy is formulated as an over-determined linear system of equations $\mathbf{A}\boldsymbol{\xi} = \mathbf{b}$, where A is a library matrix, b represents the measurement data, and ξ represents the sparse selection vector, and the sparsity of the solution is promoted by the ℓ_0-norm $\|\boldsymbol{\xi}\|_0$. Thus sparsity is a proxy for parsimony, interpretability, and generalizability. Measurement noise, however, is always present, and it corrupts the ability of the SINDy regression framework, and indeed any other model discovery paradigm, to accurately extract governing models.

There are many variants of sparse regression, all of which typically attempt to approximate the solution to an NP-hard, ℓ_0-norm penalized regression. Sparsity-promoting methods like the LASSO [34, 35] use the ℓ_1-norm as a proxy for sparsity since tractable computations can be performed. The iterative least-squares thresholding technique of the SINDy algorithm promotes sparsity through a sequential procedure. Recently, Zhang and Schaeffer [36] have provided several rigorous theoretical guarantees on the convergence of the SINDy algorithm. Specifically, they proved that the algorithm approximates local minimizers of an unconstrained ℓ_0-penalized least-squares problem, which allows them to provide sufficient conditions for general convergence, the rate of convergence, and conditions for one-step recovery. Using a relaxed formulation, Champion *et al* [37] show how the SINDy regression framework can accommodate additional structure, robustness to outliers, and nonlinear parameter estimation using the *sparse relaxed regularized regression* (SR3) formulation [38]. SINDy results in interpretable models, and it has been widely applied in many scientific disciplines [39–54]. Moreover, it has been extended to incorporate control [55, 56], rational or implicit dynamics [10, 57, 58], partial differential equations [8, 59], parametric model dependencies [60], discrepancy models [47, 48], multiscale physics [61], stochastic dynamics [62], constrained physics [42], among many other innovations [9, 37, 40, 62–72]. The `PySINDy` Python package executes many of these variants [73].

Despite its flexibility, modularity, and extensibility, SINDy and its variants typically rely on approximating time-derivative of the measured time-series data. Computing derivatives of noisy measurement data is known to be a challenging problem, with many algorithmic innovations and mathematical architectures developed to produce accurate derivative approximations [74]. These methods include finite-differences, spectral methods [75], spline smoothing, filtering procedures, polynomial fitting, low-rank projection, and total variations, to highlight some of the diverse techniques employed for this critical task of scientific computing. This task is made even more difficult, depending upon the noise statistics. Gaussian noise is often easier to learn and characterize than noise distributions that have non-zero means and are not symmetric. Ultimately, there is a need for methods that are robust to noisy measurements and diverse probability distributions.

Recent innovations in automatic differentiation have enabled the solution of an optimization problem directly related to the computation of the required derivatives [32]. Since its inception, automatic differentiation has been widely used in the machine learning community to enable complicated optimization problems without manually computing Jacobians [32, 33, 76–83]. More recently, this approach has been used with NNs to separate a signal from noise and model the signal when a model is unknown [32], and to improve Kalman smoothing when the governing equations are known [80]. The success of these algorithms suggest that they could be leveraged for noise signal separation in the SINDy framework. In this work, we extend this simultaneous de-noising and discovery approach to SINDy. Specifically, automatic differentiation enables differentiation with respect to the functions in the SINDy library, thus circumventing a direct differentiation of the noisy time-series data. The modified SINDy algorithm is more robust to noise and further allows for an explicit characterization (discovery) of the underlying probability distribution of the noise, something that current state-of-the-art methods cannot do and is a unique feature of our method.

In section 2, we illustrate the modified SINDy algorithm. In section 3, we show the comparison between modified SINDy and noise signal separation approach based on the NN proposed by Rudy *et al* [32]. In section 4, we show the use of modified SINDy on various numerical examples. We also show how modified SINDy can be used to extract the noise distribution information and how it can be used in the discrepancy modeling framework. In section 5, we show our conclusions and possible future improvements.

2. Methods

In what follows, we introduce the basic mathematical architecture behind the SINDy algorithm, demonstrating explicitly its sensitivity to noisy measurements. This guides our introduction of the modified SINDy for simultaneously learning the system model and denoising the signal.

2.1. Sparse identification of nonlinear dynamics

The SINDy algorithm [7] provides a principled, data-driven discovery method for nonlinear dynamics of the form

$$\frac{d}{dt}\mathbf{x}(t) = \boldsymbol{f}(\mathbf{x}(t)), \tag{1}$$

where $\mathbf{x}(t) = [x_1(t), x_2(t), \cdots, x_n(t)] \in \mathbb{R}^{1\times n}$ is system states represented as a row vector. SINDy posits a set of candidate functions that would characterize the right hand side of the governing equations. Candidate model terms form the library $\boldsymbol{\Theta}(\mathbf{X}) = [\theta_1(\mathbf{X}), \theta_2(\mathbf{X}), \cdots, \theta_p(\mathbf{X})] \in \mathbb{R}^{m\times p}$ of potential right hand side terms, where $\mathbf{X} = [\mathbf{x}(t_1); \mathbf{x}(t_2); \cdots; \mathbf{x}(t_m)] \in \mathbb{R}^{m\times n}$ is formed by m row vectors. This then allows for the formulation of a regression problem to select only the few candidate terms necessary to describe the dynamics:

$$\arg\min_{\boldsymbol{\Xi}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|_2 + \lambda\|\boldsymbol{\Xi}\|_0, \tag{2}$$

where the matrix $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_n] \in \mathbb{R}^{p\times n}$ is comprised of the sparse vectors $\boldsymbol{\xi}_i \in \mathbb{R}^{p\times 1}$ that select candidate model terms. The amount of sparsity promotion is controlled by the parameter λ, which determines the penalization by the ℓ_0-norm. The $\theta_i(\mathbf{X})$ can be any candidate function that may describe the system dynamics $\boldsymbol{f}(\mathbf{x}(t))$ such as trigonometric functions $\theta_i(\mathbf{X}) = \cos(\mathbf{X})$ or polynomial functions $\theta_i(\mathbf{X}) = \mathbf{X}^3$, for example. By solving equation (2), we can identify a model of system dynamics:

$$\frac{d}{dt}\mathbf{x}(t) = \boldsymbol{f}(\mathbf{x}(t)) \approx \boldsymbol{\Theta}(\mathbf{x}(t))\boldsymbol{\Xi}. \tag{3}$$

Many different optimization techniques can be used to obtain the sparse coefficients $\boldsymbol{\Xi}$, such as sequentially thresholded least squares [7, 36], LASSO [35], sparse relaxed regularized regression (SR3) [37, 38], stepwise sparse regression (SSR) [84], and Bayesian approaches [58, 85].

In practice, noise-free measurements of $\mathbf{x}(t)$ are not available, and only the full state noisy measurement:

$$\mathbf{y}(t) = \mathbf{x}(t) + \mathbf{n}(t), \tag{4}$$

is provided to SINDy from sensors, where $\mathbf{y}(t) = [y_1(t), y_2(t), \cdots, y_n(t)] \in \mathbb{R}^{1\times n}$ is noisy measurement and $\mathbf{n}(t) = [n_1(t), n_2(t), \cdots, n_n(t)] \in \mathbb{R}^{1\times n}$ is the noise added to true state. Thus, equation (2) then becomes:

$$\dot{\mathbf{Y}} = \dot{\mathbf{X}} + \dot{\mathbf{N}} = \boldsymbol{\Theta}(\mathbf{Y})\boldsymbol{\Xi} = \boldsymbol{\Theta}(\mathbf{X}+\mathbf{N})\boldsymbol{\Xi}, \tag{5}$$

where $\mathbf{Y} = [\mathbf{y}(t_1); \mathbf{y}(t_2); \cdots; \mathbf{y}(t_m)] \in \mathbb{R}^{m\times n}$ is noisy measurement matrix formed by m row vectors measurement of size $1 \times n$ and $\mathbf{N} = [\mathbf{n}(t_1); \mathbf{n}(t_2); \cdots; \mathbf{n}(t_m)] \in \mathbb{R}^{m\times n}$ is noise matrix also formed by m row vector of size $1 \times n$. From equation (5), note that the solution $\boldsymbol{\Xi}$ is no longer the same $\boldsymbol{\Xi}$ shown in equation (2) due to the presence of noise. Moreover, the noise will be magnified when approximating the derivatives $\dot{\mathbf{X}}$ by a factor of $\mathcal{O}(1/dt)$ [8], and it will non-linearly corrupt the library matrix $\boldsymbol{\Theta}$. Extensive research has been done to improve the robustness of the SINDy framework. The integral formulation [65] and weak formulation [59, 71, 72, 86] improved the regression robustness by avoiding taking derivative of noisy data. Other approaches, such as subsampling [87], increased the noise robustness of the SINDy framework by doing regression on the subsampled measurement that has less noise. Corrupt data can also be handled with methods from robust statistics [37, 64]. In the next section, we introduce an alternative approach that simultaneously learns the noise $\mathbf{N}$ while using the denoised data to perform model identification.

2.2. Simultaneously denoising and learning system model

To improve the noise robustness of the SINDy regression, we determine the estimated noise $\hat{\mathbf{n}}(t) \in \mathbb{R}^{1\times n}$ as a hyper-parameter and formulate $\hat{\mathbf{N}} = [\hat{\mathbf{n}}(t_1); \hat{\mathbf{n}}(t_2); \cdots; \hat{\mathbf{n}}(t_m)] \in \mathbb{R}^{m\times n}$ in order to optimize the difference between the estimated derivative and system's vector field such that:

$$e_d = \|\dot{\hat{\mathbf{X}}} - \boldsymbol{\Theta}(\hat{\mathbf{X}})\boldsymbol{\Xi}\|_2^2, \tag{6}$$

where $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}}$ is formed by m estimated true states $\hat{\mathbf{x}}(t) \in \mathbb{R}^{1\times n}$, and e_d is the derivative approximation error. For details on calculating this error, please see appendix L. Note that $\hat{\mathbf{X}} = [\hat{\mathbf{x}}(t_1); \hat{\mathbf{x}}(t_2); \cdots; \hat{\mathbf{x}}(t_m)] \in \mathbb{R}^{m\times n}$. When $\hat{\mathbf{N}} = \mathbf{N}$, the effect of noise can be eliminated, and the accuracy of SINDy will be significantly improved. However, there exist many trivial solutions for minimizing the equation (6) with two uncorrelated optimization parameters $\hat{\mathbf{N}}$ and $\boldsymbol{\Xi}$. Thus, an additional constraint is needed to regularize equation (6).

The additional constraint proposed here uses the estimated vector field of the system model similar to the one proposed by Rudy *et al* [32]. Equation (3) gives the estimate $\boldsymbol{\Theta}(\mathbf{x}(t))\boldsymbol{\Xi}$ of the true vector field $f(\mathbf{x}(t))$. Integrating over a segment of time t_j to t_{j+1} gives the integrated vector field, or flow map,

$$\mathbf{x}(j+1) = \mathbf{F}(\mathbf{x}(j)) = \mathbf{x}(j) + \int_{t_j}^{t_{j+1}} \boldsymbol{\Theta}(\mathbf{x}(\tau))\boldsymbol{\Xi}\, d\tau. \tag{7}$$

This can be generalized to integrate the system either forward or backward in time q steps. This gives:

$$\mathbf{x}(j+q) = \mathbf{F}^q(\mathbf{x}(j)) = \mathbf{x}(j) + \int_{t_j}^{t_{j+q}} \boldsymbol{\Theta}(\mathbf{x}(\tau))\boldsymbol{\Xi}\, d\tau. \tag{8}$$

To obtain the $\mathbf{x}(j+q)$ in equation (8), a numerical simulation scheme such as Runge-Kutta can be used. In what follows, we employ a 4th-order Runge-Kutta method to simulate the dynamics forward/backward in time q-steps. Similar to equation (8), when the noisy measurement data $\mathbf{y}$ is given, the estimated state $\hat{\mathbf{x}} = \mathbf{y} - \hat{\mathbf{n}}$ satisfies:

$$\mathbf{y}(j+q) - \hat{\mathbf{n}}(j+q) = \hat{\mathbf{x}}(j+q) = \hat{\mathbf{F}}^q(\hat{\mathbf{x}}(j)) = \hat{\mathbf{x}}(j) + \int_{t_j}^{t_{j+q}} \boldsymbol{\Theta}(\hat{\mathbf{x}}(\tau))\boldsymbol{\Xi}\, d\tau, \tag{9}$$

when $\hat{\mathbf{n}} = \mathbf{n}$ and the exact value of $\boldsymbol{\Xi}$ is known. Thus, by minimizing

$$e_{s,j} = \sum_{i=-q, i\neq 0}^{q} \omega_i \|\mathbf{y}(j+i) - \hat{\mathbf{n}}(j+i) - \hat{\mathbf{F}}^i(\hat{\mathbf{x}}(j))\|_2^2, \tag{10}$$

the optimization parameters $\hat{\mathbf{N}}$ and $\boldsymbol{\Xi}$ are coupled, resulting in additional structural constraint of the model. The parameter ω_i is used to account for the numerical error and is set to $\omega_i = c^{|i|-1}$, where $0 < c \leq 1$ is a constant (throughout this paper, we use $c = 0.9$). The use of ω suggests that the simulation error too far ahead in the future, or too far backward in the past, should be penalized less due to the error of the numerical simulation scheme. The error incurred by simulating the vector filed forward/backward on the entire trajectory can be written as:

$$e_s = \sum_{j=q+1}^{m-q} e_{s,j} = \sum_{j=q+1}^{m-q} \sum_{i=-q, i\neq 0}^{q} \omega_i \|\mathbf{y}(j+i) - \hat{\mathbf{n}}(j+i) - \hat{\mathbf{F}}^i(\hat{\mathbf{x}}(j))\|_2^2. \tag{11}$$

Using subscripts to represent the time step, the final cost function is then:

$$\mathcal{L}(\boldsymbol{\Xi}, \hat{\mathbf{N}}) = e_s + e_d = \sum_{j=q+1}^{m-q} \sum_{i=-q, i\neq 0}^{q} \omega_i \|\mathbf{y}_{j+i} - \hat{\mathbf{n}}_{j+i} - \hat{\mathbf{F}}^i\left(\hat{\mathbf{x}}_j\right)\|_2^2 + \|\dot{\hat{\mathbf{X}}} - \boldsymbol{\Theta}(\hat{\mathbf{X}})\boldsymbol{\Xi}\|_2^2, \tag{12}$$

which is the summation of the derivative approximation error e_d and simulation error e_s. The optimization problem to simultaneously denoise and learn the system model can then be written as:

$$\begin{aligned} \boldsymbol{\Xi}, \hat{\mathbf{N}} &= \arg\min_{\boldsymbol{\Xi}, \hat{\mathbf{N}}} \mathcal{L}(\boldsymbol{\Xi}, \hat{\mathbf{N}}), \\ \text{s.t.} \quad & (|\boldsymbol{\Xi}| < \lambda) = 0. \end{aligned} \tag{13}$$

The global optimal solution for equation (13) needs to satisfy $\hat{\mathbf{N}} = \mathbf{N}$ and $\boldsymbol{f}(\mathbf{x}) = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\Xi}$. However, the global optimum of equation (12) is not unique, as appendix K suggests. To solve for equation (13), it is necessary to calculate the Jacobian $\partial\mathcal{L}/\partial\hat{\mathbf{N}}$ and $\partial\mathcal{L}/\partial\boldsymbol{\Xi}$, which is a difficult task to do analytically or computationally. However, recent automatic differentiation packages such as Tensorflow [76] and Julia Flux [88] make it possible to directly extract the gradients of $\mathcal{L}$ with respect to $\hat{\mathbf{N}}$ and $\boldsymbol{\Xi}$. Other alternatives include JAX MD [89, 90]. This allows us to solve the optimization problem in equation (13) easily using gradient descent method such as Adam [91]. Throughout this paper, we use the Tensorflow 2.0 and Adam optimizer to solve the equation (13). Moreover, to enforce the sparsity of the identified model, a thresholding approach [7] is used and the equation (13) is solved for N_{loop} times (the sparsity is enforced to the model structure $\boldsymbol{\Xi}$ not the noise $\mathbf{N}$). Each iteration uses the previous iteration's optimization result $\hat{\mathbf{N}}$ as the initial guess of the new iteration. The values of $\hat{\mathbf{N}}$ is also used to calculate the estimated state $\hat{\mathbf{X}}$, which is used to calculate the new estimated values of the selection parameter $\boldsymbol{\Xi}$. Furthermore, if the elements in $|\boldsymbol{\Xi}|$ are smaller than a threshold λ at the end of an optimization loop, those elements will be constrained to zero for

