SIAM J. SCI. COMPUT.
Vol. 44, No. 3, pp. A1145–A1175

ROBUST IDENTIFICATION OF DIFFERENTIAL EQUATIONS BY NUMERICAL TECHNIQUES FROM A SINGLE SET OF NOISY OBSERVATION*

YUCHEN HE†, SUNG-HA KANG‡, WENJING LIAO‡, HAO LIU§, AND YINGJIE LIU‡

Abstract. We propose robust methods to identify the underlying Partial Differential Equation (PDE) from a given single set of noisy time-dependent data. We assume that the governing equation of the PDE is a linear combination of a few linear and nonlinear differential terms in a prescribed dictionary. Noisy data make such identification particularly challenging. Our objective is to develop robust methods against a high level of noise and approximate the underlying noise-free dynamics well. We first introduce a Successively Denoised Differentiation (SDD) scheme to stabilize the amplified noise in numerical differentiation. SDD effectively denoises the given data and the corresponding derivatives. Second, we present two algorithms for PDE identification: Subspace pursuit Time evolution (ST) error and Subspace pursuit Cross-validation (SC). Our general strategy is to first find a candidate set using the Subspace Pursuit (SP) greedy algorithm, then choose the best one via time evolution or cross-validation. ST uses a multishooting numerical time evolution and selects the PDE which yields the least evolution error. SC evaluates the cross-validation error in the least-squares fitting and picks the PDE that gives the smallest validation error. We present various numerical experiments to validate our methods. Both methods are efficient and robust to noise.

Key words. inverse problem, PDE identification, noisy data

AMS subject classifications. 35R30, 65Z05, 65M32

DOI. 10.1137/20M134513X

1. Introduction.

Partial Differential Equations (PDEs) are used to model various real-world phenomena in science and engineering. Numerical solvers for PDEs and analysis of various properties of the solutions have been widely studied in the literature. In this paper, we focus on the inverse problem: Given a single set of time-dependent noisy data, how does one identify the underlying PDE?

Let the given noisy time-dependent discrete data set be

$$\mathbf{D} := \{U_{\mathbf{i}}^n \in \mathbb{R} \mid n = 0, \dots, N; \mathbf{i} = (i_1, \dots, i_d) \text{ with } i_j = 0, \dots, M-1, j = 1, \dots, d\} \tag{1.1}$$

for sufficiently large integers $N, M \in \mathbb{N}$, where $\mathbf{i}$ is a d-dimensional spatial index of a discretized domain in $\mathbb{R}^d$, and n represents the time index at time t^n. The objective is to find an evolutionary PDE of the form

$$\partial_t u = f(u, \partial_{\mathbf{x}} u, \partial_{\mathbf{x}}^2 u, \dots, \partial_{\mathbf{x}}^k u, \dots) \,, \tag{1.2}$$

which represents the dynamics of the given data $\mathbf{D}$. Here t is the time variable, $\mathbf{x} = [x_1, \dots, x_d] \in \mathbb{R}^d$ denotes the space variable, and $\partial_{\mathbf{x}}^k u$ denotes the set of partial

*Submitted to the journal's Methods and Algorithms for Scientific Computing section June 12, 2020; accepted for publication (in revised form) November 22, 2021; published electronically May 5, 2022.

https://doi.org/10.1137/20M134513X

Funding: The work of the second author was partially supported by Simons Foundation grants 282311 and 584960. The work of the third author was partially supported by NSF grants DMS 1818751 and DMS 2012652. The work of the fifth author was partially supported by NSF grants DMS-1522585 and DMS-CDS&E-MSS-1622453.

†Institute of Natural Sciences, Shanghai Jiao Tong University, Shanghai, 200240 China (yuchenroy@sjtu.edu.cn).

‡School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0160 USA (kang@math.gatech.edu, wliao60@gatech.edu, yingjie@math.gatech.edu).

§Department of Mathematics, Hong Kong Baptist University, Hong Kong (haoliu@hkbu.edu.hk).

derivatives of u with respect to the space variable of order k for $k = 0, 1, \ldots$, i.e., $\partial^k_{\mathbf{x}} u := \left\{ \frac{\partial^k u}{\partial x_1^{k_1} \partial x_2^{k_2} \cdots \partial x_d^{k_d}} \mid k_1, \ldots, k_d \in \mathbb{N},\ \sum_{j=1}^d k_j = k \right\}$. We assume that f is a polynomial of its arguments so that the right-hand side of (1.2) is a linear combination of linear and nonlinear differential terms. The model in (1.2) includes a class of parametric PDEs where the parameters are the polynomial coefficients in f.

