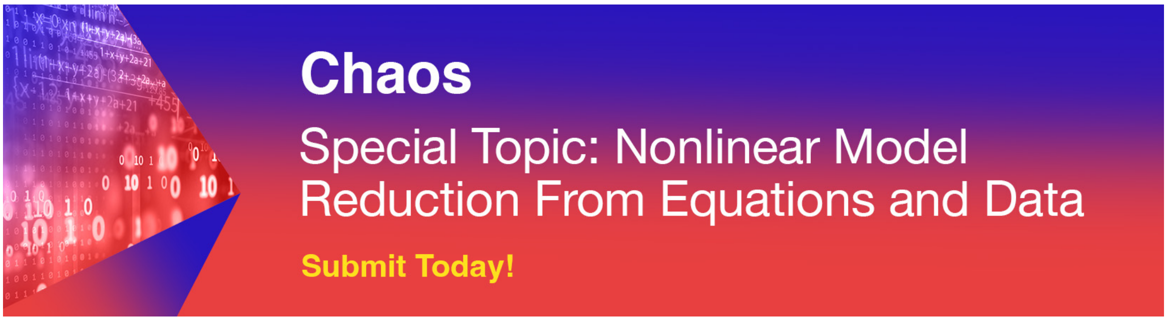
Chaos
An Interdisciplinary Journal
of Nonlinear Science

RESEARCH ARTICLE | DECEMBER 16 2022

Learning chaotic systems from noisy data via multi-step optimization and adaptive training

Lei Zhang ; Shaoqiang Tang ; Guowei He ✉

Chaos 32, 123134 (2022)
https://doi.org/10.1063/5.0114542



AIP Publishing

Learning chaotic systems from noisy data via multi-step optimization and adaptive training

Cite as: Chaos **32**, 123134 (2022); doi: 10.1063/5.0114542
Submitted: 26 July 2022 · Accepted: 17 November 2022 ·
Published Online: 16 December 2022

View Online Export Citation CrossMark

Lei Zhang,[1,2] Shaoqiang Tang,[3] and Guowei He[1,2,a)]

AFFILIATIONS

[1]The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Engineering Science, University of Chinese Academy of Sciences, Beijing 100049, China
[3]HEDPS and LTCS, College of Engineering, Peking University, Beijing 100871, China

[a)]**Author to whom correspondence should be addressed:** hgw@lnm.imech.ac.cn

ABSTRACT

A data-driven sparse identification method is developed to discover the underlying governing equations from noisy measurement data through the minimization of Multi-Step-Accumulation (MSA) in error. The method focuses on the multi-step model, while conventional sparse regression methods, such as the Sparse Identification of Nonlinear Dynamics method (SINDy), are one-step models. We adopt sparse representation and assume that the underlying equations involve only a small number of functions among possible candidates in a library. The new development in MSA is to use a multi-step model, i.e., predictions from an approximate evolution scheme based on initial points. Accordingly, the loss function comprises the total error at all time steps between the measured series and predicted series with the same initial point. This enables MSA to capture the dynamics directly from the noisy measurements, resisting the corruption of noise. By use of several numerical examples, we demonstrate the robustness and accuracy of the proposed MSA method, including a two-dimensional chaotic map, the logistic map, a two-dimensional damped oscillator, the Lorenz system, and a reduced order model of a self-sustaining process in turbulent shear flows. We also perform further studies under challenging conditions, such as noisy measurements, missing data, and large time step sizes. Furthermore, in order to resolve the difficulty of the nonlinear optimization, we suggest an adaptive training strategy, namely, by gradually increasing the length of time series for training. Higher prediction accuracy is achieved in an illustrative example of the chaotic map by the adaptive strategy.

Governing equations are fundamental to physical systems. The rapid-development data-driven approaches provide a powerful tool to discover the governing equations from data even for the complex systems or behaviors that are difficult to derive by using conventional approaches from the physical principles. However, many conventional data-driven methods only use one-step models, such as regression-based methods, which leads to difficultly deal with the noisy data from measurements. Furthermore, inaccurate predictions due to noise result in a failure for extrapolation or long-time predictions, especially for the predictions around the bifurcation points in the chaotic systems. To address this issue, in this work, a data-driven sparse identification method to discover the underlying governing equations from noisy data, called the Multi-Step-Accumulation (MSA) method, is proposed. We take into account multi-step error accumulation and suggest an adaptive training strategy to overcome the difficulty in the multi-step optimization, especially for a long duration of time. The proposed method is numerically shown to be robust and accurate and presents a highly accurate prediction for discovering chaotic systems around the bifurcation points from noisy data.

I. INTRODUCTION

Governing equations provide fundamental models in a mathematical way for physical systems. With the rapid development of data science, data-driven approaches provide a powerful tool to discover the unknown governing equations from data.[1] Consider governing equations of the general form in a discrete map

$$x_k = f(x_{k-1}) \tag{1}$$

or in continuous dynamical system

$$\dot{x} = f(x(t)), \tag{2}$$

where $\dot{x}$ denotes the derivative with respect to time t, and the explicit form of $f(x)$ is priorly unknown. Data-driven modeling of dynamics aims to discover unexplored function $f(x)$ from data only. However, experimental data or measurements with large noises remain a big challenge for these methods. On the one hand, many methods only focus on the local relationship of dynamics, such as regression-based methods,[2–4] which, thus, has a weak ability to resist the noise. In particular, many denoising techniques are based on the continuity of data, and, hence, these techniques might not work well for the data in a discrete map due to the discontinuity. On the other hand, inaccurate predictions can occur for extrapolation or long-time predictions and even might lead to errors in a trajectory along a wrong bifurcation branch in the chaotic systems. To address the above issues, the objective of the present study is to develop a data-driven model of nonlinear dynamical systems, which is robust and accurate in predictions.

Recently, with the development of data science, the data-driven models for discovery of nonlinear dynamics, such as classical linear approaches,[2,3] nonlinear autoregressive models,[5] a stepwise regression variational system identification method and a Bayesian inference approach,[6] dynamic mode decomposition,[7–9] reservoir computing approaches,[10] and neural networks,[11–15] have been proposed. The above methods are successfully used to reconstruct the equations in an approximate form. However, an extrapolation or a long-time prediction of complex nonlinear systems with bifurcations or even with chaos remains challenging for these approximations. Notably, a novel data-driven method based on sparse representation, named the Sparse Identification of Nonlinear Dynamics method (SINDy),[4] was proposed and able to provide the explicit form of $f(x)$. SINDy assumes that $f(x)$ is a linear combination of possible candidate functions, and the corresponding coefficients are the unknown parameters to be identified. This provides opportunities to yield function forms that are the same as the original systems, which benefit the extrapolation and long-time predictions. Uncomplicated implementations and good performances make SINDy widely used and developed in many fields, such as slow timescale dynamics,[16] disease dynamics,[17] vortex-induced vibration,[18] algebraic Reynolds-Stress model,[19] multiscale nonlinear dynamics,[20] rapid model recovery from abrupt system changes,[21] chemical processes,[22,23] and so on. However, when considering noisy measurements in practice, SINDy has two limitations: the method focuses on the local relationship between two neighboring snapshots, which leads to a weak resistance to the noise; the gradients are obtained by the finite difference scheme numerically, which is heavily influenced by the noise. Due to these, in spite of denoising technique used in the implementation of the SINDy, the results are still sensitive to the noise level. Xu *et al.*[24,25] combined the neural network to generate additional meta-data and calculate derivatives for better robustness, while the neural network requires large datasets and numerical cost. In addition, these denoising techniques might not be proper for the discrete map due to the discontinuity of data. As a result, when dealing with the chaotic systems around the bifurcation points from the noisy measurements, a small deviation of predictions due to noises likely leads to errors in a trajectory, as illustrated in Sec. III with examples of the logistic map, the Lorenz system, and a reduced order model of a self-sustaining process. Thus, it is necessary to find a robust and accurate identification approach in such cases.

To address this, we propose a multi-step model considering error accumulation, called the Multi-Step-Accumulation (MSA) method. We generate the predicted data based on the approximate dynamical systems with the form of sparse representation with a library of possible candidate functions. Our loss function accounts for the total error between measured series and predicted series with the same initial point. In another word, the prediction at a step is obtained from an approximate evolution scheme until this step instead of the previous one-step measurements. Compared with SINDy, which only focuses on the regression relationship between two neighboring steps, the proposed MSA method considers multi-step error accumulation over the prediction horizon and, thus, has a better chance to achieve accurate multi-step (long-time) predictions and resists the corruption of noise. Combined with time integration scheme, MSA has no need to compute gradients and avoids the error from numerical gradients.

Multi-step optimization has been used for different purposes. The paper[26] used it to construct the neural network for learning. Neural networks combining with multi-step optimization are proposed to model system dynamics from observations in Ref. 27. However, the number of steps in Ref. 27 is usually small, and the author pointed out that "the training did not converge for using lookahead = 30" for the chaotic Lorenz attractor ("lookahead" refers to the number of steps in Ref. 27). As a comparison, we achieve 200-step optimization via an adaptive training strategy for the continuous dynamical system including the Lorenz system, as illustrated in numerical examples. The multistep neural networks proposed in Ref. 12 choose a multi-step integration scheme considering multi-step states in one integration time step to calculate the loss. The difference of the present method from multistep neural networks is as follows: in multistep neural networks, the error for a single time step is used for optimization, while a multi-step time-stepping scheme is used for the evolution of systems; in the present method, the accumulated error for multiple time steps is used for optimization, while a multi-step or one-step time-stepping scheme is used for evolution.

The MSA method presents a highly accurate prediction for discovering chaotic systems around the bifurcation points. We shall illustrate this with numerical examples including the logistic map, and a reduced order model of a self-sustaining process in turbulent shear flows. However, the multi-step model of nonlinear dynamics leads to the difficulty of optimization in the numerical implementations. We, thus, adopt an adaptive training strategy to address this issue. To test our method, we further consider several cases, including discrete maps and continuous dynamical systems, noisy measurements, and missing data.

The rest of the paper is organized as follows. Section II presents the proposed MSA method for discrete map and continuous dynamical systems, and then the adaptive training strategy. Numerical examples under several conditions are given in Sec. III, including a two-dimensional chaotic map, the logistic map, a two-dimensional damped oscillator, the Lorenz system, and a reduced order model of a self-sustaining process in turbulent shear flows.

II. THE MULTI-STEP-ACCUMULATION (MSA) METHOD

Inferring dynamical systems from data is a useful tool to represent and understand the new physical phenomenon. We propose a class of sparse identification methods considering multi-step error accumulation to discover governing equations from noisy measurement data, called the Multi-Step-Accumulation (MSA) method, which performs good robustness and accuracy. We first introduce our method for discrete maps in Sec. II A. Then, we combine the Runge–Kutta scheme and extend the method to continuous dynamical systems in Sec. II B. Finally, an efficient adaptive training strategy is given in Sec. II C to facilitate the implementation.

A. Discovering governing equations of discrete maps

Consider a discrete map of form (1). We obtain a set of time-series data $\boldsymbol{X} = [x_1, x_2, \dots, x_S]^T$ with initial point x_1, where S is the length of the time series.