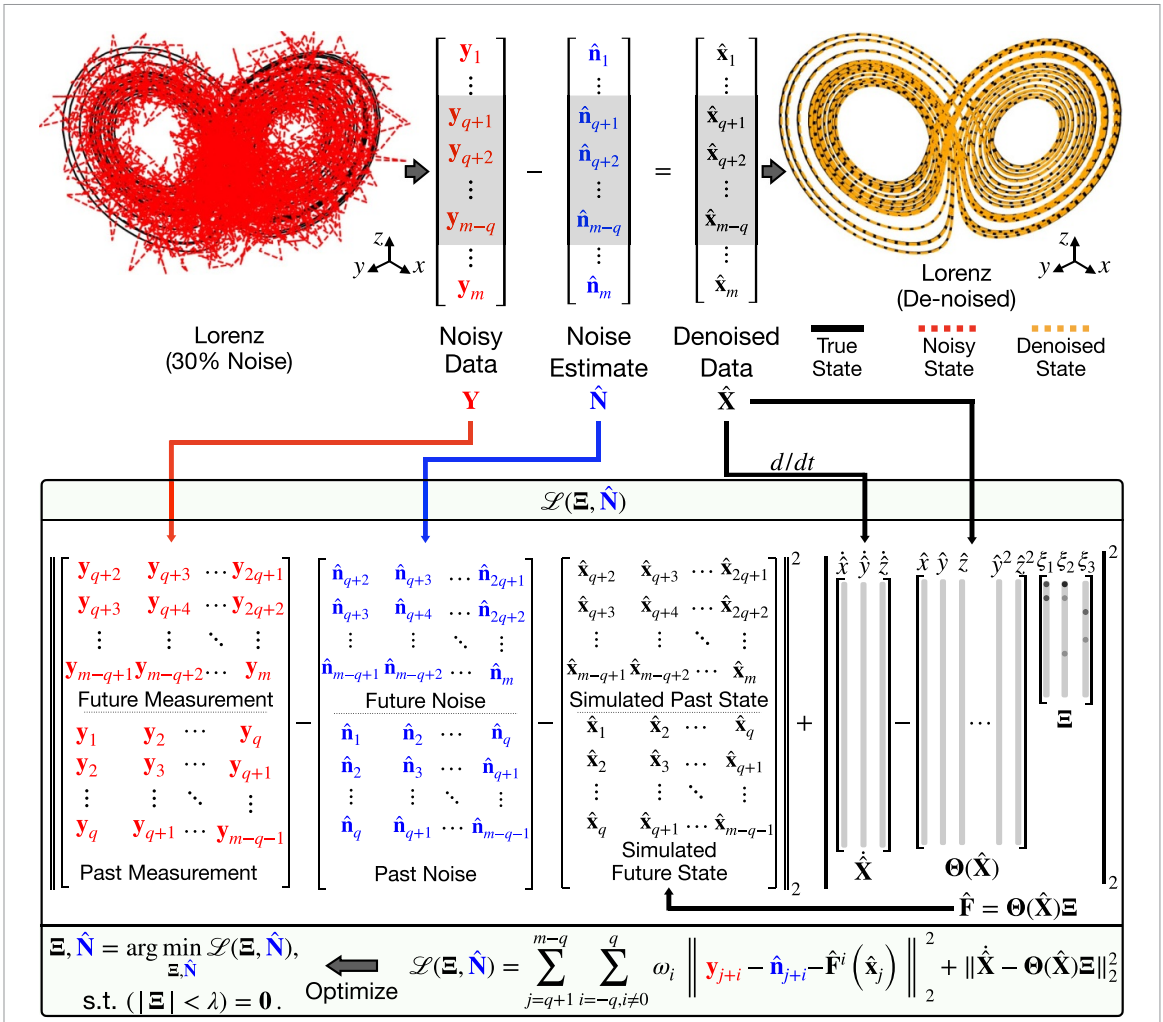

Figure 1. This figure illustrates modified SINDy algorithm. The goal is to learn the system model $\Theta(\mathbf{X})\Xi$ and noise $\mathbf{N}$. The noise is subtracted from the measurement to obtain the clean data. To achieve this, the estimated noise $\hat{\mathbf{N}}$ is set as an optimization parameter and the cost function $\mathcal{L}(\Xi, \hat{\mathbf{N}})$ is minimized. This optimization is performed 8 times, and the small values of $|\Xi|$ is enforced to be zero for the remainder of the optimization process. The proposed de-noising algorithm is related to the scheme of Rudy *et al* [32], but embedded into the SINDy model discovery framework.

the remainder of the optimization process. Figure 1 illustrates this process, and appendix A shows the detailed algorithm for simultaneous denoising and sparse model identification. Some guidance on the selection of the hyper-parameters λ, q, and N_{loop} is given in appendices B–D.

3. Performance comparison with neural network denoising approach

The advocated optimization framework of modified SINDy is compared with a NN denosing approach by Rudy *et al* [32]. Additionally, the robustness to noise and the amount of data is considered.

3.1. Performance criteria

For ease of comparison, we use the same performance criteria developed by Rudy *et al* [32]. Specifically, these are the vector field error E_f, the noise identification error $E_{\mathbf{N}}$, and the prediction error $E_{\mathbf{F}}$. The vector filed error is:

$$E_f = \frac{\sum_{i=1}^{m} \left\| \boldsymbol{f}(\mathbf{x}_i) - \hat{\boldsymbol{f}}(\mathbf{x}_i) \right\|_2^2}{\sum_{i=1}^{m} \|\boldsymbol{f}(\mathbf{x}_i)\|_2^2}, \tag{14}$$

which calculates the relative squared ℓ_2 error between the true vector filed and identified vector field $\hat{\boldsymbol{f}}$. The noise identification error is:

$$E_{\mathbf{N}} = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{n}_i - \hat{\mathbf{n}}_i\|_2^2, \tag{15}$$

which is the mean ℓ_2 difference between the true noise $\mathbf{N}$ and identified noise $\hat{\mathbf{N}}$. The prediction error is:

$$E_{\mathbf{F}} = \frac{1}{\|\mathbf{X}\|_F^2} \sum_{i=1}^{m-1} \left\| \mathbf{x}_i - \hat{\mathbf{F}}^i(\mathbf{x}_1) \right\|_2^2, \tag{16}$$

and it calculates the difference between forward simulation trajectory and true trajectory. For comparison of modified SINDy and recently published Weak-SINDy [72], as shown in appendix F, two more performance criteria are used. The first one is the normalized parameter error:

$$E_{\boldsymbol{p}} = \frac{\|\boldsymbol{\Xi} - \hat{\boldsymbol{\Xi}}\|_2}{\|\boldsymbol{\Xi}\|_2}, \tag{17}$$

which reflects how much the identified parameters $\hat{\boldsymbol{\Xi}}$ is off from the true parameters $\boldsymbol{\Xi}$. The other one is the success rate, which describes the percentage of identifying the model's correct structure in multiple trials.

3.2. Robustness to noise

The Lorenz attractor is used as an example to test the noise robustness of the approach. The model of the chaotic Lorenz is:

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z, \end{aligned} \tag{18}$$

where $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. The Lorenz attractor is simulated with initial condition $x_0 = [5, 5, 25]$, $T = 25$, and $dt = 0.01$. The prediction step is chosen as $q = 3$ for both approaches compared and $N_{loop} = 6$ for our proposed method. Unless otherwise noted, Adam optimizer is used to optimize the problem with maximum iteration set to 5000 for modified SINDy and 30 000 for NN approach [32]. Different magnitudes of Gaussian noise are added to generate the noisy training data. The noise level is defined as:

$$\text{Noise Level } (\%) = \sqrt{\frac{\text{var(Noise)}}{\text{var(Signal)}}} \times 100\% = \frac{\text{std(Noise)}}{\text{std(Signal)}} \times 100\%. \tag{19}$$

For each noise level, 10 different sets of noisy data are generated and used as data for both approaches. The NN approach [32] uses the same set up as [32], with 3 hidden layers, and each layer having 64 neurons. Moreover, the regularization parameter is chosen as 10^{-8}, and the penalty for $\hat{\mathbf{N}}$ is chosen as 10^{-5}. Unless otherwise noted, we use the same set up for all the NNs in this paper. For modified SINDy, the library is constructed with terms up to second order (not including the constant term). Moreover, the value of the sparsity parameter λ varies based on the noise added. For most of the case, $\lambda = 0.1$. A Tikhonov regularization approach is used to pre-smooth the noisy data as in [32], although we have found that pre-smoothing does not affect the results appreciably when using zero-mean noise.

Figure 2 show the noise identification error of the NN approach [32] and the modified SINDy approach. The vector field error and short term prediction error can be seen in figure 3. For all the noise levels, modified SINDy correctly identified the Lorenz model. To calculate the prediction error, the identified model is simulated 6 seconds forward in time, with $dt = 0.01$, for both modified SINDy and NN denoising approach [32]. Figure 3 suggests that modified SINDy identified model has better performance when simulated forward in time. Other useful way to determine the forward simulation accuracy is by using the Lyapunov exponent of the model. However, unlike the standard way of defining the Lyapunov exponent [92], where the parameters of the model are perfect and only initial conditions of the simulation is perturbed by ϵ, in the situation shown in figure 3, we have the exact opposite. In figure 3, the parameters of the identified system is off by ϵ while the initial conditions are perfectly known. Thus, some modification of the definition of Lyapunov exponent is needed before it can be used to define the model forward simulation accuracy. Appendix E shows noise robustness comparison between the modified SINDy and original SINDy [7]. In general, the modified SINDy is about 2 times more robust than original SINDy [7]. A comparison between modified SINDy and the recently developed Weak-SINDy approach [72] is presented in appendix F.

3.3. Robustness to data length

We also compare the performance of the NN denoising approach by Rudy *et al* [32] and modified SINDy under different data usage with a fixed noise level. The minimum amount of data needed by modified SINDy to correctly identify the system model is shown by using Lorenz attractor as an example. To perform the

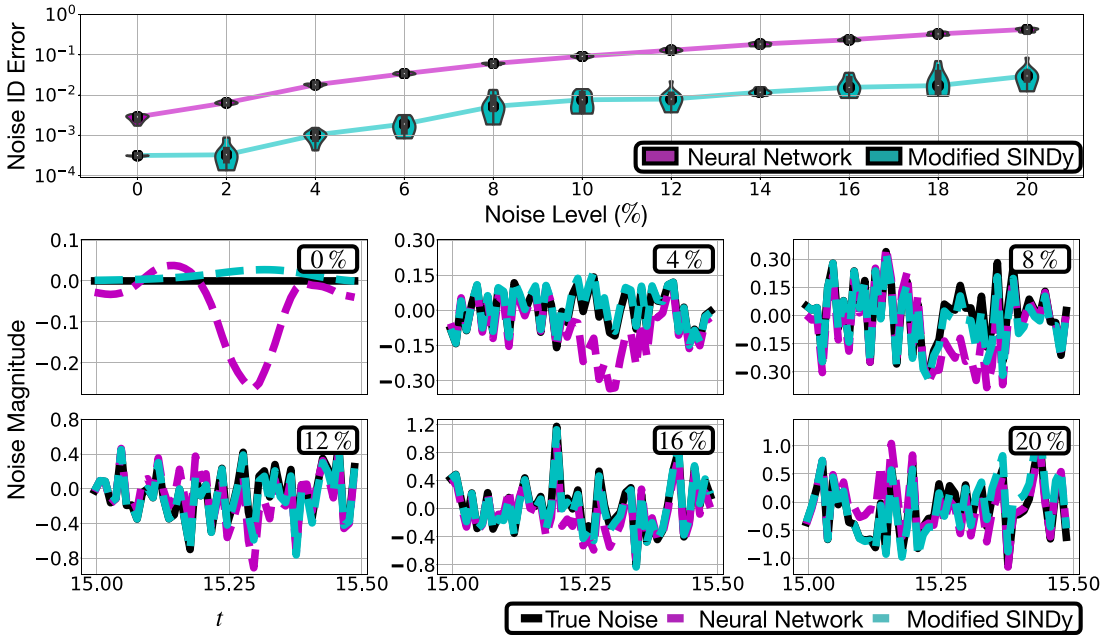

Figure 2. Top: Noise identification error of modified SINDy (labeled as SINDy) and NN denoising approach by Rudy *et al* [32]. The black circle represents the median of ten runs while the violin shape represents the distribution of error. The modified SINDy approach shows better noise identification error. Bottom: Comparison between the average noise applied to the Lorenz system and the noise identified by the two approaches. As shown on the left, both approaches can not produce the correct zero noise result when no noise is applied, which happens since there is a tiny difference between the learned dynamics and true dynamics.

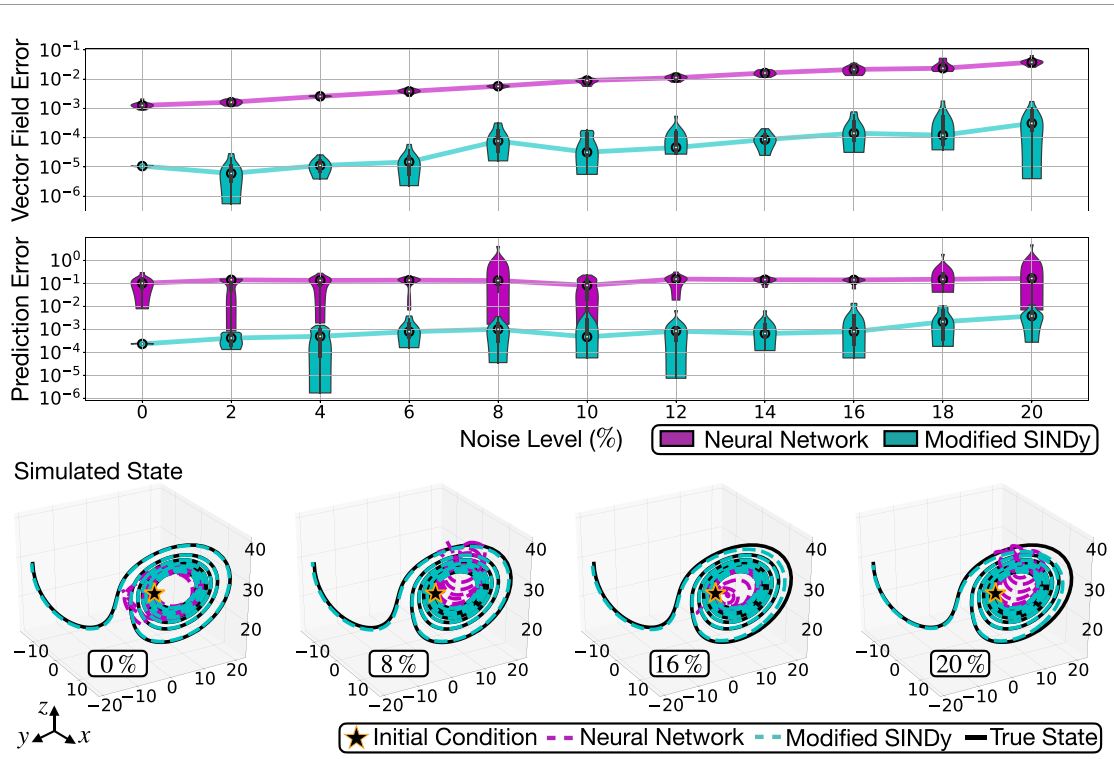

Figure 3. Top: The vector field error and prediction error of modified SINDy (labeled as SINDy) and NN denoising approach by Rudy *et al* [32] is shown. The black dot is the median of the 10 runs, and the violin shape represents the distribution of the error. Bottom: The simulated trajectory is shown with the initial condition chosen as $x_0 = [5, 5, 25]$.