Parameter identification in differential equations and dynamical systems has been considered by physicists and applied scientists. Earlier works include [1, 2, 3, 4, 28, 29, 30], among which, [2, 28] considered the PDE model as in (1.2). Two important papers [5, 38] used symbolic regression to recover the underlying physical systems from experimental data. Recently, sparse regression and L_1-minimization were introduced to promote sparsity in the identification of PDEs or dynamical systems [7, 17, 34, 35]. In [7], Brunton, Proctor, and Kutz considered the discovery of nonlinear dynamical systems with sparsity-promoting techniques. The underlying dynamical systems are assumed to be governed by a small number of active terms in a prescribed dictionary, and sparse regression is used to identify these active terms. The extensions of this sparse regression approach can be found in [16, 21, 26]. In [35], Schaeffer considered the problem of PDE identification using the spectral method and focused on the benefit of using L_1-minimization for sparse coefficient recovery. The identification of dynamical systems with highly corrupted and undersampled data is considered in [37, 43]. In [34], Rudy et al. proposed identifying PDEs by solving the L_0-regularized regression problem followed by a postprocessing step of thresholding. Sparse Bayesian regression was considered in [49] for the recovery of dynamical systems. This series of work focused on the benefit of using L_1-minimization to resolve dynamical systems or PDEs with specific sparse pattern [36]. In Appendix A, we compare some existing methods in terms of the objectives in minimization. Recent works such as [13, 27] introduced PDE learning in a weak formulation to ameliorate the errors due to the instability of numerical differentiation, when the given data are contaminated by noise. This weak formulation gives rise to a robust recovery, while it requires the underlying PDE to possess a weak formulation such that all partial derivatives in the PDE can be transferred to a test function through integration by parts. Another related problem is to infer the interaction law in a system of agents from the trajectory data. In [6, 24], nonparametric regression was used to predict the interaction function, and a theoretical guarantee was established. Another category of methods uses deep learning [18, 22, 23, 25, 31, 32, 33].

The most closely related work to this paper is [17], where Identifying Differential Equation with Numerical Time evolution (IDENT) was proposed, also for a single set of given data. It is based on the convergence principle of numerical PDE schemes. LASSO is used to find a candidate set efficiently, and the correct PDE is identified by computing the numerical Time Evolution Error (TEE). Among all the PDEs from the candidate set, the one whose numerical solution best matches the given data dynamics is chosen as the identified PDE. When the given data are contaminated by noise, the authors used a Least-Squares Moving Average method to denoise the data as a preprocessing step. When the coefficients vary in the spatial domain, a Base Element Expansion (BEE) technique was proposed to recover the varying coefficients.

Despite the developments of many useful methods, when the given data are noisy, PDE identification is still challenging. A small amount of noise can make a recovery unstable, especially for high order PDEs. It was shown in [17] that the noise-to-signal ratio for LASSO depends on the order of the underlying PDE, and IDENT can handle a small amount of noise when the PDE contains high order derivatives. A significant

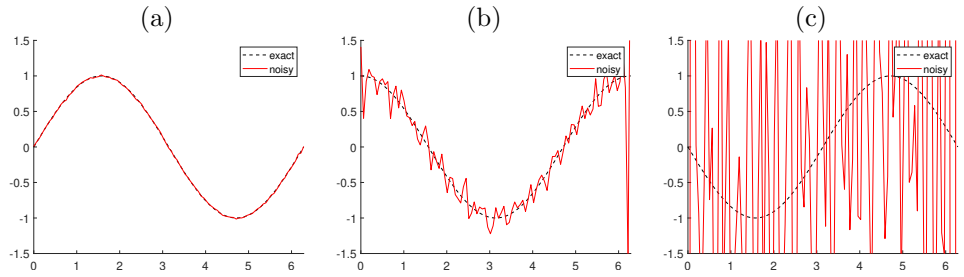

FIG. 1. *The sensitivity of numerical differentiation to noise.* (a) *Graph of* $\sin(x)$, $0 \le x \le 2\pi$ *(black), and its noisy version (red) with Gaussian noise of mean* 0 *and standard deviation* 0.01. (b) *The first order derivatives of the function (black) and the data (red).* (c) *The second order derivatives of the function (black) and the data (red). The derivatives of data in* (b) *and* (c) *are computed using the five-point ENO scheme. As the order of derivative increases, the noise gets amplified. (Figure in color online.)*

issue is that the numerical differentiation often magnifies noise, which is illustrated by an example in Figure 1.