To infer the unknown $f(x)$ from data, the key idea in MSA is to generate approximate series based on the sparse representation, i.e., to assume that $f(x)$ is approximated by a linear combination of specific basis functions from a prescribed library $\boldsymbol{\Theta}(x)$ with lots of candidate functions,

$$f(x) = \boldsymbol{\Theta}(x)\boldsymbol{\xi}. \tag{3}$$

For example, the library might consist of constant, polynomial, trigonometric, exponential functions, etc.,

$$\boldsymbol{\Theta}(x) = [1, x, x^2, x^3, \dots, x^p, \sin(x), \cos(x), \dots, \exp(x), \dots]. \tag{4}$$

The coefficient vector $\boldsymbol{\xi}$ is sparse, that is to say, most entries are zero, meaning only a few candidate functions are selected (active).

Thus, the approximate dynamical system reads

$$o_k = \boldsymbol{\Theta}(o_{k-1})\boldsymbol{\xi}, \tag{5}$$

with the same initial point

$$o_1 = x_1. \tag{6}$$

This gives an S-step approximate time series $\boldsymbol{O} = [o_1, o_2, \dots, o_S]^T$. The discovery of equations then becomes a sparse regression problem to determine the sparse coefficient vector $\boldsymbol{\xi}$. Considering the time-history, we define the accumulated error between approximate time series and real ones as the loss function with respect to $\boldsymbol{\xi}$,

$$loss = \frac{1}{S-1}\|\boldsymbol{O} - \boldsymbol{X}\|_{L^2}^2 = \frac{1}{S-1}\sum_{k=2}^{S}(o_k - x_k)^2. \tag{7}$$

For example, when $S = 3$, the loss reads

$$\begin{aligned} loss &= \frac{1}{2}\sum_{k=2}^{S=3}(o_k - x_k)^2 \\ &= \frac{1}{2}\left((o_2 - x_2)^2 + (o_3 - x_3)^2\right) \\ &= \frac{1}{2}\left((\boldsymbol{\Theta}(o_1)\boldsymbol{\xi} - x_2)^2 + (\boldsymbol{\Theta}(o_2)\boldsymbol{\xi} - x_3)^2\right) \\ &= \frac{1}{2}\left((\boldsymbol{\Theta}(x_1)\boldsymbol{\xi} - x_2)^2 + (\boldsymbol{\Theta}(\boldsymbol{\Theta}(x_1)\boldsymbol{\xi})\boldsymbol{\xi} - x_3)^2\right). \end{aligned} \tag{8}$$

For a general S, the loss reads

$$\begin{aligned} loss = \frac{1}{S-1}&\left((\boldsymbol{\Theta}(x_1)\boldsymbol{\xi} - x_2)^2 + (\boldsymbol{\Theta}(\boldsymbol{\Theta}(x_1)\boldsymbol{\xi})\boldsymbol{\xi} - x_3)^2\right. \\ &\left. + \cdots + (\boldsymbol{\Theta}(\boldsymbol{\Theta}(\cdots\boldsymbol{\Theta}(\boldsymbol{\Theta}(x_1)\boldsymbol{\xi})\boldsymbol{\xi}\cdots)\boldsymbol{\xi})\boldsymbol{\xi} - x_S)^2\right). \end{aligned} \tag{9}$$

Generally, there exist nonlinear terms in the library $\boldsymbol{\Theta}$. Thus, the optimization problem is nonlinear, and (9) shows that the complexity of the form of the loss increases with S increasing, which is the main difficulty in the optimization. The last term in (9) performs like an S-layer neural network with $\boldsymbol{\Theta}$ as the activation functions. Here, S is equivalent to the depth of neural networks. A large-number composition of non-linear functions leads to the issue of vanishing or exploding gradients in optimization, which makes the learning of long-term dependencies particularly difficult.[28] In the numerical implementations, we found that the training hardly converges for large S without the following adaptive training strategy in Sec. II C. Furthermore, Ref. 27 also pointed out that "the training did not converge for using lookahead = 30" for the chaotic Lorenz attractor ("lookahead" refers to the number of steps S).

The system property is another factor to influence the optimization. In this paper, we focus on chaotic systems, especially the issue to discover the dynamics near the bifurcation points. The trajectories are, thus, sensitive to the parameters, which increases the difficulties to identify correct parameters via optimization. In contrast, if a system is stable,[27] it pointed out that the training of such multi-step optimization should converge for any choice of S. In the following numerical examples, when taking $S = 40$, we can apply MSA to the identification of damped oscillators directly without the following adaptive strategy but fail to deal with other chaotic systems at the same S directly.

As a comparison, the loss of SINDy[4] for the same time series $\boldsymbol{X} = [x_1, x_2, \dots, x_S]^T$ is

$$\begin{aligned} loss^{\text{SINDy}} = \frac{1}{S-1}&\left((\boldsymbol{\Theta}(x_1)\boldsymbol{\xi} - x_2)^2 + (\boldsymbol{\Theta}(x_2)\boldsymbol{\xi} - x_3)^2\right. \\ &\left. + \cdots + (\boldsymbol{\Theta}(x_{S-1})\boldsymbol{\xi} - x_S)^2\right). \end{aligned} \tag{10}$$

The corresponding optimization problem is a simple least-square problem without a composition of non-linear functions.

Different from SINDy that focuses on the regression relationship between the states of the current step and the previous step, the proposed method is a *multi-step* model. We consider error accumulation over S steps, as key to achieving higher accuracy. On the other hand, the loss function measures the deviation of the approximate motion from a multi-step (long-time) view, so the method is able to capture the long-time dynamical behavior from the measurements, enabling resistance to the corruption of noise.

Note that we can adopt multiple sets of time series as training data to further improve the performance. For example, suppose we have m time series

$$\begin{aligned} \boldsymbol{X}^{(1)} = [x_1^{(1)}, x_2^{(1)}, \dots, x_S^{(1)}]^T, \boldsymbol{X}^{(2)} = [x_1^{(2)}, x_2^{(2)}, \dots, x_S^{(2)}]^T, \dots, \\ \boldsymbol{X}^{(m)} = [x_1^{(m)}, x_2^{(m)}, \dots, x_S^{(m)}]^T, \end{aligned} \tag{11}$$

with the superscript (l) counting the time-series index and S length of the time series. The approximate dynamical system (5) induces the approximate time series correspondingly,

$$\boldsymbol{O}^{(1)} = \left[o_1^{(1)}, o_2^{(1)}, \ldots, o_S^{(1)}\right]^T, \boldsymbol{O}^{(2)} = \left[o_1^{(2)}, o_2^{(2)}, \ldots, o_S^{(2)}\right]^T, \ldots, \tag{12}$$

$$\boldsymbol{O}^{(m)} = \left[o_1^{(m)}, o_2^{(m)}, \ldots, o_S^{(m)}\right]^T,$$

with the same initial points as those of original time series $\boldsymbol{X}^{(l)}$, $l = 1, 2, \ldots, m$, i.e.,

$$o_1^{(l)} = x_1^{(l)}, l = 1, 2, \ldots, m. \tag{13}$$

The loss function is the mean of the error for each pair $\{\boldsymbol{O}^{(l)}, \boldsymbol{X}^{(l)}\}$, $l = 1, 2, \ldots, m$,

$$\begin{aligned} loss &= \frac{1}{m(S-1)} \sum_{l=1}^{m} \left\| \boldsymbol{O}^{(l)} - \boldsymbol{X}^{(l)} \right\|_{L^2}^2 \\ &= \frac{1}{m(S-1)} \sum_{l=1}^{m} \sum_{k=2}^{S} \left(o_k^{(l)} - x_k^{(l)}\right)^2. \end{aligned} \tag{14}$$

The coefficient vector $\boldsymbol{\xi}$ is determined by minimizing the above loss function with consideration of sparsity.

Remark that the resulting optimization problem is not a simple linear regression but a nonlinear one. This requires a numerical optimization algorithm, such as gradient-based methods, Newton's method, and quasi-Newton methods. In the following, to balance the accuracy and efficiency, we adopt L-BFGS method, one of the quasi-Newton methods, to solve this nonlinear optimization problem. For large S, the optimization becomes difficult, and the algorithm might be divergent. An adaptive strategy for training will be presented in Sec. II C.

Furthermore, the calculation of gradients can be implemented by borrowing the backward-propagation (BP) idea, which considerably enhances efficiency. For details, please refer to the Appendix. The complexity of the calculation of gradients is of the order $O(NdL)$, with the number of data points $N = m \times S$, the dimension d, and the number of active candidate functions L. To improve the computational efficiency, an alternative way is to reduce L via group sparsity approaches.

There are several approaches to address the sparsity of $\boldsymbol{\xi}$, such as sequentially thresholded least squares (STLSQ),[4] least absolute shrinkage and selection operator (LASSO),[29] sparse relaxed regularized regression (SR3),[30] stepwise sparse regression (SSR),[31] and Bayesian approaches.[32] Here, following the work in Ref. 4, we adopt the STLSQ algorithm.

When $S = 2$, MSA reduces to SINDy based on a small part of the database, i.e., one only uses first two data points of each time series. In this case, we have

$$o_2^{(l)} = \boldsymbol{\Theta}\left(o_1^{(l)}\right)\boldsymbol{\xi} = \boldsymbol{\Theta}\left(x_1^{(l)}\right)\boldsymbol{\xi},\ l = 1, 2, \ldots, m, \tag{15}$$

and the loss function (14) reduces to

$$loss = \frac{1}{m}\sum_{l=1}^{m}\left(o_2^{(l)} - x_2^{(l)}\right)^2 = \frac{1}{m}\sum_{l=1}^{m}\left(\boldsymbol{\Theta}(x_1^{(l)})\boldsymbol{\xi} - x_2^{(l)}\right)^2. \tag{16}$$

Of course, the corresponding optimization problem reduces to the least-square problem with sparsity.

ALGORITHM 1. STLSQ algorithm for sparsity.

```
1. Minimize the loss function (14) using the L-BFGS method and
   obtain optimal coefficients $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots]^T$.
2. Set a small quantity $\lambda$.
   Loop until the solution $\boldsymbol{\xi}$ does not change:
      (a) If the term of the coefficient vector $|\xi_k| < \lambda$, we enforce
          $\xi_k = 0$, and do not optimize $\xi_k$ in the following process;
      (b) Solve $\boldsymbol{\xi}$ by minimizing the loss function (14) again with
          the rest non-zero entries.
```

B. Discovering governing equations of continuous dynamical systems with a Runge–Kutta scheme

We then extend the MSA method to the dynamical system of form (2) with a discrete time scheme given by

$$x(t_k) = F\left(x(t_{k-1}), f(x(t_{k-1}))\right). \tag{17}$$

For convenience and clarity, we denote $x_k = x(t_k)$.

We can adopt the MSA method to infer $f(x)$ from data in the same manner. First, $f(x)$ is approximated by $\boldsymbol{\Theta}(x)\boldsymbol{\xi}$, where $\boldsymbol{\Theta}$ is a library of candidate functions and $\boldsymbol{\xi}$ is the unknown and sparse coefficient vector. Then, we obtain the following approximate dynamical system in discrete form:

$$o_k = F(o_{k-1}, \boldsymbol{\Theta}(o_{k-1})\boldsymbol{\xi}). \tag{18}$$

We remark that the numerical integration scheme in MSA can be any explicit numerical solver. For precisely, we choose the fourth order Runge–Kutta (RK4) time scheme as illustration in the numerical examples due to its good performance and stability.