numerical experiment, the same initial point, $x_0 = [-5, 5, 25]$, is used to generate noise-free data of different temporal lengths. The time step is fixed at $dt = 0.01$ with 10% of Gaussian noise added to generate noisy training data. The success rate of modified SINDy is calculated to indicate the minimum amount of data needed to identify the correct system model. The prediction error is not shown since the simulation of the

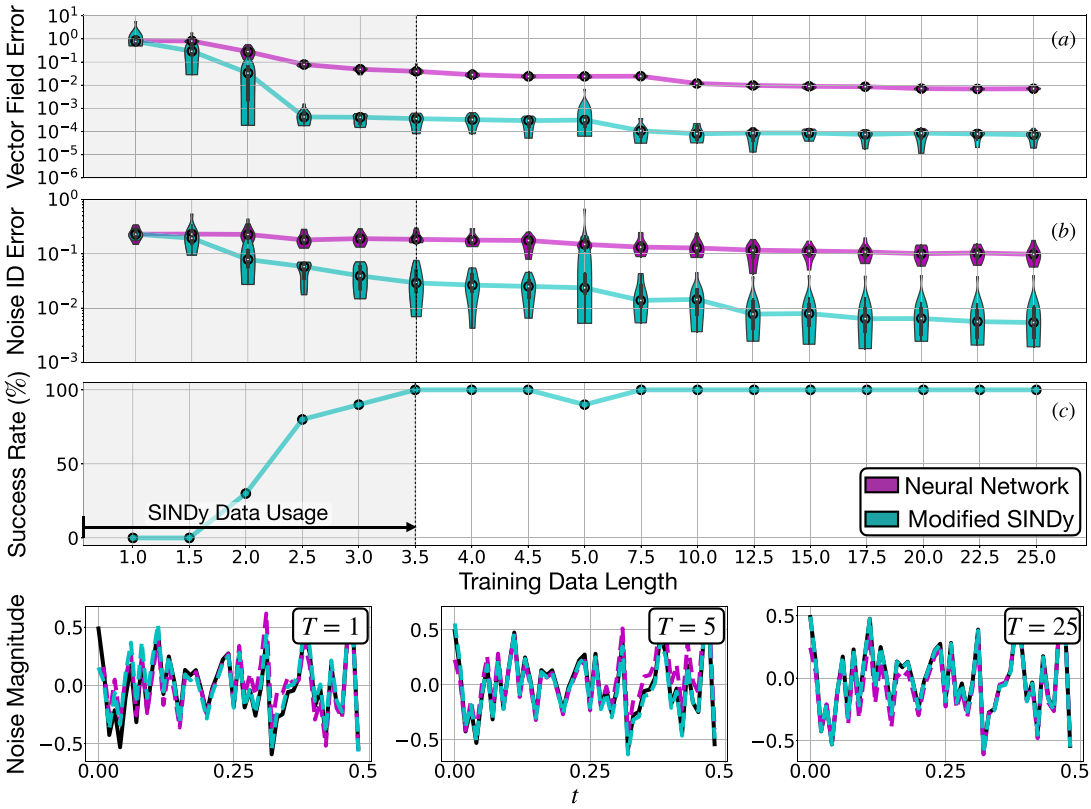

Figure 4. (a), (b): As the training data length increases, the vector field error and noise identification error of modified SINDy (labeled as SINDy) and NN denoising approach by Rudy *et al* [32] decreases. (c) The modified SINDy can use 3.5 seconds of data to identify the system model with 100% accuracy. There is a tiny drop in the success rate when the training data length is 5 seconds due to the choice of a fixed thresholding parameter. By using a larger thresholding parameter, the success rate can be back to 100%. Moreover, since the NN is a black box model, we will not be able to determine whether it learned the correct symbolic model or not, thus success rate of NN is not plotted. Bottom: a comparison of averaged true noise and identified noise by two approaches is shown.

identified model in the low data limit is not stable. With a learning rate of 0.001, Adam is used to optimize the problem with the prediction step set to $q = 3$ for both approaches. A fixed thresholding parameter $\lambda = 0.1$ with $N_{loop} = 6$ is used for modified SINDy and the library is constructed with up to second order terms (without constant term added). Figure 4 suggests that when the correct parameters and library is used for modified SINDy, it will out-perform the NN denoising approach by Rudy *et al* given the same amount of data.

4. Results

In this section, we demonstrate the ability of modified SINDy to separate signal and noise while learning the system model. The Van der Pol oscillator will be used as the example test case to show that modified SINDy can identify the correct distribution of the Gaussian noise added to the system. Additionally, we highlight several other examples tested with modified SINDy and summarize the performance. Furthermore, as a more advanced example, we show that modified SINDy can be used to separate non-Gaussian, non-zero mean, and non-symmetric noise distributions from the dynamics. Finally, we show how modified SINDy can be integrated to the discrepancy modeling approach [48].

4.1. Van der Pol oscillator

The Van der Pol oscillator is used as our test case to demonstrate the ability of modified SINDy to denoise and learn the system dynamics simultaneously. The Van der Pol oscillator is given by:

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= \mu\left(1 - x^2\right) y - x, \end{aligned} \tag{20}$$

where the nonlinear damping/gain parameter $\mu = 0.5$ is used for demonstration purposes. The system is simulated with initial condition $[-2, 1]$, $T = 10$, and $dt = 0.01$. The Adam optimizer with learning rate of

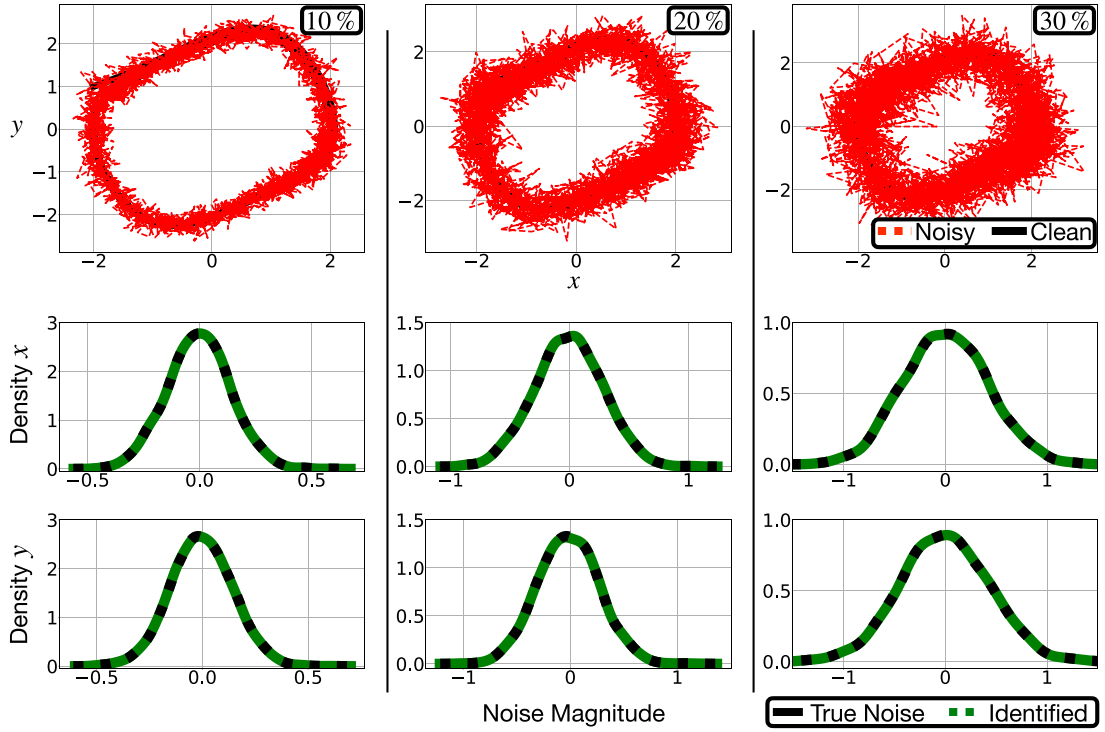

Figure 5. Distribution of the noise learned by modified SINDy is shown.

0.001 is used for all noise levels. The parameters of modified SINDy are chosen as $q = 1$ and $\lambda = 0.05$, and the library of candidate functions is constructed with polynomial terms up to third order (without constant term). Three different levels of noise are applied and the distribution of identified noise is shown in figure 5. Figure 5 shows that modified SINDy correctly identified the distribution of true noise.

4.2. Rössler attractor

The second example we use is the Rössler attractor that is governed by:

$$\begin{aligned} \dot{x} &= -y - z, \\ \dot{y} &= x + ay, \\ \dot{z} &= b + z(x - c), \end{aligned} \tag{21}$$

where $a = 0.2$, $b = 0.2$, and $c = 5.7$. The system is simulated with initial condition $[3, 5, 0]$, $T = 25$, and $dt = 0.01$. The Adam optimizer with learning rate of 0.001 is used for all noise levels. The parameters of modified SINDy are chosen as $q = 1$ and $\lambda = 0.05$, and the library of candidate functions is constructed with polynomial terms up to second order (with constant term). Three different levels of noise are applied and the denoised signal is shown in figure 6. Figure 6 also shows the simulated trajectories of the identified models. The initial condition $[3, 5, 0]$, $T = 25$, and $dt = 0.01$ are used to simulate the identified models.

4.3. Lorenz 96 model

As our last example, we use the modified SINDy to identify Lorenz 96 model whose equation is given by:

$$\dot{x}_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \tag{22}$$

for $i = 1, 2, \ldots, N$. We assume $x_{-1} = x_{N-1}$, $x_0 = x_N$, $x_1 = x_{N+1}$, and set forcing term F as 8 to generate chaotic behavior. The number N is set as 4 such that the model has 6 states. The system is simulated with initial condition $[1, 8, 8, 8, 8]$, $T = 25$, and $dt = 0.01$. The Adam optimizer with learning rate of 0.001 is used for all noise levels. The parameters of modified SINDy are chosen as $q = 1$ and $\lambda = 0.1$ (for 30% noise, $\lambda = 0.05$). The library of candidate functions is constructed with polynomial terms up to third order (with constant term included, 84 candidates in total). Three different levels of noise are applied and the denoised signal is shown in figure 7 (for ease of visualization, only the first three states are shown). Figure 7 also shows the

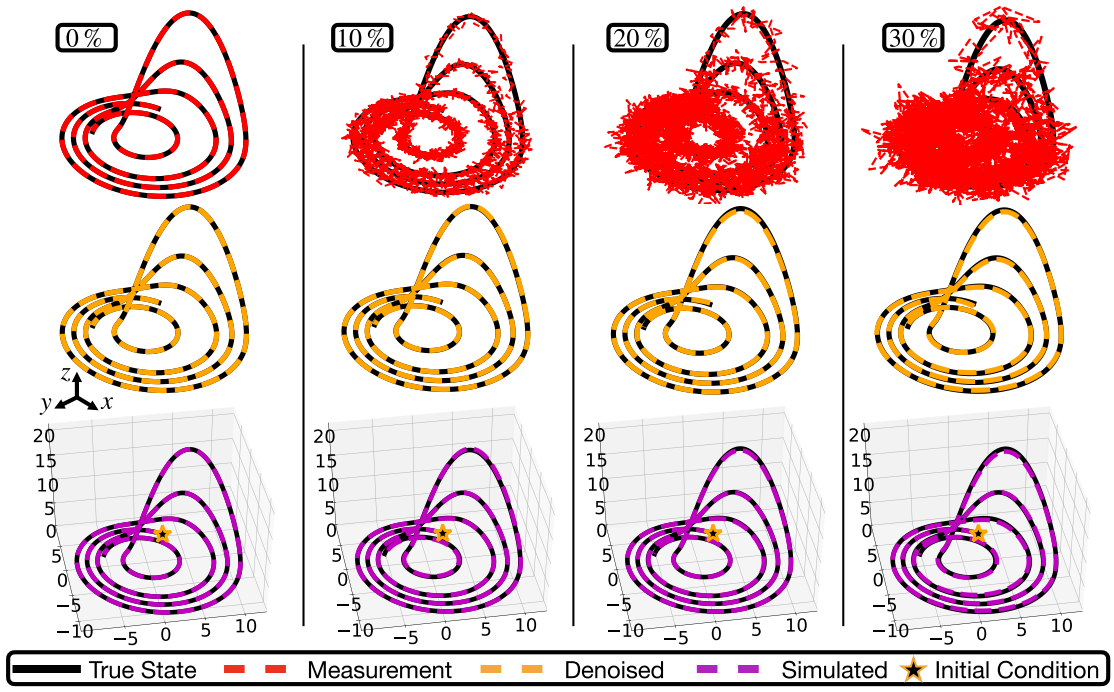

Figure 6. The modified SINDy is used to denoise the measurement of Rössler attractor while learning its model. The identified model shows high accuracy when simulating forward.

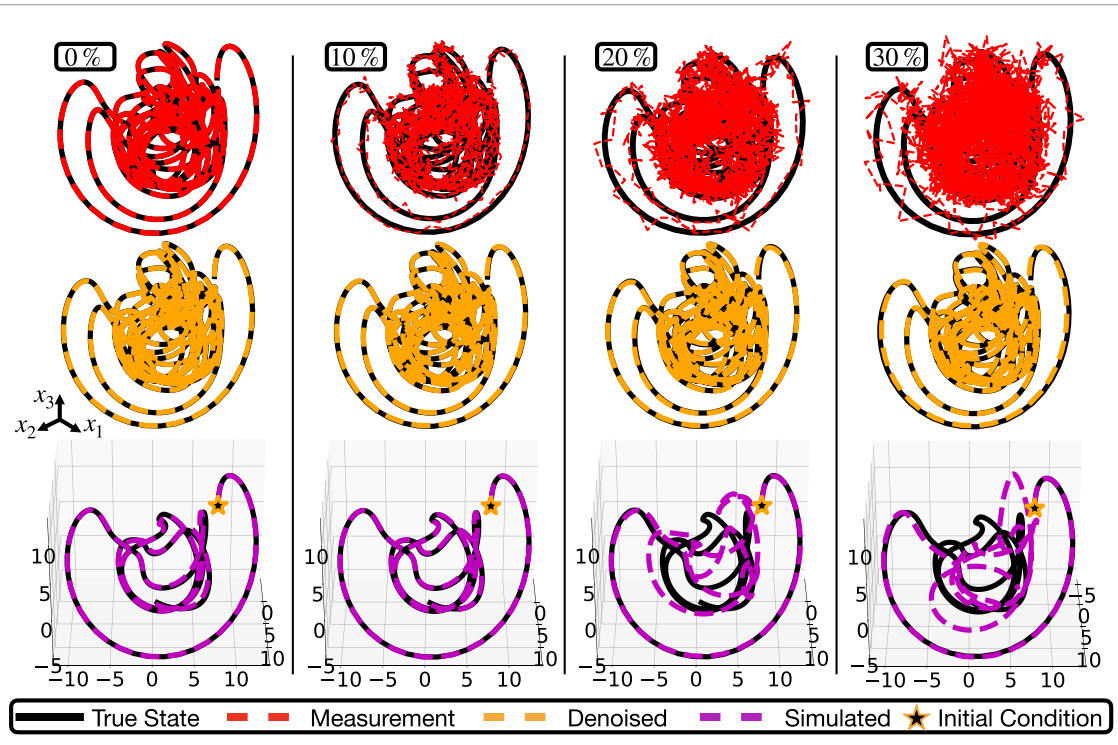

Figure 7. The modified SINDy is used to denoise the measurement of Lorenz 96 system while learning its model.

simulated trajectories of identified models. The initial condition $[1, 8, 8, 8, 8]$, $T = 5$, and $dt = 0.01$ are used to simulate the identified models.

In figure 8, the effectiveness of modified SINDy is demonstrated on a number of canonical dynamical systems models. For all examples, Gaussian noise with zero-mean is added to generate the noisy training data, and Adam optimizer is used to perform the optimization. The models and other parameters used for each example are summarized in appendix F. The modified SINDy correctly identified all the system model and noise distribution regardless of the noise magnitude used.