In this paper, we propose two robust PDE identification methods that can handle a large amount of noise given a single set of time-dependent data. Our contributions include the following:

1. First, we propose a new denoising procedure, called Successively Denoised Differentiation (SDD), to stabilize the numerical differentiation applied to noisy data.
2. Second, we present two recovery algorithms which are robust against noise: Subspace pursuit Time evolution (ST) and Subspace pursuit Cross-validation (SC). Both methods utilize the Subspace Pursuit (SP) greedy algorithm [10] for selecting a candidate set. ST considers a multishooting numerical time evolution error, and SC evaluates the cross-validation error in the least-squares fitting. Both methods are efficient and robust against noise.

This paper is organized as follows. In section 2, we introduce the PDE identification problem and describe the SDD scheme. Our proposed ST and SC algorithms are presented in section 3, and systematic numerical experiments are provided in section 4. We conclude the paper in section 5, and some details are discussed in the appendices.

2. Data organization and denoising.

2.1. Data organization and notation. Let the time-space domain be $\Omega = [0,T] \times [0,X]^d$ for some $T > 0$ and $X > 0$. Suppose the noisy data $\mathbf{D}$ are given as (1.1) on a regular grid in Ω, with time index $n = 0, \dots, N$, $N \in \mathbb{N}$, and spatial index $\mathbf{i} \in \mathbb{I}$, where $\mathbb{I} = \{(i_1, \dots, i_d) \mid i_j = 0, \dots, M-1,\ j = 1, \dots, d, M \in \mathbb{N}\}$. Denote $\Delta t := T/N$ and $\Delta x := X/(M-1)$ as the time and space spacing in the given data, respectively.

At the time t^n and the location $x_{\mathbf{i}}$, the datum is given as

$$U_{\mathbf{i}}^n = u(\mathbf{x}_{\mathbf{i}}, t^n) + \varepsilon_{\mathbf{i}}^n \,, \tag{2.1}$$

where $t^n := n\Delta t \in [0,T]$, $\mathbf{x}_{\mathbf{i}} := (i_1 \Delta x, \dots, i_d \Delta x) \in [0,X]^d$, and $\varepsilon_{\mathbf{i}}^n$ is i.i.d. random noise with mean 0. For $n = 0, 1, \dots, N-1$, we vectorize the data in all spatial domains at time t_n, and denote it as $U^n \in \mathbb{R}^{M^d}$. Concatenating the vectors $\{U^n\}_{n=0}^{N-1}$ vertically gives rise to a long vector $U \in \mathbb{R}^{NM^d}$.

The underlying function f in (1.2) is assumed to be a finite order polynomial of its arguments:

$$f(u, \partial_{\mathbf{x}} u, \partial_{\mathbf{x}}^2 u, \ldots, \partial_{\mathbf{x}}^k u, \ldots) = c_1 + c_2 \partial_{x_1} u + \cdots + c_m u \partial_{x_1} u + \cdots , \tag{2.2}$$

where $\partial_{\mathbf{x}}^k$ denotes all kth order partial derivatives and ∂_{x_j} denotes the partial derivative with respect to the jth variable. We refer to each term, such as $1, \partial_{x_1} u$, and $u\partial_{x_1} u, \ldots$, in (2.2), as a *feature*. Since f is a finite order polynomial, only a finite number of features are included. Denote the number of features by K. Under this model, the function f is expressed in a parametric form as a linear combination of K features. Our objective is to recover the parameters, or coefficients,

$$\mathbf{c} = [c_1 \;\; c_2 \;\; \ldots \;\; c_m \;\; \ldots \;\; c_K]^T \in \mathbb{R}^K,$$

where many of the entries may be zero.