The loss function is the error between the time-series data $\boldsymbol{O} = [o_1, o_2, \ldots, o_S]^T$ from (18), and the collected data $\boldsymbol{X} = [x_1, x_2, \ldots, x_S]^T$ from (2) with the same initial point $o_1 = x_1$. The process in MSA for continuous dynamical systems is formulated as

$$\begin{aligned} &\text{find} && \boldsymbol{\xi} \\ &\text{min} && loss = \frac{1}{S-1}\|\boldsymbol{O} - \boldsymbol{X}\|_{L^2}^2 = \frac{1}{S-1}\sum_{k=2}^{S}(o_k - x_k)^2 \\ &\text{where} && o_k = F(o_{k-1}, \boldsymbol{\Theta}(o_{k-1})\boldsymbol{\xi}), k = 2, 3, \ldots, S \\ &\text{and} && o_1 = x_1. \end{aligned} \tag{19}$$

C. Numerical implementations for optimization: An adaptive strategy for training

As mentioned before, the L-BFGS method might fail for large S. To address this issue, we propose the following adaptive training strategy to facilitate the optimization, i.e., using the solution for a small S as the initial guess to optimize with a large one. That is, we gradually increase the length of the time series in updating the coefficient vector $\boldsymbol{\xi}$.

ALGORITHM 2. Adaptive training strategy for MSA.

```
1. Collect $m$ time series $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \dots, \boldsymbol{X}^{(m)}$ of the length $M$.
   (a) Solve coefficients $\boldsymbol{\xi}$ by minimizing loss (14) for small $S$ and
       adopt STLSQ algorithm (Algorithm 1) to reduce the number
       of non-zero entries simultaneously.
           $\xi^{S} = \underset{\xi}{\text{argmin}}\ loss, \quad \text{(STLSQ)}. \qquad (20)$
   (b) The sparse results are set to be the initial guess $\boldsymbol{\xi}_0$ for the
       next step.
           $\boldsymbol{\xi}_0 \leftarrow \boldsymbol{\xi}^{S}. \qquad (21)$
2. While $S \le M$ do
   (a) Minimize the loss (14) with a larger $S + \Delta S$, $\Delta S \in \mathbb{N}^{+}$, using
       L-BFGS algorithm with an initial guess $\boldsymbol{\xi}_0$, and obtain better
       results $\boldsymbol{\xi}^{S+\Delta S}$:
           $\xi^{S+\Delta S} = \underset{\xi}{\text{argmin}}\ loss,$
               $\text{(L-BFGS with an initial guess } \boldsymbol{\xi}_0\text{)}. \qquad (22)$
   (b) Set $\boldsymbol{\xi}_0 \leftarrow \boldsymbol{\xi}^{S+\Delta S}$;
   (c) Set $S \leftarrow S + \Delta S$.
   End while.
```

The incremental length of the time series ΔS is a positive integer. We take $\Delta S = 1$ in general. Of course, we may try $\Delta S > 1$ sometimes to get better efficiency. Furthermore, the group sparsity approach can reduce the number of non-zero entries for training and, thus, reduce the difficulty and cost of the optimization. To further improve the efficiency, the STLSQ algorithm is implemented merely on the SINDy ($S = 2$) or the training with short time series, not on large S.

III. RESULTS AND DISCUSSIONS

In this section, we present numerical examples to test our method, including discrete maps and continuous dynamical systems. We take the classical SINDy[4] without denoising as a comparison. In the implementations, we may need to normalize the columns of $\boldsymbol{\Theta}(\boldsymbol{x})$ by their standard deviations first to ensure that the restricted isometry property holds[33,34] for SINDy. Remark that we do not need such normalization for MSA in the following examples.

To demonstrate the robustness of the proposed MSA method, we perform a numerical study under several conditions, such as measured data with additional white Gaussian noises, i.e.,

$$Data^{Measure} = Data^{Real} + \epsilon, \tag{23}$$

with $Data^{Measure}$ the measured data for training, $Data^{Real}$ the real data from the real systems, and ϵ an independent identically (i.i.d.) Gaussian noise,

$$\epsilon \sim \mathcal{N}(0, \sigma^2), \tag{24}$$

with σ the standard deviation. Here, we define the noise level following Ref. 35,

$$\eta = \frac{\sigma}{std\left(Data^{Real}\right)}, \tag{25}$$

where $std(Data^{Real})$ is the standard deviation of real data. Note that we do not add noises on the initial point of each time series in order to eliminate the effect of the initial deviation.

A. A chaotic map

The first example is a chaotic attractor generated by a non-invertible map,[36]

$$\begin{cases} x_{n+1} = 3.7x_n - (x_n)^2 - 0.1x_n y_n, \\ y_{n+1} = 3.7y_n - 0.15x_n y_n - (y_n)^2. \end{cases} \tag{26}$$

We present the plot x_{n+1} vs x_n, y_{n+1} vs y_n, and y_n vs x_n in Fig. 1, where the initial point is $(x_0, y_0) = (0.5, 0.5)$. Note that all the points are limited in the domain $[0.5, 3.5] \times [0.5, 3.5]$. We collect $m = 200$ 100-step series with random initial points for training.

In the sparse representation, the nonlinear library $\boldsymbol{\Theta}(\boldsymbol{x})$ includes the product of 1D polynomials up to the fifth order with 36 candidate terms and, thus, the approximate dynamical system is

$$\begin{cases} x_{n+1} = [1, x_n, (x_n)^2, \dots, (x_n)^5, y_n, x_n y_n, \dots, (x_n)^5 (y_n)^5]\boldsymbol{\xi}_x, \\ \\ y_{n+1} = [1, x_n, (x_n)^2, \dots, (x_n)^5, y_n, x_n y_n, \dots, (x_n)^5 (y_n)^5]\boldsymbol{\xi}_y. \end{cases} \tag{27}$$

According to the real system (23), the exact solution should be

$$\begin{aligned} \boldsymbol{\xi}_x^{real} &= [0, 3.7, -1, 0, 0, 0, 0, -0.1, 0, 0, \dots, 0]^T, \\ \boldsymbol{\xi}_y^{real} &= [0, 0, 0, 0, 0, 0, 3.7, -0.15, 0, 0, 0, 0, -1, 0, \dots, 0]^T. \end{aligned} \tag{28}$$

To measure the accuracy, we define the error

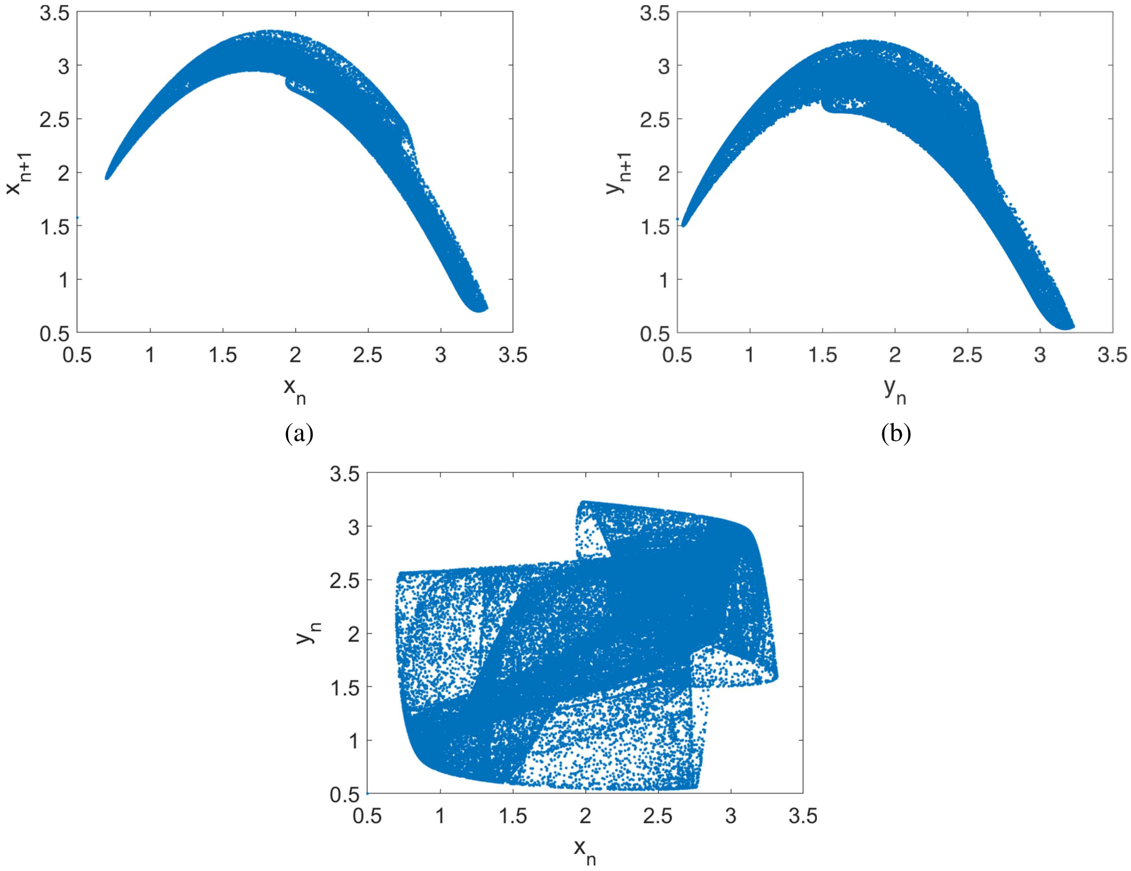
$$\Delta\boldsymbol{\xi}_x = \boldsymbol{\xi}_x^{num} - \boldsymbol{\xi}_x^{real}, \Delta\boldsymbol{\xi}_y = \boldsymbol{\xi}_y^{num} - \boldsymbol{\xi}_y^{real}, \tag{29}$$

where $\boldsymbol{\xi}_x^{num}, \boldsymbol{\xi}_y^{num}$ are the coefficients obtained by SINDy or MSA methods.

To demonstrate the ability of the present method in the model selection aspect, we first take short time series with a length $S = 20$ and then increase the length S to 100 with the adaptive strategy for training to observe the improvement in accuracy.

We compare our method with existing methods such as SINDy and Entropic Regression (ER)[37] under different noise levels. In SINDy, we normalize the data first and then use STLSQ method with a threshold $\lambda = 0.03$ to guarantee the sparsity of the coefficients. As shown in the supplementary material and Table I, for small noises ($\eta = 1.44\%$), all of three methods successfully capture the correct terms. However, when increasing noises to the level $\eta = 14.4\%$, incorrect terms occur in SINDy, while ER is still valid. However, for large noises ($\eta = 72.0\%$), both SINDy and ER fail to identify correct equations. As a comparison, MSA gives good performance for the model selection even for large noises.