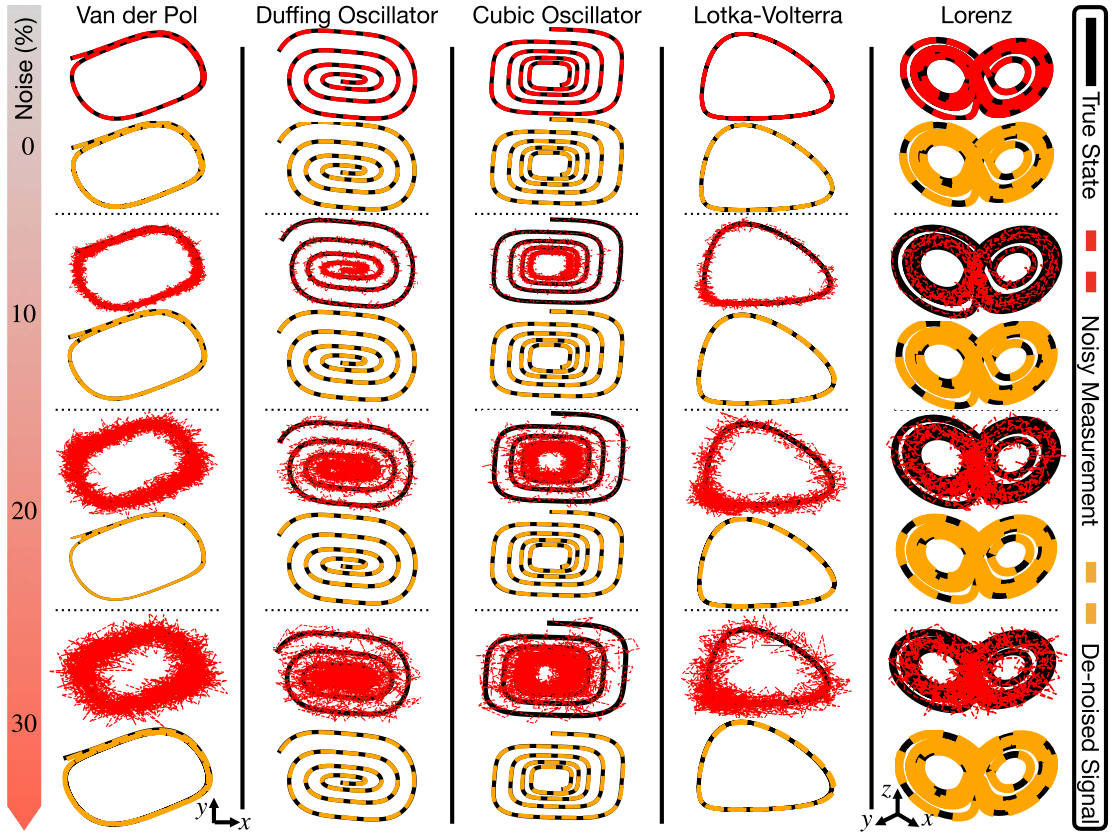

Figure 8. This plot shows the denoising ability of modified SINDy with different examples. Regardless of the noise percentage, modified SINDy correctly identified all the system models.

4.4. Identification of noise distributions

The modified SINDy algorithm has the ability to to handle different kinds of noise distributions. Three different kinds of noise distributions are used to demonstrate this: Gaussian, Uniform, and Gamma. To generate the Gamma noise, its shape and scale are set to 1. The generated noise is multiplied by Noise Percentage $\times$ var(Signal). The noise-free data of Van der Pol oscillator is generated the same way in section 4.1. The prediction step is set to $q = 2$ and the sparsity parameter is set to $\lambda = 0.15$. Figure 9 shows the distribution identified by modified SINDy. Figure 9 shows that learning the non-zero mean noise distribution is more difficult than learning a zero-mean one. For better learning results of a non-zero mean noise distribution, one can try the iterative learning approach shown in appendix H. Once the noise is separated from the signal, an additional step can be taken to identify the distribution of noise from the candidate distributions. This can be achieved by the `fitter` package in Python [93]. Appendix I shows more details of this process.

4.5. Discrepancy modeling

Modified SINDy can be easily integrated with the discrepancy modeling framework of SINDy [48]. This is of great practical value since it is often the case that parts of the dynamics is known. Suppose the known (theoretical) right-hand side dynamics in equation (1) is $\boldsymbol{g}(\mathbf{x})$. Discrepancy modeling assumes that the known model is not capable of modeling the data due to missing physics terms on the right-hand side. Thus there is a mismatch between the derivative $\dot{\mathbf{x}}$ and the known dynamics $\boldsymbol{g}(\mathbf{x})$. The discrepancy modeling approach tries to identify the missing dynamics $\boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\Xi}$ such that:

$$\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}) = \boldsymbol{g}(\mathbf{x}) + \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\Xi}. \tag{23}$$

To illustrate this process, consider a system $\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x})$, whose model is given as:

$$\begin{aligned} \dot{x} &= -10x + 10y + xy, \\ \dot{y} &= 28x - xz - y + 3z, \\ \dot{z} &= xy - 8/3z. \end{aligned} \tag{24}$$

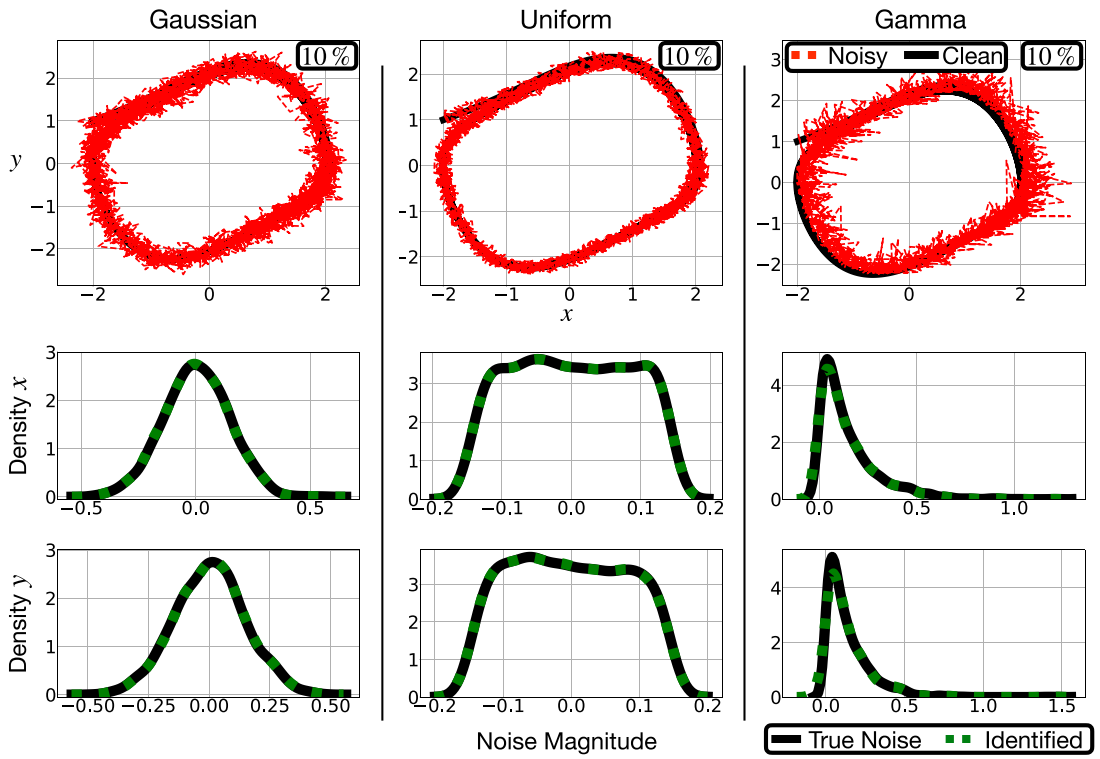

Figure 9. The ability of modified SINDy to handle different kinds of noise distribution is illustrated in this figure, and the Van der Pol oscillator is used as an example. 10% of noise is generated and added to the clean signal. As this figure shows, modified SINDy can identify different types of noise distribution correctly.

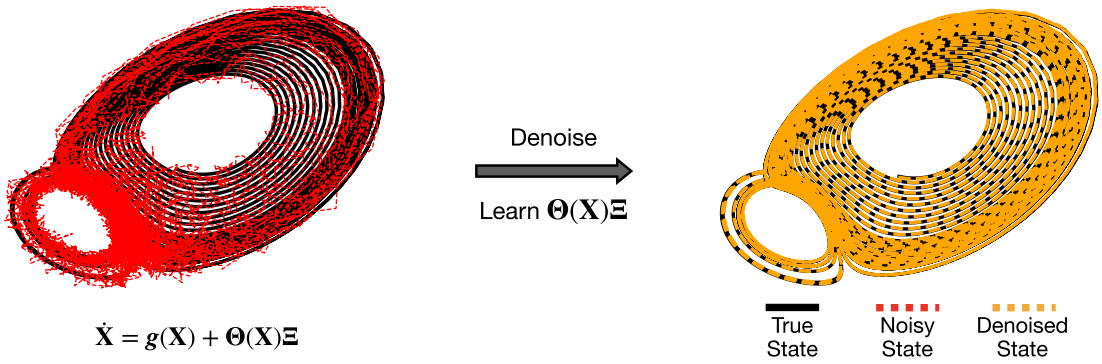

Figure 10. The modified SINDy can be easily integrated with the discrepancy learning method. The known structure $\boldsymbol{g}(\mathbf{X})$ can be utilized while we aim to learn the discrepancy and denoising the measurement.

Equation (24) is simulated with the $x_0 = [5, 5, 25]$, $T = 30$, and $dt = 0.005$ to generate noise-free data. Training data is produced by adding 10% Gaussian noise in order to create the noisy measurement. Assume that the noisy measurement of equation (24) is given. Further assume that the dynamics is modified based on $\boldsymbol{g}(\mathbf{x})$, which is given by:

$$\begin{aligned} \dot{x} &= -9.5x + 10.5y, \\ \dot{y} &= 27.6x - 1.1xz - 0.9y, \\ \dot{z} &= 1.05xy - 2.6z. \end{aligned} \tag{25}$$

The difference between the equations (24) and (18) will be the discrepancy model $\boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\Xi}$ that modified SINDy identifies. Note that this prior information of the dynamics, $\boldsymbol{g}(\mathbf{x})$, can be constrained to exist in the modified SINDy library during the optimization process, and its parameters can be used as an initial guess of the true parameters. Thus, the only thing we have to learn is the missing dynamics. Figure 10 illustrates this process. In this example, the $q = 4$, $\lambda = 0.4$, and the learning rate of Adam optimizer is 0.001. Figure 10

suggests that modified SINDy can be used to learn the discrepancy model when parts of the dynamics are already known.

5. Conclusion and future work

In this work, we introduce a new learning algorithm that leverages automatic differentiation and sparse regression for simultaneously: (1) denoising time-series data, (2) learning and parametrizing the noise probability distribution, and (3) identifying the underlying parsimonious dynamical system responsible for generating the time-series data. The method provides a critically enabling modification to the SINDy algorithm for improving robustness to noise with less training data in comparison with the previously developed NN denoising approach by Rudy *et al* [32]. Multiple numerical examples are shown to demonstrate the effectiveness of the modified SINDy method for signal and noise separation as well as model identification. It is also shown that the modified SINDy can be integrated with a discrepancy modeling framework whereby prior information of the dynamical model can be used to help identify the missing dynamics. Importantly, we have shown that modified SINDy can be used to learn various types of noise distributions, including Gaussian, uniform, and non-zero mean noise distributions, such as a Gamma distribution. Overall, the modified SINDy is a robust method with practical potential for handling highly noisy data sets and/or when partial model information is known.

The modified SINDy is modular, allowing for many easily integrated improvements. An important direction for development includes the incorporation of control inputs, since many systems of practical interest are actuated, such as the pendulum on a cart system [10, 55]. Extending modified SINDy to consider the impact of control will significantly expand its application domain. Moreover, it is also desirable to incorporate the Weak formulation into the current framework. As shown in [59, 71, 72, 86], the parameter error can be improved when a compact smooth support function is included into equation (8). In current framework, the choice of support function in equation (8) is $\omega = 1$, which is suboptimal. Thus, including the Weak formulation into current framework can have great potential to improve the parameter error. Improvements in computational speed are also desirable. In comparison with the sequential least-square thresholding of the standard SINDy algorithm, the Adam optimizer is slow. There is the potential to use the standard SINDy sparse regression algorithms to warm start the Adam optimization routine. There also exist possibility to use alternative packages like JAX MD [89, 90] to perform automatic differentiation, and it would be interesting to see the speed improvement it can achieve compared to Tensorflow. The modified SINDy can also be integrated with SINDy-PI to identify rational or implicit dynamics, which is quite difficult since the simulation error shown in equation (11) can not be calculated easily when the dynamics take a rational form. In order to denoise the signal generated from rational system, the implicit ODE solver needs to be used in order to simulate the implicit dynamics forward and backward in time. Currently, we do not have an elegant way to implement the implicit ODE solver in Tensorflow, and our future research includes extension of modified SINDy to identify rational dynamics. This is the case where the use of the NN denoising approach [32] by Rudy *et al* is ideal.

Finally, it is important to improve the robustness of the modified SINDy algorithm when a large number of library terms are used. Currently, the modified SINDy can not handle large libraries robustly due to the non-convexity of the optimization problem. When the library is too large, the problem becomes unstable without decreasing the optimizer's learning rate. One potential solution is to simulate the dynamics with a variable time step numerical simulation scheme instead of a fixed step scheme, as we used in this paper. Although there are still many improvements to be made, we believe the introduction of modified SINDy will help guide the use of automatic differentiation tools to improve the SINDy framework.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/dynamicslab/modified-SINDy and https://doi.org/10.5281/zenodo.4060354 [94].

Acknowledgments

S L B acknowledges support from the Army Research Office (ARO W911NF-19-1-0045) and the Air Force Office of Scientific Research (AFOSR FA9550-18-1-0200). J N K acknowledges support from the Air Force Office of Scientific Research (AFOSR FA9550-17-1-0329). We also acknowledge valuable discussions with Samuel H Rudy, Jared Callaham, Henning Lange, Daniel A Messenger, and Benjamin Herrmann.

Appendix A. Algorithm for simultaneously denoising and learning system model

Algorithm 1. Modified SINDy.

Input: $\mathbf{Y}$, $\boldsymbol{\Theta}(*)$, dt, λ, N_{loop}, ω
Output: $\boldsymbol{\Xi}$, $\hat{\mathbf{N}}$
/* Initialize the value of $\hat{\mathbf{N}}$ */
if SoftStart **then**
  $\hat{\mathbf{N}} = \mathbf{Y} - \text{smoothSignal}(\mathbf{Y})$ // If the soft start is true, the estimated value of noise is obtained by pre-smoothing the noisy signal.
else
  $\hat{\mathbf{N}} = \text{zeros}(\text{size}(\mathbf{Y}))$ // Else, the estimated value of noise is initialized using zero matrix.
/* Initialize the value of $\boldsymbol{\Xi}$ */
$\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}}$.
Calculate $\dot{\hat{\mathbf{X}}}$ using $\hat{\mathbf{X}}$.
$\boldsymbol{\Xi} = \text{SINDy}(\dot{\hat{\mathbf{X}}}, \boldsymbol{\Theta}(\hat{\mathbf{X}}), \lambda)$.
/* Simultaneously denoising and learning system model */
while $k < N_{loop}$ **do**
  Optimize $\mathcal{L}(\boldsymbol{\Xi}, \hat{\mathbf{N}})$ shown in equation (13).
  $(|\boldsymbol{\Xi}| < \lambda) = \mathbf{0}$. // Constrain the elements in $\boldsymbol{\Xi}$ whose absolute value smaller than λ as zero during the rest of optimization.
  $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}}$. // Get new estimate of true state.
  Calculate $\dot{\hat{\mathbf{X}}}$ using $\hat{\mathbf{X}}$. // Get new estimate of true derivative.
  $(|\boldsymbol{\Xi}| \neq \mathbf{0}) = \boldsymbol{\Theta}(\hat{\mathbf{X}}) \backslash \dot{\hat{\mathbf{X}}}$. // Regress the dynamics on terms in $\boldsymbol{\Xi}$ that are not constrained as zero.