In the implementations, to reduce the computational cost, we usually take a small batch (e.g., $m = 50$) of time series using the MSA method to identify a small number of correct candidate functions first, especially for large noises or big libraries. For example,



(a)

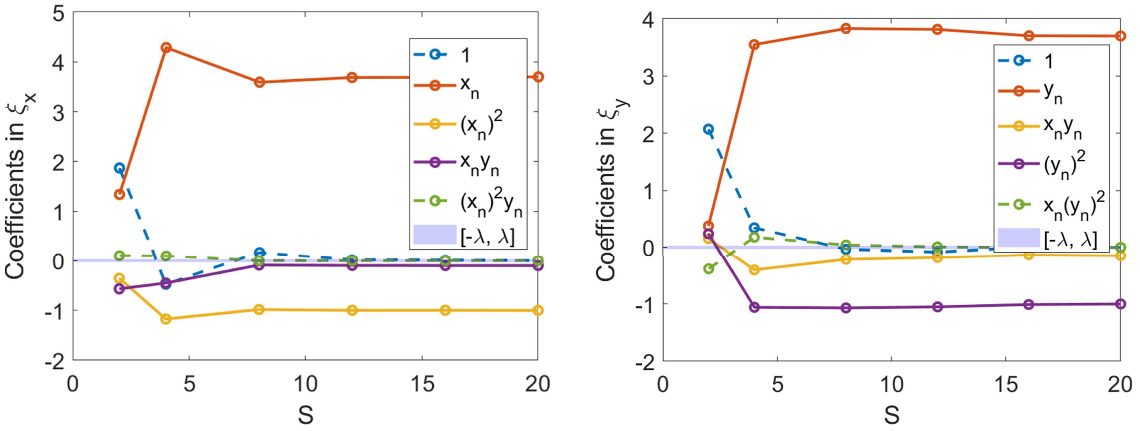
(b)

(c)

FIG. 1. 10^5 data points generated by the Ushiki map with the initial point $(x_0, y_0) = (0.5, 0.5)$: (a) x_{n+1} vs x_n; (b) y_{n+1} vs y_n; (c) y_n vs x_n. All the data points are limited in the domain $[0.5, 3.5] \times [0.5, 3.5]$. (a) x_{n+1} vs x_n; (b) y_{n+1} vs y_n; (c) y_n vs x_n.

TABLE I. Error in coefficients of SINDy, ER, and the proposed MSA method under different noise levels. We define the maximum error for coefficients, i.e., L^∞ norm of $\Delta\xi_x$ and $\Delta\xi_y$, to measure the accuracy. The measurements are corrupted by additional Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with σ being the standard deviation. Here, note that MSA uses a given length $S = 20$ of time series for training. Here, take $m = 200$ and $p = 5$.

Noise level η	SINDy	ER[37]	MSA ($S = 20$)
$\eta = 0$ (No noise)	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 8.88 \times 10^{-16}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.11 \times 10^{-15}$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 8.88 \times 10^{-16}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.11 \times 10^{-15}$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 8.88 \times 10^{-16}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.11 \times 10^{-15}$
$\eta = 1.44\%$ noise	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 4.20 \times 10^{-4}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 2.55 \times 10^{-4}$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 4.20 \times 10^{-4}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 2.55 \times 10^{-4}$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 9.45 \times 10^{-7}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 1.76 \times 10^{-6}$
$\eta = 14.4\%$ noise	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 0.472$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.761$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 0.054$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 0.153$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 6.87 \times 10^{-6}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.48 \times 10^{-5}$
$\eta = 72.0\%$ noise	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 2.359$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.323$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 3.700$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 3.700$	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 3.62 \times 10^{-5}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 6.99 \times 10^{-5}$



(a)

(b)

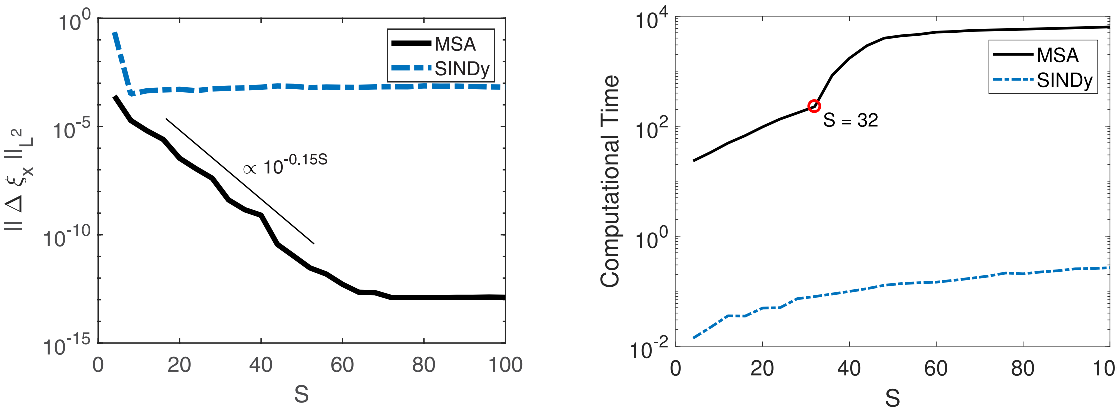
FIG. 2. History of coefficients with increasing S. Here, We take SINDy results as the initial guess and choose randomly a small batch ($m = 50$) of time series in the same data set ($\eta = 72.0\%$ noise) for training. Wrong terms are represented by dashed lines. (a) History of coefficients in ξ_x. (b) History of coefficients in ξ_y.

we test MSA method with merely $m = 50$ time series for training, and illustrate a history of coefficients of candidate terms with S increasing in Fig. 2. SINDy provides additional incorrect terms for large noises ($\eta = 72.0\%$ noise) (see the supplementary material). We take SINDy results as the initial guess, and choose randomly $m = 50$ time series in the same data set for training. As shown in the figure, correct terms (solid lines) are identified successfully, and wrong terms (dashed lines) vanish with S increasing. This clearly shows the importance of parameter S in the model selection aspect.

As shown in Table I, MSA shows a high accuracy with three or four orders of magnitude less error than SINDy and seems insensitive to the noise level. Although facing strong noise, the present method still maintains a good accuracy with a 10^{-5} error.

1. Discussions about hyperparameters S, m, and p

There exist three hyperparameters in MSA including length of time-series (number of steps) S, number of training time-series m, and the size of library K. For convenience, we focus on the



(a)

(b)

FIG. 3. Accuracy and computational time with increasing S. S is the length of time series used in the training ($m = 1000, \sigma = 0.01, \eta = 1.44\%$). The accuracy of MSA improves exponentially with increasing time series length S, until the error saturates to 10^{-14}, which is the tolerance set for the L-BFGS algorithm. When $S < 64$, the error decays with a rate about $10^{-0.15S}$. The computational time of MSA, thus, increases with S increasing. Compared with SINDy, MSA achieves higher accuracy with more computational cost. (a) $\|\Delta\xi_x\|_{L^2}$ vs S. (b) Computational time vs S.

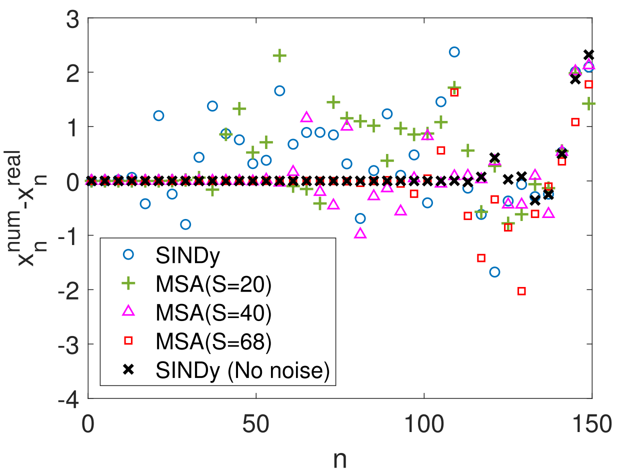


FIG. 4. The error of different prediction trajectories for different sizes of time series used with the same initial point. x_n^{real} represents the nth point of the real time series, and x_n^{num} represents the prediction by SINDy or MSA, where the subscript n is the counting index for steps of the predicted trajectory. The prediction accuracy improves with increasing S.

polynomials and denote the order of 1D polynomials as p. In particular, the library for 2D problem is

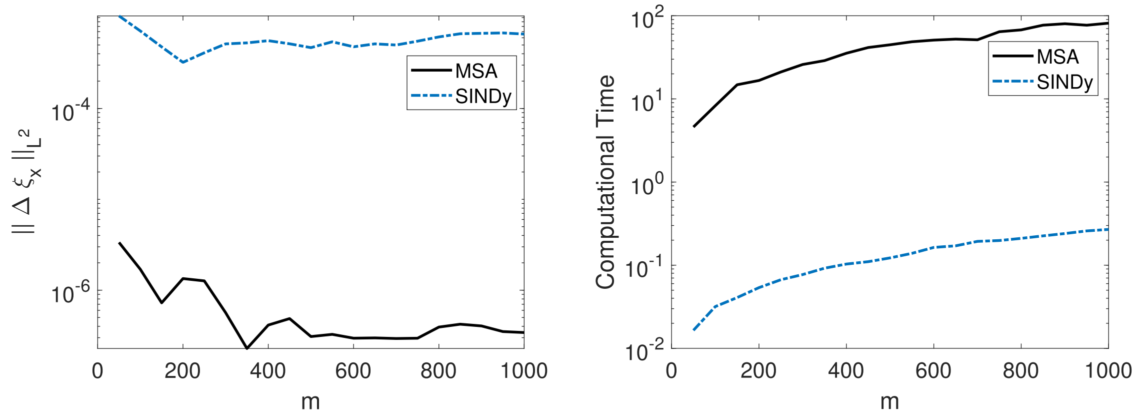
$$\mathbf{\Theta}(x,y) = \left[1, x, x^2, \ldots, x^p, y, xy, x^2y, \ldots, x^py^p\right], \tag{30}$$

with a size $K = (p+1)^2$. The problem is solved with a single CPU core of Intel Core i7-11700F in Matlab.

In the previous cases, a good accuracy is achieved under a length $S = 20$ of time series. Here, we adopt the adaptive strategy for large S and study the performance of the proposed method with S increasing. Take $m = 1000$ and $p = 5$. The noise level is $\sigma = 0.01$ and $\eta = 1.44\%$. We gradually increase the length S of time series using the adaptive strategy for training. The error $\|\Delta\boldsymbol{\xi}_x\|_{L^2}$ vs S is shown in Fig. 3(a). The accuracy improves exponentially with increasing time series length S, until the error saturating to 10^{-14}, which is the tolerance set for the L-BFGS algorithm. Meanwhile, the computational time also increases with S. A significant increase of computational time occurs at $S = 32$. As a comparison, the error for coefficients of SINDy remains about $10^{-3} - 10^{-4}$, while the cost increases slowly. In fact, the computational time of SINDy is almost linear with S. MSAs are able to achieve higher accuracy than SINDy although with more computational cost.

In Fig. 4, we plot the prediction error of a time series for different methods including SINDy and MSA with different S, with the same initial point $(x_0, y_0) = (0.5, 0.5)$. x_n^{real} represents the nth point of the real time series, and x_n^{num} represents the prediction by SINDy or MSA. As shown in the figure, the range of successful long-time predictions ($|x_n^{num} - x_n^{real}| < 10^{-3}$) directly reflects the prediction accuracy of the parameters with the length of series S increasing. Note that the SINDy results trained by real data (without noise) can reach a very high accuracy with a 10^{-16} error as shown in Table I, representing by the black crosses in Fig. 4. This error mostly comes from the machine precision. However, the exact predictions with clear measurements cannot remain over about 120 steps due to the chaotic property. The adaptive strategy for MSA does not improve the accuracy significantly when the error approaches 10^{-16} (the machine precision). It is the limit of the accuracy, and we suggest stopping the optimization before $S = 68$ in this problem to avoid additional cost.