Appendix B. Effect of thresholding parameter λ

Thresholding parameter λ is the most important parameter to tune in modified SINDy. The parameter λ will determine the sparsity of the model structure. Its effect can be seen in figure 11. In figure 11, Lorenz equation is simulated with $[-5.0, 5.0, 25.0]$, $dt = 0.01$, and $T = 25$. 10% of Gaussian noise is added and Adam optimizer with learning rate of 0.001 is used to denoise the signal. N_{loop} is set to 8 and different values of λ is used. For each λ, the numerical experiments is performed 10 times to calculate the median and distribution of the error as shown in figure 11. Figure 11 suggests that the value of λ must be properly tuned. If the value of λ is too small, the sparsity constraint will not be strong enough to enforce the correct model to be found. Moreover, $\boldsymbol{\Xi}$ and $\hat{\mathbf{N}}$ will easily get stuck in the local minimum. If the value of λ is too large, the correct terms can be wrongly eliminated and the resulting model structure will be wrong. If the model structure is wrong, there will be huge difference between the identified noise $\hat{\mathbf{N}}$ and true noise $\mathbf{N}$. To avoid swiping different values of λ, our proposed method can be easily modified to use the stepwise sparse regression (SSR) approach [62]. However, the use of SSR approach and its performance is not in the scope of this paper.

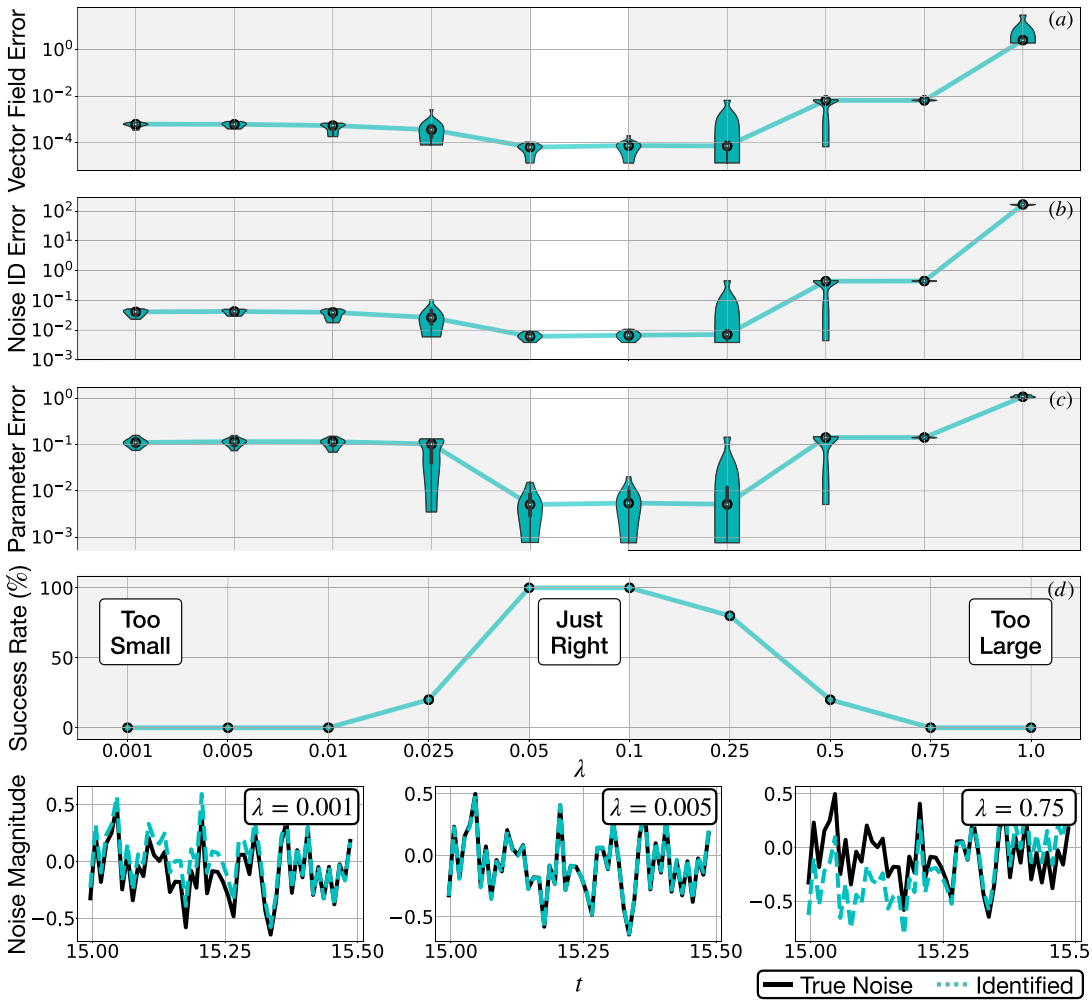

Figure 11. This figure shows how the choice of sparsity parameter λ will effect modified SINDy performance. If λ is too small, modified SINDy will not converge to the correct model in a short range of time and will be stuck in the local minimum. On the contrary, if the sparsity parameter is too large, the identified model will miss the necessary term to build the correct model. Thus, the value of λ needs to be tuned properly to determine the accurate model.

Appendix C. Effect of prediction step q

Figure 12 shows the effect of the prediction step q on the performance of NN denoising approach by Rudy *et al* [32] and modified SINDy approach. The chaotic Lorenz system is used for comparison. The Lorenz attractor is simulated by setting $x_0 = [-5, 5, 25]$, $T = 25$, and $dt = 0.01$. The noise level is set to 10% to generate noisy data. Each prediction step is run for 10 times to calculate the median of the error. Adam optimizer, with a learning rate of 0.001 is used to perform the optimization. N_{loop} is set to 3. Figure 12 suggests that the performance of modified SINDy is not hugely affected by the prediction step q. However, for the NN denoising approach shown in [32], there exist some value of q to achieve optimal performance. Figure 12 also suggests the computational time of both approaches increase linearly as the value of q increase. Thus, q can be chosen as a small value to save the computational time when using modified SINDy without sacrificing too much of the performance.

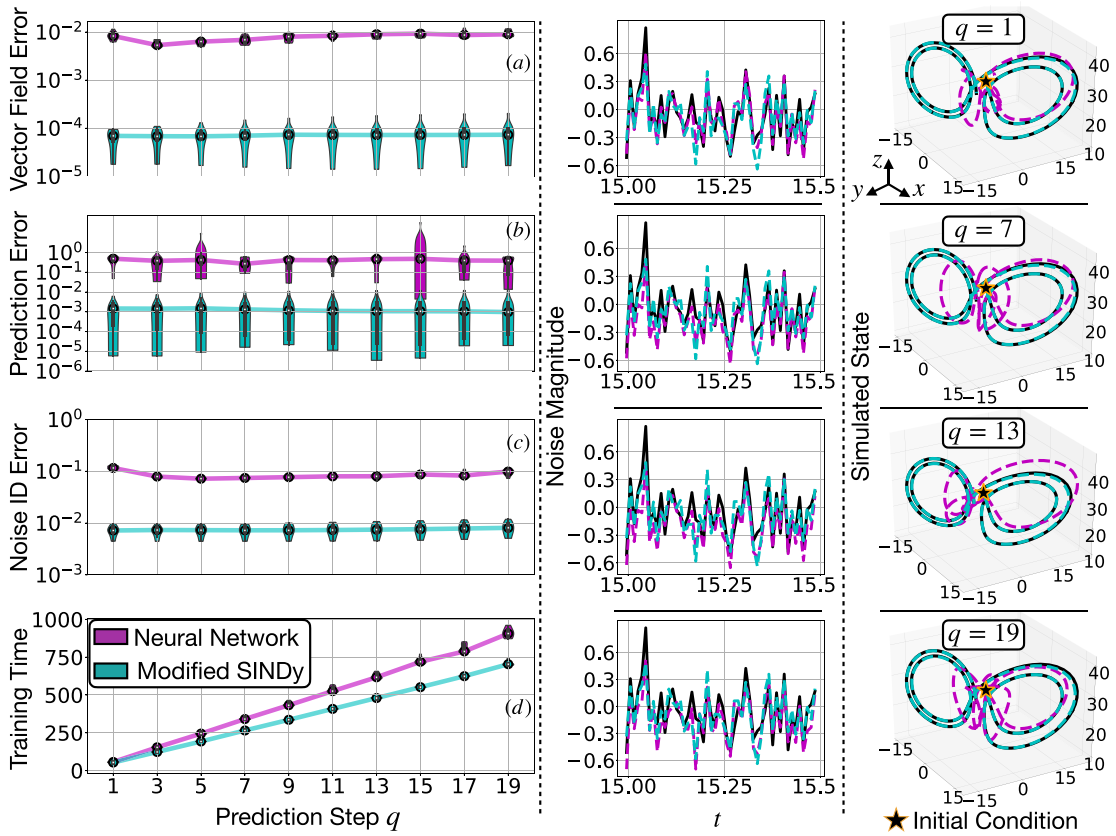

Figure 12. Left: the value of noise identification error, prediction error, and vector field error of NN denoising approach by Rudy *et al* [32] and modified SINDy (labeled as SINDy) are shown as the value of q changes. In (a)–(d), the black circle shows the median of the error of 10 runs while the violin shape represents the distribution of calculated result. Mid: The average of true noise and identified noise of modified SINDy and NN approach is shown for four different prediction steps. Right: The comparison of the simulated and true trajectory of modified SINDy and NN identified model is shown. The simulated trajectory uses $x_0 = [-5, 5, 25]$ and $dt = 0.01$. The model is simulated for 3 seconds. It could be seen that modified SINDy has better performance in this case. All the computation is performed on RTX 2080 GPU, with 32GBs of RAM and AMD Ryzen 7 2700X Processor.

Appendix D. Effect of optimization iteration N_{loop}

The parameter N_{loop} determines how many times the thresholding optimization is performed. Figure 13 shows the effect of N_{loop} on the noise identification error and vector field error using Lorenz attractor as an example. The system is simulated by setting $x_0 = [5, 5, 25]$, $T = 25$, $dt = 0.01$, and $q = 3$. Adam optimizer, with a learning rate of 0.001, is used to optimize the problem. Figure 13 suggests the performance of modified SINDy will gradually converge in the end.

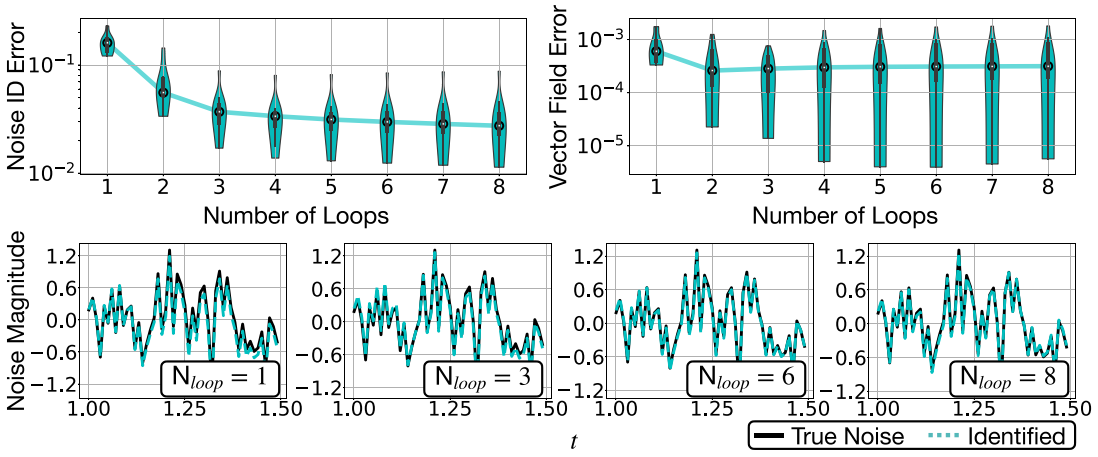

Figure 13. Top left: As the number of optimization loop increases, the noise identification error asymptotically decreases and converges to the optimal value. The black circle indicates the median of 10 runs while the violin shape represents error distribution. Top right: As the number of loop increases, the vector field error gradually converges to a certain value. Bottom: The average of the identified noise and true noise is shown for four different choice of optimization loops. As the number of loop increases, the differences between the true noise and identified noise is minimized.

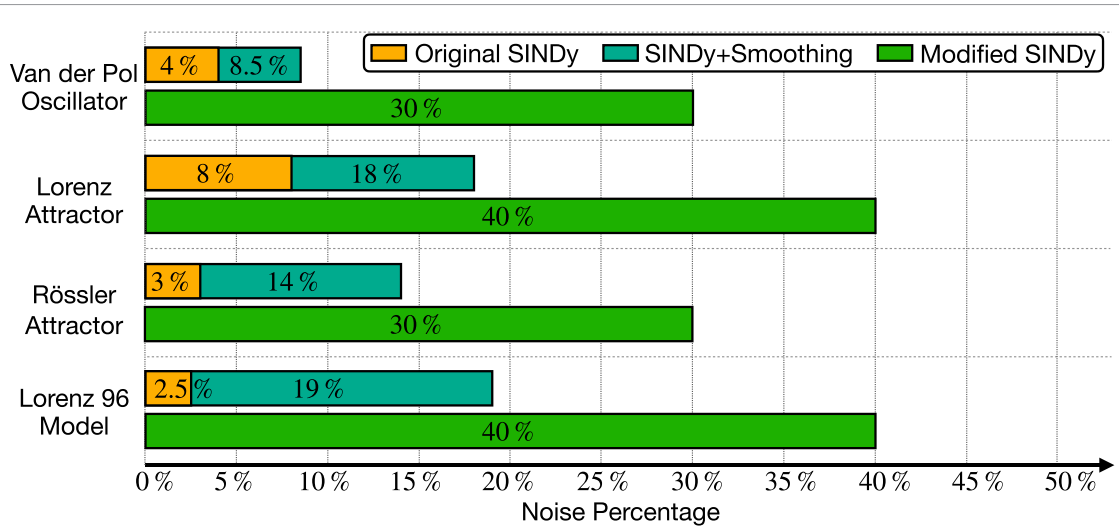

Figure 14. The maximum level of noise SINDy and modified SINDy can handle to generate the correct model structure is shown. Tikhonov regularization approach is used to pre-smooth the noisy data. It can be seen that the modified SINDy is about 2 times more robust than original SINDy [7].

Appendix E. Noise robustness comparison with SINDy

This section shows the noise robustness comparison of SINDy [7] and modified SINDy using Van der Pol oscillator, Lorenz attractor, and Rössler attractor. Figure 14 shows the maximum noise percentage each algorithm can handle to generate the correct model structure. For each noise level, 5 different noisy data sets are generated and provided to both approaches. If the tested algorithm fails to identify the correct model structure for any noisy data sets at a given noise level, we will assume it is not robust to noise at this level. For SINDy, the derivative is computed using finite difference, and we show the effect of pre-smoothing the noisy data on its performance. Note that no smoothing is applied for modified SINDy. The clean data for Lorenz attractor, Van der Pol oscillator, and Rössler attractor is generated the same way shown in sections 3.2, 4.1, and 4.2. For SINDy, the sparsity parameter λ is chosen as a hundred uniformly distributed values from 0.01 to the minimum of true parameters' absolute value. For modified SINDy, $q = 1$ and $N_{loop} = 8$ are used for all examples shown in figure 14. Table 1 shows other parameters we used for modified SINDy. Note that it is possible to make modified SINDy work at a higher noise level by tuning its parameters. However, swiping various parameters is quite computationally heavy for modified SINDy. Thus, the maximum noise level modified SINDy can tolerate in figure 14 is a lower approximate.

Table 1. Parameters used for modified SINDy in figure 14 under maximum noise it can tolerate. The constant term is included when building the library for Rössler attractor and Lorenz 96 model but not for other examples. The parameter error is calculated using equation (17).

Model	Noise percentage	Library order	Random seed	0	1	2	3	4
Lorenz	30%	2	λ	0.3	0.2	0.3	0.1	0.1
			Parameter Error	0.0046	0.051	0.031	0.032	0.077
			Max Adam Iteration	10000	15000	15000	15000	15000
Rössler	40%	2	λ	0.1	0.22	0.22	0.1	0.1
			Parameter Error	0.021	0.059	0.022	0.0062	0.020
			Max Adam Iteration	15000	15000	15000	15000	15000
Van der Pol	30%	3	λ	0.1	0.22	0.22	0.1	0.1
			Parameter Error	0.053	0.039	0.014	0.011	0.041
			Max Adam Iteration	15000	15000	15000	15000	5000
Lorenz 96	40%	3	λ	0.1	0.15	0.09	0.215	0.1
			Parameter Error	0.015	0.015	0.016	0.034	0.0075
			Max Adam Iteration	7000	7000	10000	10000	5000

Appendix F. Noise robustness comparison with Weak-SINDy

This section shows the noise robustness comparison of Weak-SINDy [72] and modified SINDy using Lorenz attractor as an example. The Lorenz attractor is simulated by setting $x_0 = [5, 5, 25]$, $T = 25$, $dt = 0.01$ and $dt = 0.001$. For both approaches, the library is constructed using up to second order terms (without constant term). Different percentage of noise is added to the clean data to generate noisy training data. The parameter error and success rate is computed for both approaches. For modified SINDy, Adam optimizer with learning rate of 0.001 is used to perform the optimization. The sparsity parameter is chosen as $\lambda = 0.1$ for most of the time. If the modified SINDy can not produce the correct result, $\lambda = 0.15$ is used instead. When $dt = 0.01$ we pre-smooth the data using approach mentioned in section 3.2 and no pre-smoothing is done when $dt = 0.001$. For Weak-SINDy, when $dt = 0.01$, 200 test functions with polynomial order of 14 are used. The width-at-half-max parameter $r_{whm} = 8$, and the support size $s = 31$. When $dt = 0.001$, 1000 test functions with polynomial order of 2 are used. The $r_{whm} = 16$, and $s = 30$. 30 different sparsity parameters evenly ranges from 0 to 0.95 are used, each generates a different candidate model for Weak-SINDy. The final model we used to calculate the parameter error for Weak-SINDy is the model that has correct structure (with only correct terms are selected from the library). If the Weak-SINDy fails to produce the model with correct active terms, the model that predicts the test data best is used to calculate the prediction error, and the test data is generated using initial condition $x_{0,test} = [-10, 10, 15]$ and simulated with $T = 25$ and $dt = 0.01$. The final comparison result of the best model generated by Weak-SINDy and modified SINDy can be seen in figure 15. Although figure 15 suggests the modified SINDy has better performance in high noise scenarios, it does have higher computational cost. When using a second order library and the above mentioned parameters, the Weak SINDy takes 0.073 second to compute the selection matrix $\hat{\Xi}$ for a given sparsity parameters λ. On the other hand, modified SINDy takes around 35 seconds to compute $\hat{\Xi}$ for a given sparsity parameters while $N_{loop} = 3$ and $q = 1$. Thus, Weak SINDy is two orders of magnitude faster than modified SINDy. Compared to backslash operation (Matlab's **mldivide**) used in Weak SINDy, the ADAM optimizer used in modified SINDy is slow. Our future work includes speeding up the modified SINDy calculation speed.

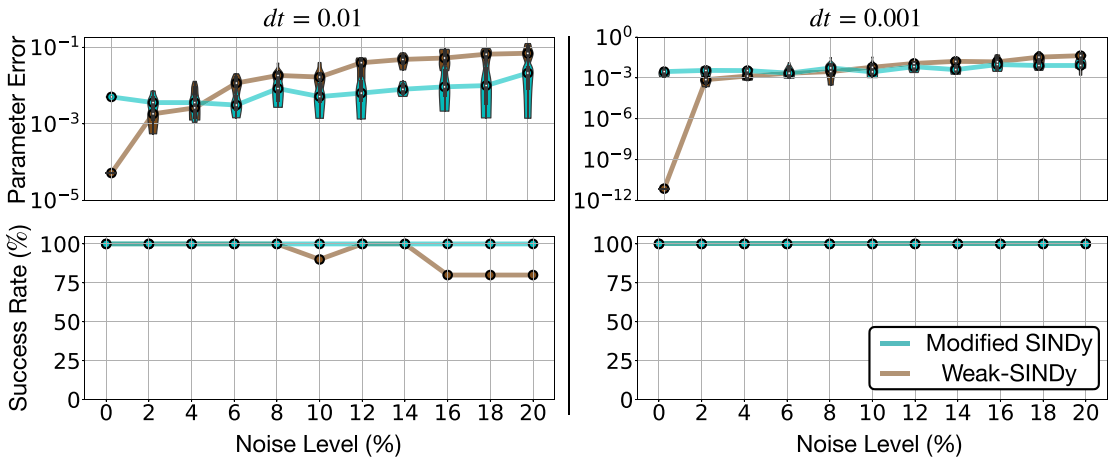

Figure 15. This figure shows noise robustness comparison of modified SINDy and Weak-SINDy using Lorenz attractor as an example. The effect of noise on the parameter error and successful identification rate is compared for both approaches. As it shows in the figure, Weak-SINDy and modified SINDy has almost the same accuracy when $dt = 0.001$. However, when $dt = 0.01$, the modified SINDy has a slightly higher success identification rate.

Table 2. Parameters used in figure 8.

Models	Initial Condition	Library Order	Learning Rate	T	q	λ
Van der Pol	$[-2, 1]$	3	0.001	10	1	0.05
Duffing	$[-2, -2]$	3	0.001	25	1	0.05
Cubic	$[0, 2]$	3	0.001	25	1	0.08
Lotka-Volterra	$[1, 2]$	3	0.001	10	1	0.2
Lorenz	$[5, 5, 25]$	2	0.001	25	3	0.1,0.15

Appendix G. Parameters used in figure 8

In this section, the models used to simulate the system in figure 8 are listed. The model used for simulating the Duffing oscillator is:

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -p_1 y - p_2 x - p_3 x^3, \end{aligned} \tag{26}$$

with $p_1 = 0.2$, $p_2 = 0.1$, and $p_3 = 1$. The model used for simulating the Cubic oscillator is:

$$\begin{aligned} \dot{x} &= p_1 x^3 + p_2 y^3, \\ \dot{y} &= p_3 x^3 + p_4 y^3, \end{aligned} \tag{27}$$

with $p_1 = -0.1$, $p_2 = 2$, $p_3 = -2$, and $p_4 = 0.1$. The model used for simulating the Lotka-Volterra system is:

$$\begin{aligned} \dot{x} &= p_1 x - p_2 xy, \\ \dot{y} &= p_2 xy - 2p_1 y, \end{aligned} \tag{28}$$

with $p_1 = 1$ and $p_2 = 0.5$. Other parameters used for training the modified SINDy is summarized in table 2. For all examples, $N_{loop} = 5$ and $dt = 0.01$.

Appendix H. Tips on learning non-zero mean noise

As section 4.4 suggests, learning non-zero mean noise distribution is much harder than learning the zero-mean noise distribution. To achieve better performance on the non-zero mean noise distribution, we propose an iterative learning approach. This approach can be summarized as follow: 1. Apply modified SINDy to the noisy data, briefly learn the distribution of noise. 2. Subtract the mean of learned noise from the noisy measurement, and use the new data to perform the learning. 3. Repeat the step 2 until the result converges and the correct model is found. The Van der Pol oscillator is used to illustrate this approach, and the clean data is generated the same way in section 4.1. 20% of Gamma noise is added to create the noisy data. The parameters are set as $q = 2$, and $\lambda = 0.15$. Figure 16 demonstrate this approach. However, there is

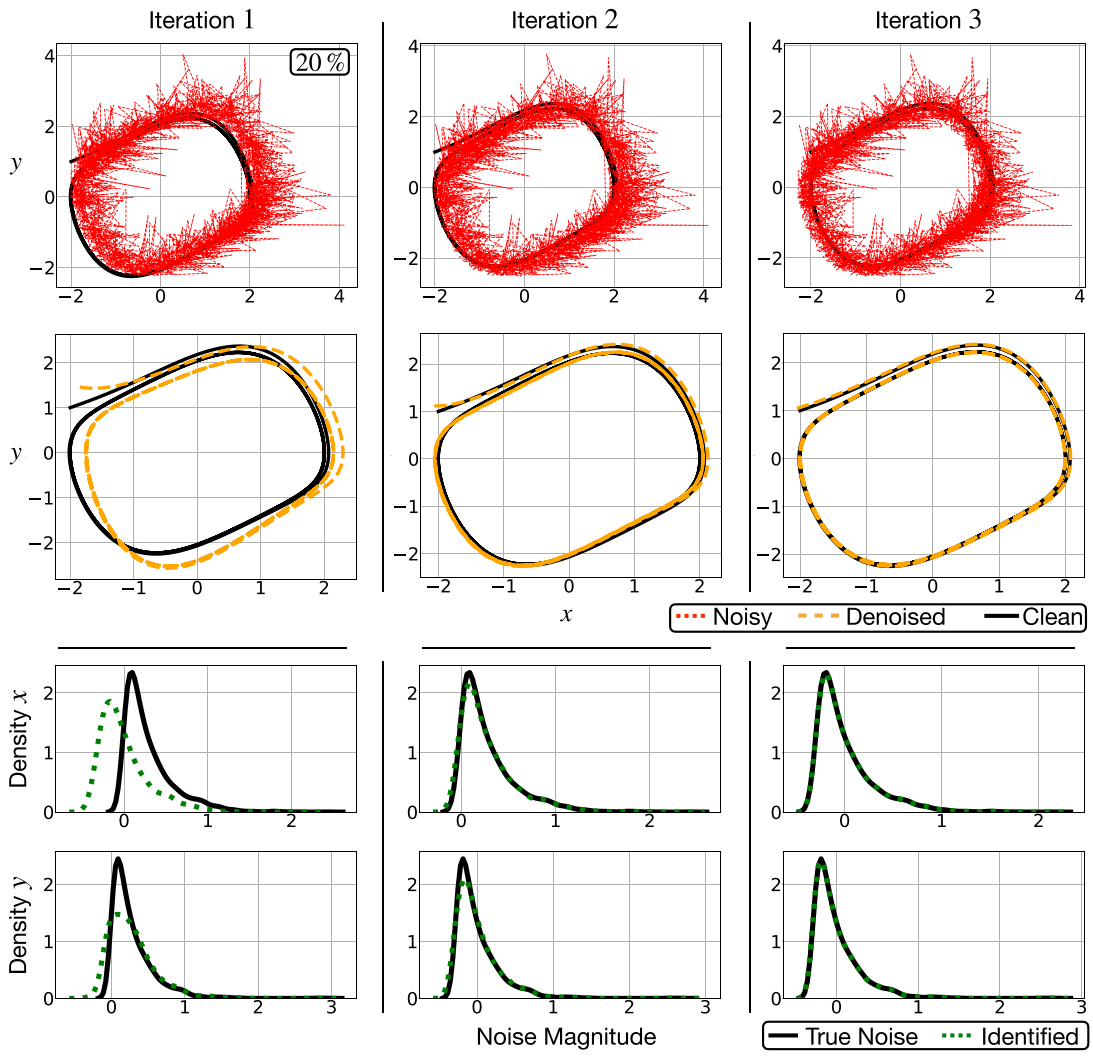

Figure 16. Iterative learning process is shown to tackle the non-zero mean noise distribution. By using this iterative process, modified SINDy can tackle noise distribution with non-zero means.

no guarantee that this approach will work when the bias of noise is too large, learning the non-zero mean noise is quite hard and careful tuning is needed. We find out using the soft start approach will also help the denoising of non-zero mean noise.

Appendix I. Identifying noise distribution type

When the noise is identified, it might be interesting to learn what type of distribution the noise follows. To illustrate this, the Van der Pol oscillator shown in equation (20) is simulated with initial condition $[-2, 1]$, $T = 50$, and $dt = 0.001$ (for Gamma and Rayleigh noise distribution, $dt = 0.01$). Next, 10% of noise is added to the simulation data to generate the noisy data. The noisy data is provided to modified SINDy to learn the dynamics and identify the noise added to the signal. We set $q = 2$, $\lambda = 0.15$ ($\lambda = 0.2$ for Gamma noise). Adam optimizer with learning rate equals to 0.001 is used, and the library order is set to 3. For all cases, the modified SINDy correctly identified the model. As table 3 shows, five different noise distributions is used to generate the noisy data. After the noise is identified, the distribution of noise is fitted into seven candidate noise distributions, which are normal distribution, uniform distribution, Gamma distribution, Dweibull distribution, Rayleigh distribution, Cauchy distribution, and Beta distribution. Next, the sum of the square errors between the $\hat{\mathbf{N}}$ and the fitted distribution is calculated, and the distribution that produces the lowest error is selected as the identified noise distribution. Notice that when there is not enough data provided, it is totally possible that other kinds of distribution is misidentified as the true underlying distribution of noise. The study of how many data points is needed to identify the correct distribution is beyond the scope of this paper. The final result can be summarized in table 3.

Table 3. The identified noise distribution versus the true distribution. For Gamma distribution, the parameter k represents shape and θ represents the scale.

True distribution	State	True parameter	Identified distribution	Identified Parameters
Gaussian	x	$\mu = 0, \sigma = 0.1413$	Gaussian	$\hat{\mu} = 0.003, \hat{\sigma} = 0.1451$
	y	$\mu = 0, \sigma = 0.1439$	Gaussian	$\hat{\mu} = 0.009, \hat{\sigma} = 0.1439$
Uniform	x	$\mu = 0, \sigma = 0.1413$	Uniform	$\hat{\mu} = -0.0717, \hat{\sigma} = 0.1438$
	y	$\mu = 0, \sigma = 0.1439$	Uniform	$\hat{\mu} = -0.0729, \hat{\sigma} = 0.1466$
Gamma	x	$k = 1, \text{loc} = 0, \theta = 0.1413$ $\mu = 0.1413, \sigma = 0.02$	Gamma	$k = 3.2714, \text{loc} = -0.095, \theta = 0.0722$ $\hat{\mu} = 0.1409, \hat{\sigma} = 0.0211$
	y	$k = 1, \text{loc} = 0, \theta = 0.1439$ $\mu = 0.1439, \sigma = 0.021$	Gamma	$k = 10.49, \text{loc} = -0.3105, \theta = 0.0432$ $\hat{\mu} = 0.1419, \hat{\sigma} = 0.0217$
Dweibull	x	$c = 2.07, \text{loc} = 0,$ $\text{scale} = 0.1413$	Dweibull	$\hat{c} = 2.064, \text{loc} = 0.8 \times 10^{-5},$ $\text{scale} = 0.1408$
	y	$c = 2.07, \text{loc} = 0,$ $\text{scale} = 0.1439$	Dweibull	$\hat{c} = 2.048, \text{loc} = -2.8 \times 10^{-5},$ $\text{scale} = 0.1438$
Rayleigh	x	$\mu = 0.1775, \sigma = 0.0085$	Rayleigh	$\hat{\mu} = 0.1775, \hat{\sigma} = 0.0085$
	y	$\mu = 0.1779, \sigma = 0.0086$	Rayleigh	$\hat{\mu} = 0.1779, \hat{\sigma} = 0.0086$

Appendix J. Caveats of the approach

This section provides some tips on using modified SINDy.

(a) Properly design the library: building the correct library for the regression is the most important part of this algorithm. If the library does not contain the terms included in the actual dynamics, the algorithm will fail to produce the correct noise and system model. Thus, whenever possible, one should include any prior information of the dynamics to build the library. In general, the library needs to be large enough to include all the terms that show up in the dynamics, and at the same time small enough to ensure the robustness. The best way to design the library is an open problem in the SINDy framework. When designing the library, one should use as much expert knowledge as possible. Usually, when some expert knowledge of the system is known, it can help user avoid unnecessary higher order terms. For example, when the signal is obtained from planetary system, then one might set the highest order of the library to 3, since cubic term shows up in the Newton's universal law of gravitation. When this kind of expert knowledge is known, usually physical law governing the dynamic, it can give us some idea on how to pick the highest polynomial term. When there's no expert knowledge available, we advise the reader start with simple second order library and see how the identified model performs. If the performance of the model is not ideal, then use one order higher library until the well performed model is identified. However, keep in mind that using a really high order library will make the optimization numerically sensitive, and at the same time increases computational burden, as figure 17 shows. Do not expect the modified SINDy will work on a library with hundreds or thousands of terms, it will break if the library is too large. For example, when using the Lorenz example with above 20% noise, the maximum order of the library modified SINDy can handle is 4 (about 34 terms). This happens since the higher order terms in the library will tend to mess up the forward and backward simulation and producing the `nan` cost, making the optimizer fails. To leverage this, one can try to decrease the learning rate of the optimizer, pre-smooth the data, get better initial estimate of $\boldsymbol{\Xi}$, reduce the library size, or set optimization parameters type as `float64`. Moreover, whether the constant term $\mathbf{1}$ should be included is case-specific. If the actual dynamics do not have a constant term and the measurement noise is non-zero mean or has significant outliers, including the constant basis in the library will make modified SINDy get stuck at the local minimum more easily. It is advised that the user tries both the library with and without constant basis.