Next, we fix $S = 20$, and observe the performance of methods with m altered. As shown in Fig. 5, both MSA and SINDy hardly improve the accuracy when $m \geq 350$. The computational time also increases with m increasing. Comparing Fig. 5 with Fig. 3, we can observe that S dominates the performance of MSA, and its increase leads to a significant improvement of accuracy and growth of computational cost.



(a) (b)

FIG. 5. Accuracy and computational time with increasing m. m is the number of time series for training (here, $S = 20, \sigma = 0.01, \eta = 1.44\%$). Both MSA and SINDy hardly improve the accuracy when $m \geq 350$. The computational time increases with m increasing. (a) $\|\Delta\xi_x\|_{L^2}$ vs m. (b) Computational time vs m.

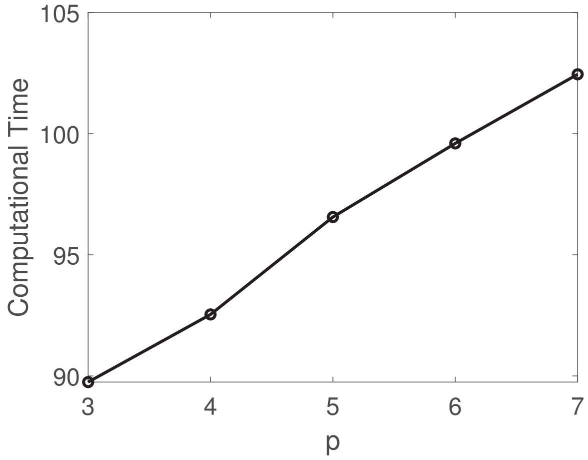


FIG. 6. Computational time with increasing p. p is the order of polynomials in the library. The library size is $(p+1)^2$.

Finally, we present the computational time with increasing order of polynomials p, i.e., increasing size of library. As shown in Fig. 6, the increase of the computation time is about 10%–20%. The reason is that we identify correct terms via sparsity algorithms for small S. Hence, although the size of the library increases, the active terms remain the same for large S.

Additionally, the dimension of systems d also affects the computational cost of the proposed method. On the one hand, for the same order of polynomials p, the size of the library K increases exponentially with the dimension d, i.e., $K=(p+1)^d$. On the other hand, the complexity of the calculation of gradients is of the order $O(NdL)$ (details refer to the Appendix), with the number of data points $N=m\times S$, the dimension d, and the number of active candidate functions L. The dimension d directly affects the computational cost of optimization.

2. Missing data case

Next, we consider a challenging case with missing data. In practice, we might not get a complete time series. The following numerical results show that MSA still works well in this case.

Drop about 30% data in each time series ($m=200$), as illustrated in Fig. 7. Each row represents a time series with blocks representing data and blank as missing data. The numerical results show that 30% missing data does not influence the identification of candidate terms of MSA, so we just list the error in coefficients in Table II. Compared with Table I, under the same noise level, the proposed method maintains high accuracy, only slightly influenced by the missing data.

3. Colored noise

In the previous study, the noises are i.i.d. Here, we test our method with colored noises, defined by

$$\epsilon_k^{Color}=\frac{1}{2}\epsilon_k^{White}+\frac{1}{2}\epsilon_{k-1}^{White}, \tag{31}$$

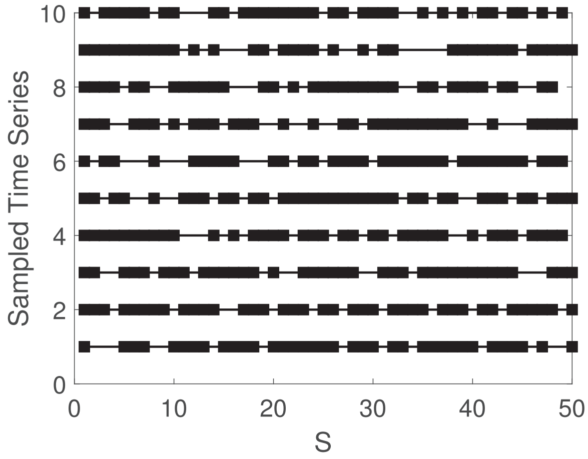


FIG. 7. Illustration for missing data of each time series. There are ten time series with about 30% missing data as illustrated in the figure. Each row (solid line) represents a time series with black blocks representing data and blank as missing data.

with

$$\epsilon_k^{White}\sim\mathcal{N}(0,\sigma^2), \tag{32}$$

and k denoting counting index for steps in a time series. The error in coefficients with different noise levels η is listed in Table III. Compared with white Gaussian noises case in Table I, there is no significant difference for MSA with white noises and colored noises.

B. Logistic map and bifurcations

In this subsection, we take the logistic map to exhibit the importance of highly accurate predictions for chaotic systems around a bifurcation point. The logistic map is a classical model that exhibits a cascade of bifurcations.

The logistic map is given by

$$x_{n+1}=rx_n(1-x_n). \tag{33}$$

Here, r is the parameter, which directly influences the dynamical behavior of the system, as shown in Fig. 8. Set $r=3.626$ and additional Gaussian noise $\epsilon\sim\mathcal{N}(0,0.01^2)$ with noise level $\eta=4.61\%$.

TABLE II. Accuracy of MSA when facing 30% missing data in each time series. The measurement is corrupted by a Gaussian noise $\epsilon\sim\mathcal{N}(0,\sigma^2)$ with σ the standard deviation. Here, take $m=200$, $S=20$, and $p=5$.

Noise level	MSA ($S=20$)
$\eta=1.44\%$ noise	$\lVert\Delta\boldsymbol{\xi}_x\rVert_{L^\infty}=1.72\times10^{-6}$ $\lVert\Delta\boldsymbol{\xi}_y\rVert_{L^\infty}=1.78\times10^{-6}$
$\eta=14.4\%$ noise	$\lVert\Delta\boldsymbol{\xi}_x\rVert_{L^\infty}=6.39\times10^{-6}$ $\lVert\Delta\boldsymbol{\xi}_y\rVert_{L^\infty}=4.59\times10^{-5}$
$\eta=72.0\%$ noise	$\lVert\Delta\boldsymbol{\xi}_x\rVert_{L^\infty}=9.39\times10^{-5}$ $\lVert\Delta\boldsymbol{\xi}_y\rVert_{L^\infty}=9.80\times10^{-5}$

TABLE III. Error in coefficients of the proposed MSA method for colored noise. We define the maximum error for coefficients, i.e., L^∞ norm of $\Delta\xi_x$ and $\Delta\xi_y$, to measure the accuracy. Here, take $m = 200$, $S = 20$, and $p = 5$.

Noise level	MSA (S=20)
$\eta = 1.44\%$ noise	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 7.10 \times 10^{-7}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 7.93 \times 10^{-7}$
$\eta = 14.4\%$ noise	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 2.31 \times 10^{-6}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 1.40 \times 10^{-5}$
$\eta = 72.0\%$ noise	$\|\Delta\boldsymbol{\xi}_x\|_{L^\infty} = 7.76 \times 10^{-5}$ $\|\Delta\boldsymbol{\xi}_y\|_{L^\infty} = 7.76 \times 10^{-5}$

There are 1000 100-step series for training. The candidate functions in the library are chosen as polynomials up to the fifth order. The predicted results of SINDy is

$$x_{n+1} = 3.6196x_n - 3.6168(x_n)^2, \tag{34}$$

with $\lambda = 0.02$ in the STLSQ method, while that of MSA is

$$x_{n+1} = 3.6260x_n - 3.6260(x_n)^2. \tag{35}$$

with an error in coefficients less than 10^{-13} ($S = 100$).

As shown in Fig. 8, due to the different prediction accuracies of these two methods, the attracting set of MSA is consistent with that of the real system, whereas SINDy gives a totally different one. The SINDy predictions perform like the logistic map at about $r \approx 3.619$, whose behavior is different from the real case due to the bifurcation. Starting from the same initial point ($x_1 = 0.5$), we plot these three trajectories of real systems, predictions of SINDy, and MSA. As shown in Fig. 9, our method remains consistent with the real system until about the 190th step, while SINDy predictions differ from the other two curves significantly at about the 15th step.

C. Two-dimensional damped oscillators

Then, MSA applies to the continuous dynamical systems. We start with a simple two-dimensional damped harmonic oscillator, governed by

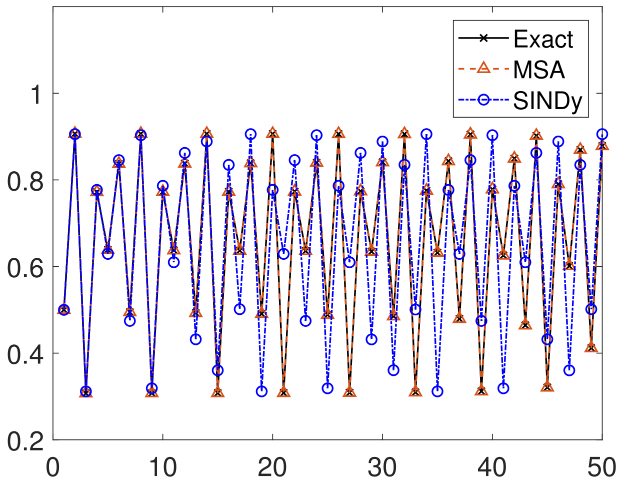


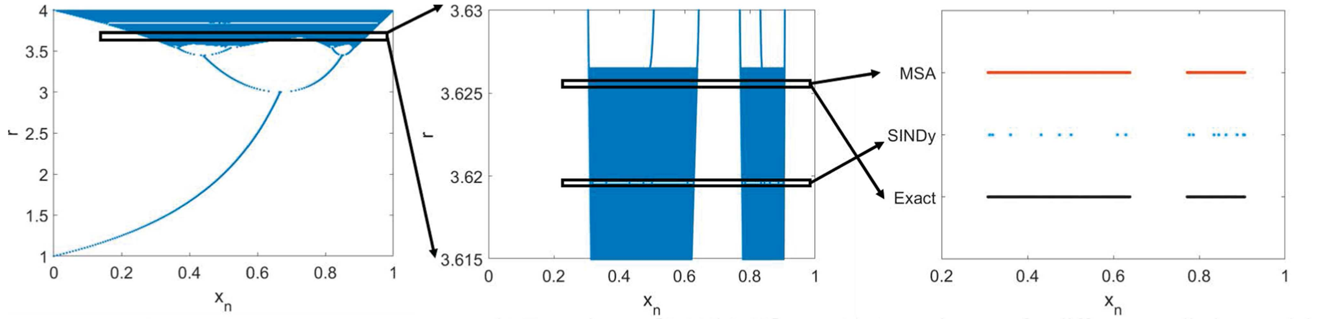
FIG. 9. Three trajectories of real systems, predictions of SINDy, and MSA. The initial point is ($x_1 = 0.5$).

$$\begin{cases} \dot{x} = -0.1x + 2y, \\ \dot{y} = -2x - 0.1y. \end{cases} \tag{36}$$

The eigenvalues of the coefficient matrix are $-0.1 + 2i$ and $-0.1 - 2i$, which leads to a general solution,

$$\begin{aligned} x &= e^{-0.1t}\left(C_1 \sin(2t) + C_2 \cos(2t)\right), \\ y &= e^{-0.1t}\left(C_1 \cos(2t) - C_2 \sin(2t)\right), \end{aligned} \tag{37}$$

with C_1 and C_2 constant coefficients determined by the initial points. The trajectory is a harmonic motion at a frequency 2, and the amplitude decays at a rate $e^{-0.1t}$. In the case of noisy measurement data, it is difficult to capture the correct decay rate.