(b) Initial guess of $\hat{\mathbf{N}}$ and $\boldsymbol{\Xi}$: having a good initial guess of the estimated noise $\hat{\mathbf{N}}$ and estimated selection parameter $\boldsymbol{\Xi}$ can improve the condition of the optimization problem and allowing us tackle harder problem with more library terms. If possible, the initial values of $\hat{\mathbf{N}}$ can be obtained by pre-smoothing the noisy signal, which will provide a good start for the optimization problem, and it is also good for estimating $\boldsymbol{\Xi}$. If no other information is given, the initial guess of the $\hat{\mathbf{N}}$ can be set as zeros.

(c) The choice of N_{train} and N_{loop}: the proper choice of parameter N_{train} and N_{loop} is also important for the successful identification of the system. The parameter N_{train} determines how many times the gradient descent step is applied. It also determines after how many gradient descent steps should the sparsity constraint be applied to $\boldsymbol{\Xi}$. For the examples used in this paper, a good choice is $N_{train} = 5000$, when the learning of ADAM is set to 0.001. As for N_{loop}, it should be a sufficiently large number that make sure the final answer converges. As seen in figure 13, after 8 iteration of loops, the noise identification error and

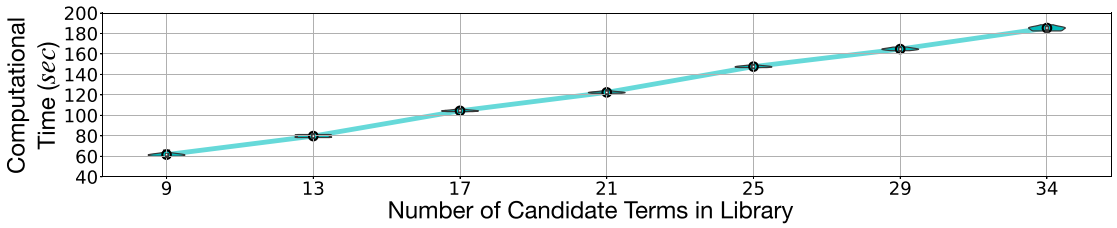

Figure 17. This figures shows the computational time needed to run the ADAM optimizer for 5000 iterations with different numbers of candidate functions. The Lorenz attractor is used as an example, with $T = 25$, $dt = 0.01$, and noise level set to 20%. This computation is performed 8 times for each number of candidate functions. When the number of candidate function is 9, it corresponds to second order library, and when the number of candidate function is 34, it corresponds to fourth order library. As it shows, the computational time increases almost linearly with the number of candidate terms in the library.

vector filed error converged. This indicates a good choice of N_{loop} is the one that drives the noise identification error or vector filed error below certain user define threshold or makes the error indicator converge. Moreover, when the noise level is high, N_{loop} should also be set higher. For all the examples used in this paper, $N_{loop} = 8$ is a good choice.

Appendix K. Global minimum of cost function equation (12)

The cost function shown in equation (12) has multiple global optimum solutions capable of achieving the optimum cost $L = 0$. Moreover, as the dimensionality of the free parameters $\hat{\mathbf{N}}$ and $\boldsymbol{\Xi}$ exceeds the dimensionality of the data $\mathbf{Y}$, the optimization problem is ill-posed. Here, we summarize several global optimal solutions of equation (12):

- The global optimum can show up if we can pick $\hat{\mathbf{N}} + \mathbf{C} = \mathbf{Y}$, which will result $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}} = \mathbf{C}$, where $\mathbf{C}$ is a constant. Thus, when picking $\boldsymbol{\Xi} = 0$, we can obtain a trivial solution that achieve the global optimum. In this case, the structure of the identified model is the sparsest. To completely avoid this solution, one can add additional penalty on the noise, such that it penalize $\hat{\mathbf{N}}$ to have large magnitude and avoid $\hat{\mathbf{N}} = \mathbf{Y}$. However, in practice we end up not adding this penalty term in our final optimization problem. We did this for several reasons:
  (a) It help us avoid adding additional tuning parameters which will make choosing the correct hyper parameter difficult.
  (b) Usually, the thresholding parameter λ is picked so that it is smaller than the maximum value of Ξ. Thus, it is unlikely to generate $\hat{\boldsymbol{\Xi}} = 0$.
  (c) Moreover, the initial guess of $\hat{\mathbf{N}}$ is picked as zero or other values that is close to true $\mathbf{N}$ (when using warm start). Thus, this initial guess of $\hat{\mathbf{N}}$ is quite small compared to the magnitude of $\mathbf{Y}$, which means sufficiently large iterations is needed to achieve $\hat{\mathbf{N}} = \mathbf{Y}$ when using ADAM optimizer. This indicates it is more natural for $\hat{\mathbf{N}}$ converge to $\mathbf{N}$ rather than $\mathbf{Y}$.

  We recreated this situation in figure 18. To generate this global optimum, we used Lorenz system with second order Polynomial library. The noisy data set is generated by using initial condition $x_0 = [5, 5, 25]$, $dt = 0.01$, $T = 25$, and 20% of Gaussian noise (using random seed 4). We picked our thresholding parameter $\lambda = 20$ so that we enforces $\hat{\boldsymbol{\Xi}} = 0$. Then we used ADAM optimizer with learning rate 0.001 to solve the optimization problem with $N_{loop} = 16$ and $N_{iter} = 5000$. As figure 18 shows, when enforcing $\hat{\boldsymbol{\Xi}} = 0$, we have $\hat{\mathbf{N}} = \mathbf{Y} + \mathbf{C}$. Note, to generate this result, we did not add penalty term on the magnitude of noise.
- The second global optimum is the true system we wish to identify. For example, when $\hat{\mathbf{N}} = \mathbf{N}$, we have $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}} = \mathbf{X}$. Thus, if we can pick the correct selection vector $\hat{\boldsymbol{\Xi}} = \boldsymbol{\Xi}$, we will achieve zero cost value. This solution is what we are looking for.
- The third case is more interesting, and we find out it tends to happen when the noise added to the system is extremely high (for example 40% to 100% of noise). In this case, our final solution will tend to converge to a 'sister system' that has similar behavior to the actual system.
  For example, suppose we are using a polynomial library, and we wish to identify the Lorenz system given its noisy simulation data with 50% Gaussian noise added. The noisy data set is generated by using initial condition $x_0 = [5, 5, 25]$, $dt = 0.01$, $T = 25$, and the Gaussian noise is generated using random seed 5. We picked our thresholding parameter $\lambda = 0.1$ and ADAM optimizer with learning rate 0.001 to solve the optimization problem with $N_{loop} = 8$ and $N_{iter} = 5000$. As figure 19 shows, then final denoised system converged to

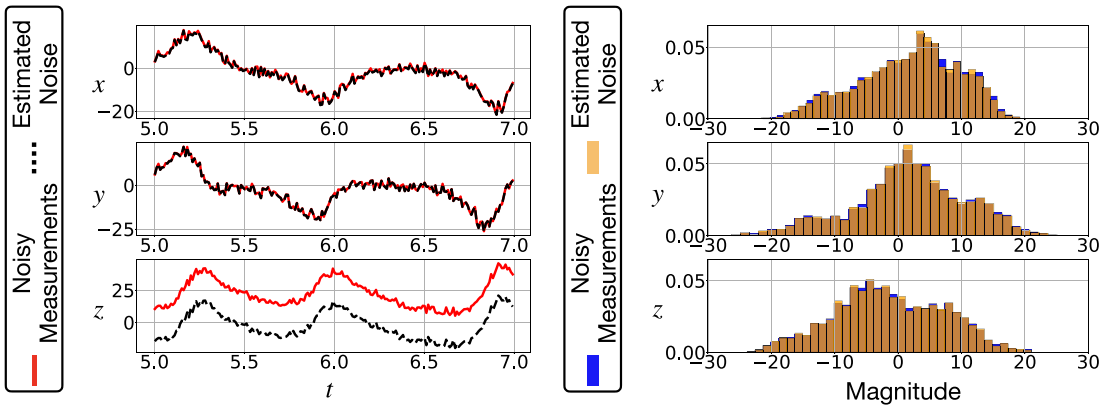

Figure 18. This figures illustrates one of the global minimum solution to equation (12). As we can see, the estimated noise $\hat{\mathbf{N}}$ differs from the noisy measurement $\mathbf{Y}$ by a constant value $\mathbf{C}$. This solution along with $\hat{\boldsymbol{\Xi}}$ forms one of the global optimum of cost function equation (12). This solution is unlikely to happen as long as $\hat{\boldsymbol{\Xi}}$ is not zero or an additional penalty term on $\hat{\mathbf{N}}$ is added to the cost function equation (12).

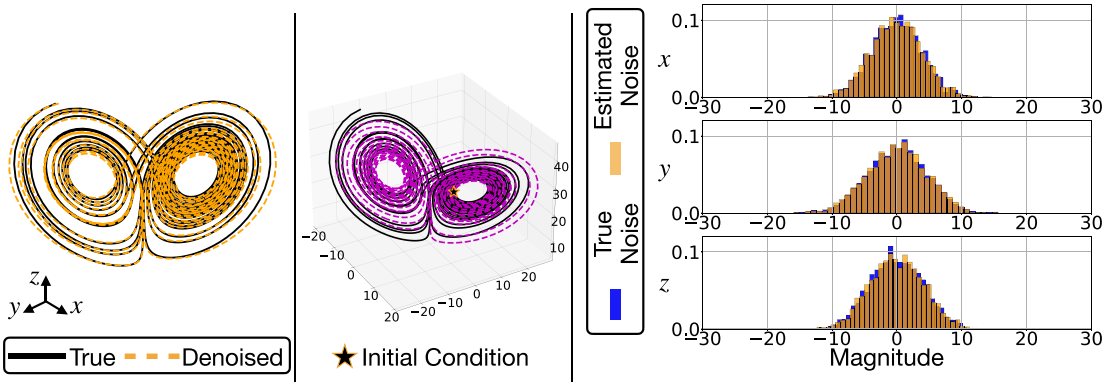

Figure 19. This figures illustrates the third type of global minimum solution to equation (12). Left: Even though the denoised signal does not perfectly match the underlying Lorenz system, its overall behavior resembles it. Middle: The simulation result of identified system shown in equation (29). Even though there's model discrepancy between the identified system and true system, the identified system still captures the shape of the Lorenz attractor. The identified system is simulated using initial condition $x_0 = [5, 5, 25]$, with $dt = 0.01$ and $T = 25$. Right: The distribution of identified noise and true noise. It can be seen that the identified noise has larger magnitude than the ground truth, which suggests adding a noise magnitude regularization term can help us converge to the true system.

a 'sister system' that has similar behavior to the original Lorenz system, with final cost value 0.001679. In this case, the identified system is:

$$\begin{aligned} \dot{x} &= -1.26x + 7.87y - 0.243xz, \\ \dot{y} &= 24.33x - 0.951xz + 0.787y, \\ \dot{z} &= 0.932xy - 2.58z, \end{aligned} \tag{29}$$

with $E_{\mathbf{N}} = 1.15$, $E_f = 0.0186$, and $E_{\mathbf{F}} = 0.436$. This example suggests that it is possible to find out a $\hat{\boldsymbol{\Xi}} \neq \boldsymbol{\Xi}$ and $\hat{\mathbf{N}} \neq \mathbf{N}$, such that the final cost value is still close to zero, which suggests the optimization problem in equation (12) without regularization term can theoretically have infinite global optimum. In other words, for any constant value of $\hat{\boldsymbol{\Xi}}$, there exist a unique $\hat{\mathbf{N}}$ that makes the cost function in equation (12) equals to zero.

In practice, we find out the third global optimum tends to show up when the noise-signal ratio is high (usually above 40%). Moreover, in this case, the identified noise tends to have a slightly larger magnitude than the true noise, as figure 19 illustrates. Thus, when dealing with highly corrupted measurement data, it is recommended the reader add a L_2 regularization term on the estimated noise magnitude to avoid the this type of global optimum. When the noise-signal ratio is low (under 30% for all the example we used in this paper), we find out this type of solution does not show up as long as the correct thresholding parameter is picked, we believe this is due to the initial guess of $\hat{\boldsymbol{\Xi}}$ and $\hat{\mathbf{N}}$ is good enough to avoid this type of solution.

Table 4. This table shows the denosing performance difference of using equation (12) and using equation (12) without derivative term e_d on Lorenz example with 40% of noise.

Terms	Cost function: equation (12)	Cost function: equation (12) without e_d
E_f	0.00259	0.0011
$E_{\mathbf{N}}$	0.271	0.1975
$E_{\mathbf{F}}$	0.003	0.00136
E_p	0.01259	0.0158
$\dot{x}$	$-9.7206x + 9.796y$	$-9.673x + 9.759y$
$\dot{y}$	$27.828x - 1.032y - 0.988xz$	$28.275x - 1.089y - 1.003xz$
$\dot{z}$	$-2.729z + 1.059xy$	$-2.678z + 1.033xy$

To summarize, the optimization problem shown in equation (12) has multiple global optimum solution. When the noise-signal ratio is low and the initial guess of $\hat{\boldsymbol{\Xi}}$ and $\hat{\mathbf{N}}$ is good enough and proper thresholding parameter λ is picked, the final solution will converge to the true dynamics and noise. Moreover, it is recommended to add noise regularization term when the added noise is too high. The effect of additional regularization term on the performance of approach, and the proper way to tune this regularization term is beyond the scope of this paper.

Appendix L. Calculation of the derivative term e_d

The derivative error shown in equations (6) and (12) are calculated using five-point stencil approach throughout the paper. Given an estimated state vector $\hat{\mathbf{X}} \in \mathbb{R}^{m\times n}$, the derivative is calculated as:

$$\dot{\hat{\mathbf{X}}} = \frac{-\hat{\mathbf{X}}[5:end,:] + 8\hat{\mathbf{X}}[4:end-1,:] - 8\hat{\mathbf{X}}[2:end-3,:] + \hat{\mathbf{X}}[1:end-4,:]}{12dt}, \tag{30}$$

where $[i:j,:]$ represent the matrix slicing. The resulting derivative estimation term will have shape $\dot{\hat{\mathbf{X}}} \in \mathbb{R}^{m-4\times n}$. Thus to calculate the derivative error e_d, we discard the first and last two rows of matrix $\hat{\mathbf{X}}$ so that the dimension of $\dot{\hat{\mathbf{X}}}$ match with $\boldsymbol{\Theta}(\hat{\mathbf{X}}[3:end-2,:])\boldsymbol{\Xi} \in \mathbb{R}^{m-4\times n}$.

It is also worth noting that the simulation error e_s already constraints the derivative error e_d. In other words, when minimizing the simulation error e_s, the derivative error is also minimized, since e_s utilize the right-hand side of system dynamics $\boldsymbol{\Theta}(\hat{\mathbf{X}})\boldsymbol{\Xi}$. Thus, we find out dropping the derivative error term e_d from the cost function equation (12) does not affect the overall performance of the approach. To illustrate this, the Lorenz attractor is simulated using $dt = 0.01, T = 25$ with initial condition $x_0 = [5,5,25]$. Then 40% of noise is added to the simulation data to generate noisy measurement. Next, ADAM optimizer is used with following parameters to optimize the cost function shown in equation (12). The parameters used is $q = 1$, $N_{train} = 5000$, $N_{loop} = 8$, learning rate is set to 0.001, and the order of library is 2. Next, the same parameters is used to minimize the cost function without the derivative error term. The final result yield by using two cost function can be seen in table 4. Note that $E_{\mathbf{F}}$ is calculate by simulating the identified system using initial condition $x_0 = [5,5,25]$, $dt = 0.01$ and $T = 3$. As table 4 suggests, dropping the e_d term from the cost function equation (12) does not change the final result too much. Moreover, the E_f, $E_{\mathbf{N}}$, and $E_{\mathbf{F}}$ error got slight improvement. We believe this happens since dropping the e_d term from the cost function avoids the numeric derivative approximation error.