(a) Attracting sets vs. parameter r (b) Zoom in $r \in [3.615, 3.63]$ (c) Attracting sets for different prediction models

FIG. 8. Attracting sets of the logistic map, predictions of SINDy and MSA: (a) attracting sets of the logistic map vs r; (b) zoom in $r \in [3.615, 3.630]$; (c) attracting sets of different prediction models. The attracting sets of MSA are consistent with those of the real systems, whereas SINDy gives a different dynamical behavior.

TABLE IV. Accuracy comparison of SINDy, SINDy with denoising, and MSA under different noise levels. We use a Gaussian-weighted moving average filter to smooth the data. In the STLSQ method, we set $\lambda = 0.07$ for SINDy.

	SINDy	SINDy (smooth, $\lambda = 0.04$)	MSA	Reference
1% noise $\Delta t = 0.001$	$\dot{x} = -0.101x + 2.001y$ $\dot{y} = -2.001x - 0.100y$	$\dot{x} = -0.101x + 1.995y$ $\dot{y} = -1.995x - 0.099y$	$\dot{x} = -0.100x + 2.000y$ $\dot{y} = -2.000x - 0.100y$	$\dot{x} = -0.1x + 2y$ $\dot{y} = -2x - 0.1y$
10% noise	$\dot{x} = 1.236 - 0.233x^2 + 2.659y$ $- 0.114x^2y - 0.942y^2 + 0.106x^2y^2$ $- 0.157y^3 + 0.137y^4$	$\dot{x} = -0.101x + 2.001y$	$\dot{x} = -0.101x + 2.001y$	
$\Delta t = 0.001$	$\dot{y} = -2.032x - 0.257y - 0.266xy^2 + 0.118x^3y^2$	$\dot{y} = -1.986x - 0.093y$	$\dot{y} = -1.998x - 0.101y$	
50% noise	$\dot{x} = -1.978 - 0.699x + 0.151x^3 + 1.689y$ $+ 0.207xy - 0.390x^2y + 0.920y^2$ $- 0.076x^2y^2 + 0.087x^2y^3$	$\dot{x} = -0.279x + 2.110y + 0.082x^3$ $- 0.096x^2y - 0.047y^3$	$\dot{x} = -0.119x + 1.990y$	
$\Delta t = 0.001$	$\dot{y} = -3.794 - 1.946x + 1.713x^2 - 0.144x^4$ $- 0.449y + 0.126x^2y + 1.156y^2$ $- 0.129y^4$	$\dot{y} = -2.072x - 0.170y + 0.095x^3$ $- 0.082x^2y + 0.043y^3$	$\dot{y} = -2.012x - 0.112y$	

SINDy and MSA reconstruct the system using the following approximate equations:

$$\begin{cases} \dot{x} = \boldsymbol{\Theta}(x, y)\boldsymbol{\xi}_x, \\ \dot{y} = \boldsymbol{\Theta}(x, y)\boldsymbol{\xi}_y. \end{cases} \tag{38}$$

with coefficient vectors $\boldsymbol{\xi}_x$ and $\boldsymbol{\xi}_y$ unknown. The library consists of the product of 1D polynomials up to the fifth order, i.e.,

$$\boldsymbol{\Theta}(x, y) = [1, x, x^2, \ldots, x^5, y, xy, \ldots, x^5y^5]. \tag{39}$$

The exact coefficients are

$$\begin{aligned} \boldsymbol{\xi}_x^{real} &= [0, -0.1, 0, 0, 0, 0, 2, 0, \ldots, 0]^T, \\ \boldsymbol{\xi}_y^{real} &= [0, -2, 0, 0, 0, 0, -0.1, 0, \ldots, 0]^T. \end{aligned} \tag{40}$$

First, we present a comparison between the proposed method and SINDy under different noise levels at a small time step size $\Delta t = 0.001$. We use 1000 200-step time series to identify the coefficients. The results are shown in Table IV. As expected, MSA improves the prediction accuracy efficiently and is impressively robust for the noise compared with SINDy. Even if a large noise ($\eta = 50\%$), the present method remains a good accuracy with an error of the parameters less than 10^{-2}. We also present SINDy results with a Gaussian-weighted moving average filter to smooth the data. As shown in the table, although the denoising technique improves the performance of SINDy, SINDy still gives additional incorrect terms for data with 50% noises. As a comparison, MSA remains the correct selection of candidate terms for large noises and gives more accurate results for small noises.

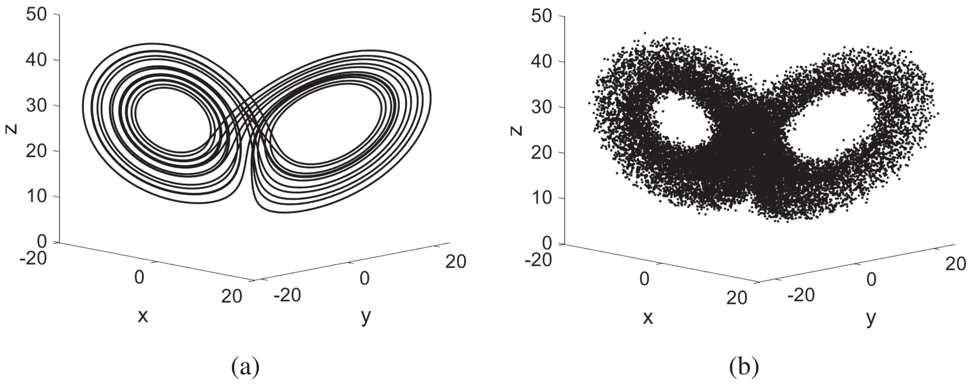
Next, we test our method with different time step sizes. As shown in Table V, for a large time step size ($\Delta t = 0.5$), SINDy fails to capture dynamics accurately with the limitation of a finite difference scheme, whereas MSA gives better results. Furthermore, we put a robust version of SINDy, namely, RK4-SINDy,[33] combining SINDy with RK4 time integration scheme. For small noises, RK4-SINDy gives the same results with MSA, because the deviation mainly comes from the time integration scheme for a large time step. Note that RK4-SINDy is still one-step model, so it induces additional high-order terms in the first equation when facing large noises ($\eta = 50\%$ noise), although the results seem better than those of the classical SINDy. The quality of time integration scheme affects the performance of MSA for a large time step size. To demonstrate this point, we test the MSA with a simple forward Euler scheme in Table VI. As the step size increases, the method fails to keep highly accurate predictions. Thus a better scheme is expected to further improve the performance of the MSA method.

TABLE V. Accuracy comparison of SINDy, RK4-SINDy,[33] and MSA under different noise levels at a large time step size. In the STLSQ method, we set $\lambda = 0.04$.

	SINDy	RK4-SINDy[33]	MSA	Reference
1% noise $\Delta t = 0.5$	$\dot{x} = -0.054x + 1.685y$ $\dot{y} = -1.685x - 0.054y$	$\dot{x} = -0.090x + 2.015y$ $\dot{y} = -2.015x - 0.090y$	$\dot{x} = -0.090x + 2.015y$ $\dot{y} = -2.015x - 0.090y$	$\dot{x} = -0.1x + 2y$ $\dot{y} = -2x - 0.1y$
50% noise	$\dot{x} = 0.943y + 0.206x^2y + 0.175y^3$	$\dot{x} = -0.721x + 1.997y + 0.056x^2y$ $+ 0.425xy^2 + 0.050y^3$	$\dot{x} = -0.087x + 2.013y$	
$\Delta t = 0.5$	$\dot{y} = -0.939x - 0.044y - 0.178x^3$ $- 0.206xy^2$	$\dot{y} = -1.971x - 0.553y$	$\dot{y} = -2.016x - 0.092y$	

TABLE VI. Accuracy comparison of MSA with different time integration schemes.

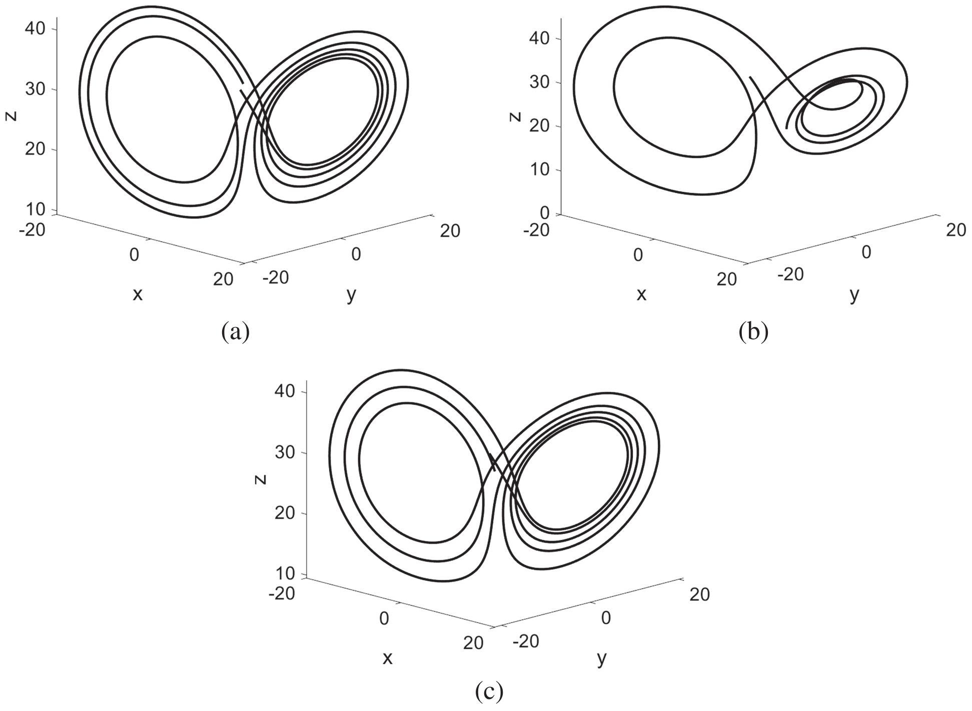
	MSA (Forward Euler)	MSA (RK4)	Reference
1% noise $\Delta t = 0.001$	$\dot{x} = -0.101x + 2.001y$ $\dot{y} = -2.001x - 0.100y$	$\dot{x} = -0.100x + 2.000y$ $\dot{y} = -2.000x - 0.100y$	$\dot{x} = -0.1x + 2y$ $\dot{y} = -2x - 0.1y$
1% noise $\Delta t = 0.01$	$\dot{x} = -0.120x + 1.998y$ $\dot{y} = -1.998x - 0.120y$	$\dot{x} = -0.100x + 2.000y$ $\dot{y} = -2.000x - 0.100y$	
1% noise $\Delta t = 0.1$	$\dot{x} = -0.297x + 1.967y$ $\dot{y} = -1.967x - 0.297y$	$\dot{x} = -0.100x + 2.000y$ $\dot{y} = -2.000x - 0.100y$	



(a)

(b)

FIG. 10. Illustration for the Lorenz system: (a) the trajectory of the Lorenz system; (b) noisy measurements of the Lorenz system with $\eta = 10\%$ noise. Data are collected by solving the dynamical systems using ODE45 solver in Matlab with an initial point $(x(0), y(0), z(0)) = (-8, 7, 27)$. The right subplot shows measurements with $\eta = 10\%$ noise. (a) The trajectory of the Lorenz system. (b) Noisy measurements of the Lorenz system.



(a)

(b)

(c)

FIG. 11. Trajectories of real systems, predictions of SINDy and MSA for data with 50% noise. The initial point is $(x(0), y(0), z(0)) = (-8, 7, 27)$. (a) Exact solution, (b) SINDy, (c) MSA.

D. Lorenz system

Now, we consider the Lorenz system to explore the data-driven identification of chaotic systems. The governing equations are

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta z. \end{cases} \tag{41}$$

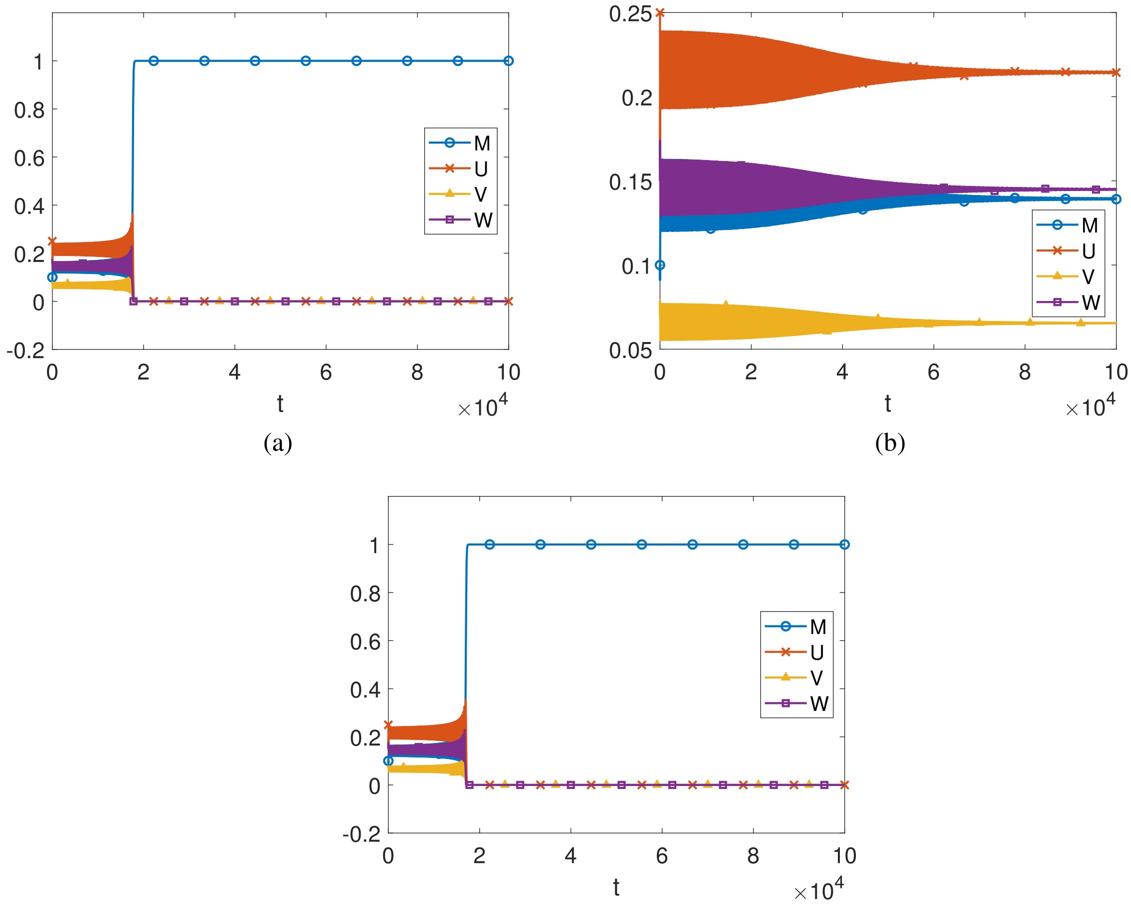
Here, we set the standard parameters as $\sigma = 10, \rho = 28$, and $\beta = \frac{8}{3}$. Data are collected by solving the dynamical systems using ODE45 solver in Matlab. Figure 10 illustrates a trajectory from $t = 0$ to $t = 20$ with an initial condition $(x(0), y(0), z(0)) = (-8, 7, 27)$.

The results by SINDy, ER and MSA under noisy measurements are shown in the supplementary material, with $m = 100$ and $S = 200$. When computing libraries $\boldsymbol{\Theta}$ for SINDy, data are normalized by standard deviations first and then a threshold is set as $\lambda = 0.1$ for sparsity. The library consists of the product of 1D polynomials up to the fifth order (also see the supplementary material). For small noises, all of three methods successfully discover the Lorenz system accurately. However, for large noises, there exist incorrect terms and coefficients in the predictions of SINDy and ER, but the coefficients of MSA are still relatively close to the real ones.

TABLE VII. Coefficients at $\alpha = 1.49$, $\beta = \pi/2$, $\gamma = 1.82$, $R = 182$ in the reduced order model (42).

κ_m	κ_u	κ_v	κ_w	σ_m	σ_u	σ_v	σ_w	R
1.571	1.820	2.404	2.378	0.240	1.189	0.099	0.331	182

The trajectories of predictions of SINDy and MSA for data with 50% noise are plotted in Fig. 11, which directly reflects the prediction accuracy of the governing equations. The numerical results



(a)

(b)

(c)

FIG. 12. Prediction trajectories of different methods with the same initial point $[M(0), U(0), V(0), W(0)] = [0.1, 0.25, 0.07, 0.15]$: (a) the exact solution; (b) the prediction of SINDy; (c) the prediction of MSA. The exact solution and our method both reduce to the laminar solution ($[M, U, V, W] = [1, 0, 0, 0]$), while the SINDy prediction converges to another steady solution ($[M, U, V, W] \approx [0.139, 0.215, 0.065, 0.145]$). (a) Exact solution, (b) SINDy, (c) MSA.

clearly show the robustness of MSA for large noises and verify the effectiveness of the proposed method for the Lorenz system.

E. Reduced order model of a self-sustaining process in turbulent shear flows

Shear flows are a fundamental class of fluid flows. Waleffe[38–40] has studied wall-bounded shear flows based on the concept of the self-sustaining process, where streamwise rolls redistribute the mean shear to create streaks that break down to recreate the rolls. He derived a reduced order model of the process from the Navier–Stokes equations for a sinusoidal shear flow. The reduced order model is a four-dimensional nonlinear system, given by[40]

$$\begin{cases} \left(\frac{d}{dt} + \frac{\kappa_m^2}{R}\right) M = \sigma_m W^2 - \sigma_u UV + \frac{\kappa_m^2}{R}, \\ \left(\frac{d}{dt} + \frac{\kappa_u^2}{R}\right) U = -\sigma_w W^2 + \sigma_u MV, \\ \left(\frac{d}{dt} + \frac{\kappa_v^2}{R}\right) V = \sigma_v W^2, \\ \left(\frac{d}{dt} + \frac{\kappa_w^2}{R}\right) W = \sigma_w UW - \sigma_m MW - \sigma_v VW, \end{cases} \tag{42}$$

with the amplitudes of the mean shear M, the streaks U, the rolls V, and the streak eigenmode W. The W (fourth) equation describes the instability of the streaky flow. All the coefficients are positive, defined by

$$\kappa_m = \beta, \quad \kappa_u = \gamma, \kappa_v = \sqrt{\beta^2 + \gamma^2}, \quad \kappa_w = \sqrt{(2\gamma^2 + \beta^2 + \alpha^2)/2},$$

$$\sigma_m = \frac{\alpha\beta}{2}\sqrt{\frac{\gamma^2 - \alpha^2}{(\alpha^2 + \beta^2)(\alpha^2 + \gamma^2)}}, \quad \sigma_u = \frac{\beta\gamma}{\sqrt{\beta^2 + \gamma^2}},$$

$$\sigma_v = \frac{\alpha^2(\gamma^2 - \beta^2)}{2\gamma\sqrt{(\alpha^2 + \beta^2)(\beta^2 + \gamma^2)}}, \quad \sigma_w = \frac{\alpha}{2}\sqrt{\frac{\gamma^2 - \alpha^2}{\alpha^2 + \gamma^2}},$$

depending on three parameters α, β, γ with $\gamma^2 - \alpha^2 > 0$, $\gamma^2 - \beta^2 > 0$, and R is the Reynolds number. For details, please refer to Ref. 40.

There exists a critical Reynolds number R_{sn} (e.g., $R_{sn} = 137.17$ with $\alpha = 1.49, \beta = \pi/2, \gamma = 1.82$) for the reduced order model where a saddle-node bifurcation introduces two new steady solutions in addition to the laminar solution ($[M, U, V, W] = [1, 0, 0, 0]$). Both solutions are typically unstable, with the lower branch corresponding to a saddle point with a single positive real eigenvalue, and the upper branch corresponding to an unstable node with two real positive eigenvalues. As R increases, the upper branch becomes a stable node (e.g., at $R \approx 180$ with $\alpha = 1.49, \beta = \pi/2, \gamma = 1.82$), which implies a Hopf bifurcation.

We take $\alpha = 1.49, \beta = \pi/2, \gamma = 1.82, R = 182$, and the corresponding coefficients are listed in Table VII. The system is solved by Matlab ODE45 solver. One thousand 200-step time series are sampled with a time step size $\Delta t = 0.5$ and imposed by a 1% noise. We take the library as the polynomials up to the third order. The threshold for sparsity is $\lambda = 0.01$. The identified equations are given in the supplementary material, and the corresponding prediction trajectories with the initial point $[M(0), U(0), V(0), W(0)] = [0.1, 0.25, 0.07, 0.15]$ are shown in Fig. 12. Both the exact solution and that by our method reduce to the laminar solution ($[M, U, V, W] = [1, 0, 0, 0]$) after a long-time evolution. By SINDy results, the trajectory converges to another steady solution ($[M, U, V, W] \approx [0.139, 0.215, 0.065, 0.145]$). This comparison shows that the accuracy and robustness are crucial for predictions around the bifurcation points when facing noisy measurements.