ORCID iDs

Kadierdan Kaheman https://orcid.org/0000-0003-2279-2793
Steven L Brunton https://orcid.org/0000-0002-6565-5118
J Nathan Kutz https://orcid.org/0000-0002-6004-2275

References

[1] Nelles O 2013 *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models* (Berlin, NY: Springer)
[2] Ljung L 2010 Perspectives on system identification *Annu. Rev. Control* **34** 1–12
[3] Kutz J N, Brunton S L, Brunton B W and Proctor J L 2016 *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (Philadelphia, PA: SIAM)
[4] Akaike H 1969 Fitting autoregressive models for prediction *Ann. Inst. Stat. Math.* **21** 243–7
[5] Billings S A 2013 *Nonlinear System Identification: Narmax Methods in the Time, Frequency and Spatio-Temporal Domains* (New York: Wiley)

[6] Brunton S L and Kutz J N 2019 *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control* (Cambridge, MA: Cambridge University Press)
[7] Brunton S L, Proctor J L and Kutz J N 2016 Discovering governing equations from data by sparse identification of nonlinear dynamical systems *Proc. Natl Acad. Sci.* **113** 3932–7
[8] Rudy S H, Brunton S L, Proctor J L and Kutz J N 2017 Data-driven discovery of partial differential equations *Sci. Adv.* **3** e1602614
[9] Schaeffer H 2017 Learning partial differential equations via data discovery and sparse optimization *Proc. R. Soc.* A **473** 20160446
[10] Kaheman K, Kutz J N and Brunton S L 2020 Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics *Proc. R. Soc.* A **476** 20200279
[11] Schmid P J 2010 Dynamic mode decomposition of numerical and experimental data *J. Fluid Mech.* **656** 5–28
[12] Klus S, Nüske F, Koltai P, Wu H, Kevrekidis I, Schütte C and Noé F 2018 Data-driven model reduction and transfer operator approximation *J. Nonlinear Sci.* **28** 985–1010
[13] Yang L, Zhang D and Karniadakis G E 2018 Physics-informed generative adversarial networks for stochastic differential equations (arXiv:1811.02033)
[14] Wehmeyer C and Noé F 2018 Time-lagged autoencoders: deep learning of slow collective variables for molecular kinetics *J. Chem. Phys.* **148** 1–9
[15] Mardt A, Pasquali L, Wu H and Noé F 2018 VAMPnets: deep learning of molecular kinetics *Nat. Commun.* **9** 5
[16] Vlachas P R, Byeon W, Wan Z Y, Sapsis T P and Koumoutsakos P 2018 Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks *Proc. R. Soc.* A **474** 20170844
[17] Lu L, Meng X, Mao Z and Karniadakis G E 2019 DeepXDE: a deep learning library for solving differential equations (arXiv:1907.04502)
[18] Raissi M, Perdikaris P and Karniadakis G 2019 Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations *J. Comput. Phys.* **378** 686–707
[19] Champion K, Lusch B, Kutz J N and Brunton S L 2019 Data-driven discovery of coordinates and governing equations *Proc. Natl Acad. Sci.* **116** 22445–51
[20] Bongard J and Lipson H 2007 Automated reverse engineering of nonlinear dynamical systems *Proc. Natl Acad. Sci.* **104** 9943–8
[21] Schmidt M and Lipson H 2009 Distilling free-form natural laws from experimental data *Science* **324** 81–5
[22] Budišić M, Mohr R and Mezić I 2012 Applied Koopmanism *Chaos* **22** 047510
[23] Mezic I 2013 Analysis of fluid flows via spectral properties of the Koopman operator *Annu. Rev. Fluid Mech.* **45** 357–78
[24] Williams M O, Kevrekidis I G and Rowley C W 2015 A data-driven approximation of the Koopman operator: extending dynamic mode decomposition *J. Nonlinear Sci.* **6** 1307–46
[25] Pathak J, Hunt B, Girvan M, Lu Z and Ott E 2018 Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach *Phys. Rev. Lett.* **120** 024102
[26] Raissi M and Karniadakis G E 2017 Machine learning of linear differential equations using Gaussian processes (arXiv:1701.02440)
[27] Raissi M and Karniadakis G E 2018 Hidden physics models: machine learning of nonlinear partial differential equations *J. Comput. Phys.* **357** 125–41
[28] Giannakis D and Majda A J 2012 Nonlinear Laplacian spectral analysis for time series with intermittency and low-frequency variability *Proc. Natl Acad. Sci.* **109** 2222–7
[29] Yair O, Talmon R, Coifman R R and Kevrekidis I G 2017 Reconstruction of normal forms by learning informed observation geometries from data *Proc. Natl Acad. Sci.* **114** 201620045
[30] Daniels B C and Nemenman I 2015 Automated adaptive inference of phenomenological dynamical models *Nat. Commun.* **6** 8133
[31] Yao C and Bollt E M 2007 Modeling and nonlinear parameter estimation with Kronecker product representation for coupled oscillators and spatiotemporal systems *Physica* D **227** 78–99
[32] Rudy S H, Kutz J N and Brunton S L 2019 Deep learning of dynamics and signal-noise decomposition with time-stepping constraints *J. Comput. Phys.* **396** 483–506
[33] Baydin A G, Pearlmutter B A, Radul A A and Siskind J M 2017 Automatic differentiation in machine learning: a survey *J. Mach. Learn. Res.* **18** 5595–637
[34] Su W *et al* 2017 False discoveries occur early on the Lasso path *Ann. Stat.* **45** 2133–50
[35] Tibshirani R 1996 Regression shrinkage and selection via the Lasso *J. R. Statist. Soc.* B **58** 267–88
[36] Zhang L and Schaeffer H 2019 On the convergence of the SINDy algorithm *Multiscale Model. Simul.* **17** 948–72
[37] Champion K, Zheng P A, Brunton S and Nathan Kutz J 2019 A unified sparse optimization framework to learn parsimonious physics-informed models from data (arxiv:1906.10612v1)
[38] Zheng P, Askham T, Brunton S L, Kutz J N and Aravkin A Y 2018 A unified framework for sparse relaxed regularized regression: SR3 *IEEE Access* **7** 1404–23
[39] Sorokina M, Sygletos S and Turitsyn S 2016 Sparse identification for nonlinear optical communication systems: SINO method *Opt. Express* **24** 30433–43
[40] Loiseau J-C and Brunton S L 2018 Constrained sparse Galerkin regression *J. Fluid Mech.* **838** 42–67
[41] Dam M, Brøns M, Juul Rasmussen J, Naulin V and Hesthaven J S 2017 Sparse identification of a predator-prey system from simulation data of a convection model *Phys. Plasmas* **24** 022310
[42] Loiseau J-C, Noack B R and Brunton S L 2018 Sparse reduced-order modelling: sensor-based dynamics to full-state estimation *J. Fluid Mech.* **844** 459–90
[43] Hoffmann M, Fröhner C and Noé F 2019 Reactive SINDy: discovering governing reactions from concentration data *J. Chem. Phys.* **150** 025101
[44] Loiseau J-C 2020 Data-driven modeling of the chaotic thermal convection in an annular thermosyphon *Theor. Computat. Fluid Dyn.* **34** 1–27
[45] El Sayed M Y, Semaan R and Radespiel R 2018 Sparse modeling of the lift gains of a high-lift configuration with periodic coanda blowing *2018 AIAA Aerospace Sciences Meeting* p 1054
[46] Narasingam A and Kwon J S-I 2018 Data-driven identification of interpretable reduced-order models using sparse regression *Comput. Chem. Eng.* **119** 101–11
[47] de Silva B, Higdon D M, Brunton S L and Kutz J N 2019 Discovery of physics from data: universal laws and discrepancy models (arXiv:1906.07906)
[48] Kaheman K, Kaiser E, Strom B, Kutz J N and Brunton S L 2019 Learning discrepancy models from experimental data *Conf. Decision and Control*
[49] Thaler S, Paehler L and Adams N A 2019 Sparse identification of truncation errors *J. Comput. Phys.* **397** 108851

[50] Lai Z and Nagarajaiah S 2019 Sparse structural system identification method for nonlinear dynamic systems with hysteresis/inelastic behavior *Mech. Syst. Signal Process.* **117** 813–42
[51] Deng N, Noack B R, Morzyński M and Pastur L R 2020 Low-order model for successive bifurcations of the fluidic pinball *J. Fluid Mech.* **884** A37
[52] Schmelzer M, Dwight R P and Cinnella P 2020 Discovery of algebraic Reynolds-stress models using sparse symbolic regression *Flow Turbul. Combust.* **104** 579–603
[53] Pan S, Arnold-Medabalimi N and Duraisamy K 2020 Sparsity-promoting algorithms for the discovery of informative Koopman invariant subspaces (arXiv:2002.10637)
[54] Beetham S and Capecelatro J 2020 Formulating turbulence closures using sparse regression with embedded form invariance (arXiv:2003.12884)
[55] Brunton S L, Proctor J L and Kutz J N 2016 Sparse identification of nonlinear dynamics with control (SINDYc) *IFAC-PapersOnLine* **49** 710–15
[56] Kaiser E, Kutz J N and Brunton S L 2018 Sparse identification of nonlinear dynamics for model predictive control in the low-data limit *Proc. R. Soc.* A **474** 20180335
[57] Mangan N M, Brunton S L, Proctor J L and Kutz J N 2016 Inferring biological networks by sparse identification of nonlinear dynamics *IEEE Trans. Mol. Biol. Multi-Scale Commun.* **2** 52–63
[58] Zhang S and Lin G 2018 Robust data-driven discovery of governing physical laws with error bars *Proc. R. Soc.* A **474** 20180305
[59] Messenger D A and Bortz D M 2020 Weak SINDy for partial differential equations (arXiv:2007.02848)
[60] Rudy S, Alla A, Brunton S L and Kutz J N 2019 Data-driven identification of parametric partial differential equations *SIAM J. Appl. Dyn. Sys.* **18** 643–60
[61] Champion K P, Brunton S L and Kutz J N 2019 Discovery of nonlinear multiscale systems: sampling strategies and embeddings *SIAM J. Appl. Dyn. Syst.* **18** 312–33
[62] Boninsegna L, Nüske F and Clementi C 2018 Sparse learning of stochastic dynamical equations *J. Chem. Phys.* **148** 241723
[63] Mangan N M, Kutz J N, Brunton S L and Proctor J L 2017 Model selection for dynamical systems via sparse regression and information criteria *Proc. R. Soc.* A **473** 20170009
[64] Tran G and Ward R 2017 Exact recovery of chaotic systems from highly corrupted data *Multiscale Model. Simul.* **15** 1108–29
[65] Schaeffer H and McCalla S G 2017 Sparse model selection via integral terms *Phys. Rev.* E **96** 023302
[66] Schaeffer H, Tran G and Ward R 2018 Extracting sparse high-dimensional dynamics from limited data *SIAM J. Appl. Math.* **78** 3279–95
[67] Wu K and Xiu D 2019 Numerical aspects for approximating governing equations using data *J. Comput. Phys.* **384** 200–21
[68] Mangan N M, Askham T, Brunton S L, Kutz J N and Proctor J L 2019 Model selection for hybrid dynamical systems via sparse regression *Proc. R. Soc.* A **475** 20180534
[69] Gelß P, Klus S, Eisert J and Schütte C 2019 Multidimensional approximation of nonlinear dynamical systems *J. Comput. Nonlinear Dynam.* **14** 061006
[70] Goeßmann A, Götte M, Roth I, Sweke R, Kutyniok G and Eisert J 2020 Tensor network approaches for learning non-linear dynamical laws (arXiv:2002.12388)
[71] Reinbold P A, Gurevich D R and Grigoriev R O 2020 Using noisy or incomplete data to discover models of spatiotemporal dynamics *Phys. Rev.* E **101** 010203
[72] Messenger D A and Bortz D M 2020 Weak SINDy: Galerkin-based data-driven model selection (arXiv:2005.04339)
[73] de Silva B M, Champion K, Quade M, Loiseau J-C, Kutz J N and Brunton S L 2020 PySINDy: a Python package for the sparse identification of nonlinear dynamics from data (arXiv:2004.08424)
[74] van Breugel F, Kutz J N and Brunton B 2020 Numerical differentiation of noisy data: A unifying multi-objective optimization framework (arXiv:2009.01911)
[75] Kutz J N 2013 *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data* (Oxford: Oxford University Press)
[76] Abadi M *et al* 2016 Tensorflow: a system for large-scale machine learning *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* pp 265–83
[77] Rackauckas C and Nie Q 2017 Differential Equations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia *J. Open Res. Softw.* **5** 15
[78] Van Merriënboer B, Breuleux O, Bergeron A and Lamblin P 2018 Automatic differentiation in ML: where we are and where we should be going *Advances in Neural Information Processing Systems* pp 8757–67
[79] Chen R T, Rubanova Y, Bettencourt J and Duvenaud D K 2018 Neural ordinary differential equations *Advances in Neural Information Processing Systems* pp 6571–83
[80] Rudy S H, Brunton S L and Kutz J N 2019 Smoothing and parameter estimation by soft-adherence to governing equations *J. Comput. Phys.* **398** 108860
[81] Both G-J, Choudhury S, Sens P and Kusters R 2019 DeepMoD: Deep learning for model discovery in noisy data (arXiv:1904.09406)
[82] Rackauckas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, Skinner D and Ramadhan A 2020 Universal differential equations for scientific machine learning (arXiv:2001.04385)
[83] Lange H, Brunton S L and Kutz N 2020 From Fourier to Koopman: Spectral methods for long-term time series prediction (arXiv:2004.00574)
[84] Boninsegna L, Nüske F and Clementi C 2018 Sparse learning of stochastic dynamical equations *J. Chem. Phys.* **148** 241723
[85] Pan W, Yuan Y, Gonçalves J and Stan G 2016 A sparse Bayesian approach to the identification of nonlinear state-space systems *IEEE Trans. Autom. Control* **61** 182–7
[86] Gurevich D R, Reinbold P A K and Grigoriev R O 2019 Robust and optimal sparse regression for nonlinear pde models *Chaos* **29** 103113
[87] Zhang S and Lin G 2019 Robust data-driven discovery of governing physical laws using a new subsampling-based sparse Bayesian method to tackle four challenges (large noise, outliers, data integration, and extrapolation) (arXiv:1907.07788)
[88] Innes M 2018 Flux: elegant machine learning with Julia *J. Open Source Softw.* **3** 602
[89] Schoenholz S S, Cubuk E D and Jax M 2018 End-to-end differentiable, hardware accelerated, molecular dynamics in pure python (arXiv:1912.04232)
[90] Goodrich C P, King E M, Schoenholz S S, Cubuk E D and Brenner M P 2021 Designing self-assembling kinetics with differentiable statistical physics models *Proc. Natl Acad. Sci.* **118** e2024083118

[91] Kingma D P and Ba J 2014 Adam: A method for stochastic optimization (arXiv:1412.6980)
[92] Vulpiani A, Cecconi F and Cencini M 2009 *Chaos: From Simple Models to Complex Systems* vol 17 (Singapore: World Scientific)
[93] Cokelaer T, Brian, Stringari C E Broda E and Pruesse E 2020 cokelaer/fitter: v1.2.3 synchronised on pypi (available at: https://doi.org/10.5281/zenodo.3995055)
[94] Kaheman K, Brunton S L and Kutz J N 2020 Dynamicslab/modified-SINDy: V1.0.0 code for modified-SINDy algorithm *Zenodo Version V1.0.0* (available at: https://doi.org/10.5281/zenodo.4060354)