IV. CONCLUSION

In this paper, we propose a sparse identification method considering multi-step error accumulation to discover governing equations from noisy measurement data, called the Multi-Step-Accumulation (MSA) method. The key idea is to use *multi-step* models instead of *single-step* models (such as SINDy) and to identify the parameters accurately by minimizing the total error of measured series and approximate ones. The accumulated errors for the previous time steps are used to reconstruct dynamical systems, since the dynamical states may not only depend on the nearest previous states, and, thus, one-time-step error is insufficient to account for the history-dependent behaviors. On the one hand, in the proposed method, controlling the error accumulation enhances the accuracy of the predictions, especially for the chaotic systems whose behaviors are sensitive to the parameters. On the other hand, MSA combines evolution schemes and, thus, captures the dynamics directly from the noisy measurements, resisting the corruption of noise. As a result, the proposed method can be used to identify the dominant terms which cannot be identified by the use of SINDy, especially for large noises and discover chaotic systems at bifurcation points from noisy measurement data.

MSA is numerically shown to be robust for high accuracy predictions and successfully discovers the chaotic systems around the bifurcation points from the noisy measurements. The test cases include a discrete chaotic map, the logistic map, a damped oscillator, the Lorenz system and a reduced order model of a self-sustaining process. These examples verify the good robustness and accuracy of the proposed method for the datasets with large noises. Compared with conventional methods, MSA remains a good selection of model terms for large noises and realizes highly accurate predictions of parameters for small noises. The last three examples show that the strategy combined with Runge–Kutta scheme is still valid for snapshots with a large time step size. Furthermore, the examples of the logistic map and a reduced order model of a self-sustaining process clearly show that the prediction accuracy significantly influences the dynamics around the bifurcation points. In these two examples, our method successfully captures the correct dynamical behaviors from noisy measurement data.

To resolve the difficulty of optimization of MSA, we propose an adaptive training strategy. The adaptive strategy uses the solution for small time steps as the initial estimation to optimize with large ones and, thus, gradually increases the length of time series for training. Consequently, this strategy provides the opportunities for complex nonlinear optimization with long time series and makes the implementations of MSA not limited to the length of series used. Furthermore, numerical examples present a significant improvement in accuracy with increasing length of time series used by this adaptive strategy.

MSA is possibly extended to the partial differential equations (PDEs), while it is used for the discrete maps and continuous dynamical systems in the present study. It is noted that the discovery of PDEs requires a proper solution scheme and an efficient optimization algorithm. The primary limitation of the extensions of MSA lies in the necessity of an explicit time integration scheme. The quality of the time integration scheme directly influences the performance. In this paper, we choose the Runge–Kutta scheme due to its good performance. We may borrow the idea of multistep neural networks,[12] i.e., using the multi-step time-stepping scheme to develop the present method and expect it to provide better performance. In addition, the training data used in this paper are from the known systems, and the implementations of this method on experimental and real systems should be developed in future research.

SUPPLEMENTARY MATERIAL

See the supplementary material for more details on the libraries and additional numerical results for a chaotic map, the Lorenz system, and a reduced order model of a self-sustaining process in turbulent shear flows.

ACKNOWLEDGMENTS

L. Zhang, S. Tang, and G. He were supported by the National Natural Science Foundation of China (NSFC) Basic Science Center Program for “Multiscale Problems in Nonlinear Mechanics” (Grant No. 11988102). L. Zhang was also supported by the National Natural Science Foundation of China (NSFC) under Grant No. 12202451.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Lei Zhang: Methodology (equal); Software (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Shaoqiang Tang:** Supervision (equal); Writing – review & editing (equal). **Guowei He:** Funding acquisition (lead); Supervision (lead); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX: CALCULATION OF DERIVATIVES WITH RESPECT TO COEFFICIENTS ξ

We can borrow the idea of backward-propagation algorithm to compute the derivatives of the loss function with respect to coefficients $\boldsymbol{\xi}$ efficiently.

The derivatives are derived recursively from (5) and (7) according to the chain rule, namely,

$$\frac{\partial loss}{\partial \xi_l} = \frac{2}{S-1}\sum_{k=2}^{S}(o_k - x_k)\frac{\partial o_k}{\partial \xi_l}, \tag{A1}$$

$$\frac{\partial o_k}{\partial \xi_l} = \Theta_l(o_{k-1}) + \frac{d\boldsymbol{\Theta}(o_{k-1})}{do_{k-1}}\boldsymbol{\xi}\frac{\partial o_{k-1}}{\partial \xi_l}, \tag{A2}$$

where Θ_l is the lth term in the library $\boldsymbol{\Theta}$ corresponding to the lth term ξ_l of the coefficient vector $\boldsymbol{\xi}$, and $\frac{d\boldsymbol{\Theta}(x)}{dx}$ is the derivatives of the candidate functions in the library, e.g.,

$$\frac{d\boldsymbol{\Theta}(x)}{dx} = [0, 1, 2x, 3x^2, \ldots, px^{p-1}, \cos(x), -\sin(x), \ldots, \exp(x), \ldots], \tag{A3}$$

corresponding to library (4).

Similar to backward-propagation algorithm, the calculation of derivatives consists of two steps: a forward step and a backward step. In a forward step, the series o_k and corresponding $\boldsymbol{\Theta}(o_k)$ are obtained according to evolution (5). In a backward step, we derive $\frac{\partial o_k}{\partial \xi_l}$ from the recursive formulation (A2) first and then assemble the derivatives $\frac{\partial loss}{\partial \xi_l}$ according to (A1).

The complexity is linear with the number of derivatives, i.e., proportional to NdL, with the number of data points $N = m \times S$, the dimensions d, and the number of active candidate functions L. Note that L might be small after a group sparsity approach such as STLSQ algorithm applied to the SINDy or the training with short time series.

REFERENCES

[1] P. Perdikaris and S. Tang, “Mechanistic machine learning: Theory, methods, and applications,” Theor. Appl. Mech. Lett. **10**, 141–142 (2020).

[2] L. Ljung, “Perspectives on system identification,” Annu. Rev. Control **34**(1), 1–12 (2010).

[3] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes* (Springer Nature, 2020).

[4] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” Proc. Natl. Acad. Sci. U.S.A. **113**(15), 3932–3937 (2016).

[5] S. A. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains* (John Wiley & Sons, 2013).

[6] Z. Wang, B. Wu, K. Garikipati, and X. Huan, “A perspective on regression and Bayesian approaches for system identification of pattern formation dynamics,” Theor. Appl. Mech. Lett. **10**(3), 188–194 (2020).

[7] P. J. Schmid, “Dynamic mode decomposition of numerical and experimental data,” J. Fluid Mech. **656**, 5–28 (2010).

[8] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (SIAM, 2016).

[9] Q. Li, F. Dietrich, E. M. Bollt, and I. G. Kevrekidis, “Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator,” Chaos **27**(10), 103111 (2017).

[10] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach,” Phys. Rev. Lett. **120**(2), 024102 (2018).

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “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 (2019).

[12] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," arXiv:1801.01236.

[13] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos, "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," Proc. R. Soc. A **474**(2213), 20170844 (2018).

[14] C. Wehmeyer and F. Noé, "Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics," J. Chem. Phys. **148**(24), 241703 (2018).

[15] S. Li and Y. Yang, "A recurrent neural network framework with an adaptive training strategy for long-time predictive modeling of nonlinear dynamical systems," J. Sound Vib. **506**, 116167 (2021).

[16] J. J. Bramburger, D. Dylewsky, and J. N. Kutz, "Sparse identification of slow timescale dynamics," Phys. Rev. E **102**(2), 022204 (2020).

[17] J. Horrocks and C. T. Bauch, "Algorithmic discovery of dynamic models from infectious disease data," Sci. Rep. **10**(1), 1–18 (2020).

[18] S. Li, E. Kaiser, S. Laima, H. Li, S. L. Brunton, and J. N. Kutz, "Discovering time-varying aerodynamics of a prototype bridge by sparse identification of nonlinear dynamical systems," Phys. Rev. E **100**(2), 022220 (2019).

[19] M. Schmelzer, R. P. Dwight, and P. Cinnella, "Discovery of algebraic Reynolds-stress models using sparse symbolic regression," Flow Turbul. Combust. **104**(2), 579–603 (2020).

[20] J. J. Bramburger and J. N. Kutz, "Poincaré maps for multiscale physics discovery and nonlinear Floquet theory," Physica D **408**, 132479 (2020).

[21] M. Quade, M. Abel, J. Nathan Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for rapid model recovery," Chaos **28**(6), 063116 (2018).

[22] B. Bhadriraju, M. S. F. Bangi, A. Narasingam, and J. S.-I. Kwon, "Operable adaptive sparse identification of systems: Application to chemical processes," AIChE J. **66**(11), e16980 (2020).

[23] M. Hoffmann, C. Fröhner, and F. Noé, "Reactive SINDy: Discovering governing reactions from concentration data," J. Chem. Phys. **150**(2), 025101 (2019).

[24] H. Xu, H. Chang, and D. Zhang, "DL-PDE: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data," Commun. Comput. Phys. **29**(3), 698–728 (2021).

[25] H. Xu, H. Chang, and D. Zhang, "DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm," J. Comput. Phys. **418**, 109584 (2020).

[26] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," Adv. Neural Inf. Process. Syst. **31**, 6571–6583 (2018).

[27] T. Z. Jiahao, M. A. Hsieh, and E. Forgoston, "Knowledge-based learning of nonlinear dynamics and chaos," Chaos **31**(11), 111101 (2021).

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

[29] R. Tibshirani, "Regression shrinkage and selection via the lasso," J. R. Stat. Soc. Ser. B **58**(1), 267–288 (1996).

[30] P. Zheng, T. Askham, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin, "A unified framework for sparse relaxed regularized regression: SR3," IEEE Access **7**, 1404–1423 (2018).

[31] L. Boninsegna, F. Nüske, and C. Clementi, "Sparse learning of stochastic dynamical equations," J. Chem. Phys. **148**(24), 241723 (2018).

[32] W. Pan, Y. Yuan, J. Gonçalves, and G.-B. Stan, "A sparse Bayesian approach to the identification of nonlinear state-space systems," IEEE Trans. Autom. Control **61**(1), 182–187 (2015).

[33] P. Goyal and P. Benner, "Discovery of nonlinear dynamical systems using a Runge-Kutta inspired dictionary-based sparse regression approach," Proc. R. Soc. A **478**(2262), 20210883 (2022).

[34] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, and C. Grebogi, "Predicting catastrophes in nonlinear dynamical systems by compressive sensing," Phys. Rev. Lett. **106**(15), 154101 (2011).

[35] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," Sci. Adv. **3**(4), e1602614 (2017).

[36] S. Ushiki, "Discrete population models and chaos," Lecture Notes Num. App. Anal. **2**, 1–25 (1980).

[37] A. A. R. AlMomani, J. Sun, and E. Bollt, "How entropic regression beats the outliers problem in nonlinear system identification," Chaos **30**(1), 013107 (2020).

[38] F. Waleffe, "Hydrodynamic stability and turbulence: Beyond transients to a self-sustaining process," Stud. Appl. Math. **95**(3), 319–343 (1995).

[39] F. Waleffe, "Transition in shear flows. Nonlinear normality versus non-normal linearity," Phys. Fluids **7**(12), 3060–3066 (1995).

[40] F. Waleffe, "On a self-sustaining process in shear flows," Phys. Fluids **9**(4), 883–900 (1997).