From $\mathbf{D}$, we numerically approximate the time and spatial derivatives of u to obtain the following *approximated time derivative vector* $D_t U \in \mathbb{R}^{NM^d}$ and *approximated feature matrix* $F \in \mathbb{R}^{NM^d \times K}$:

$$D_t U = \begin{bmatrix} \frac{U^1 - U^0}{\Delta t} \\ \frac{U^2 - U^1}{\Delta t} \\ \vdots \\ \frac{U^N - U^{N-1}}{\Delta t} \end{bmatrix}, \quad F = \begin{bmatrix} \mathbf{1}_{M^d \times 1} & U^0 & \cdots & U^0 \circ D_{x_1} U^0 & \cdots \\ \mathbf{1}_{M^d \times 1} & U^1 & \cdots & U^1 \circ D_{x_1} U^1 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \cdots \\ \mathbf{1}_{M^d \times 1} & U^{N-1} & \cdots & U^{N-1} \circ D_{x_1} U^{N-1} & \cdots \end{bmatrix}. \tag{2.3}$$

In this paper, the time derivatives in $D_t U$ are approximated by the forward difference scheme, and the spatial derivatives, such as $D_{x_1} U^n$ for $n = 0, 1, \ldots, N-1$ in F, are computed using the 5-point essentially nonoscillatory (ENO) scheme [14]. Other numerical differentiation schemes can be used here. (See [17] for an error estimation.) The vector $\mathbf{1}_{M^d \times 1} \in \mathbb{R}^{M^d}$ denotes the 1-vector of size M^d, and the Hadamard product $\circ$ is the elementwise multiplication between two vectors. Each column of F is referred to as a *feature column*. The PDE model in (1.2) suggests that an optimal coefficient vector $\mathbf{c}$ should satisfy the following approximation:

$$D_t U \approx F\mathbf{c} \, . \tag{2.4}$$

The objective of this paper is to find the correct set of coefficients in (2.2). Due to the large size of K, the idea of sparsity becomes useful.

The framework of our methods relies on a prescribed dictionary, and the dictionary should contain all possible terms in the underlying PDE. If we do not have any a priori knowledge, one strategy is to use the pairwise product of the partial derivatives of u up to certain order. One can also view the right-hand side of the target PDE in (1.2) as a functional of u and its partial derivatives up to certain order, and then approximate the right-hand side by a Taylor polynomial up to certain degree. Our method is capable of identifying this Taylor polynomial, as an approximation to the right-hand side of the underlying PDE. Another strategy is to estimate the possible features from the given data, which is an open problem to be studied in the future.

Throughout this paper, we denote F_0 as the true feature matrix whose elements are the exact derivatives evaluated at the corresponding time and space location as

those in F. For a vector $\mathbf{c}$, $\|\mathbf{c}\|_p := (\sum_j |c_j|^p)^{\frac{1}{p}}$ is the L_p norm of $\mathbf{c}$. In particular, $\|\mathbf{c}\|_\infty := \max_j |c_j|$. When $p = 0$, $\|\mathbf{c}\|_0 := \#\{c_j : c_j \neq 0\}$ represents the L_0 seminorm of $\mathbf{c}$. The support of $\mathbf{c}$ is denoted by $\text{supp}(\mathbf{c}) := \{j : c_j \neq 0\}$. The vector $\mathbf{c}$ is said to be k-sparse if $\|\mathbf{c}\|_0 = k$ for a nonnegative integer k. For any matrix $A_{m\times n}$ and index sets $\mathcal{L}_1 \subseteq \{1, 2, \ldots, n\}$, $\mathcal{L}_2 \subseteq \{1, 2, \ldots, m\}$, we denote $[A]_{\mathcal{L}_1}$ as the submatrix of A consisting of the columns indexed by $\mathcal{L}_1$, and $[A]^{\mathcal{L}_2}$ as the submatrix of A consisting of the rows indexed by $\mathcal{L}_2$. A^T, A^*, and $A^\dagger$ denote the transpose, conjugate transpose, and Moore–Penrose pseudoinverse of A, respectively. For $x \in \mathbb{R}$, $\lfloor x \rfloor$ denotes the largest integer no larger than x.

2.2. Successively Denoised Differentiation (SDD). As shown in Figure 1, when the given data are contaminated by noise, numerical differentiation amplifies noise. It introduces a large error in the time derivative vector $D_t U$ and the approximated feature matrix F. With random noise, the regularity of the given data is different from the PDE solution's regularity. Thus, the denoising plays a vital role in PDE identification.

We introduce a smoothing operator S to process the data. Kernel methods are good options for S, such as Moving Average [39] and Moving Least Squares (MLS) [19]. In this paper, the smoothing operator S is chosen as the MLS, where data are locally fit by quadratic polynomials. In MLS, a weighted least-squares problem, in the time domain or the spatial domain, is solved at each time t^n and spatial location $x_{\mathbf{i}}$ as follows:

(2.5)
$$S_{(\mathbf{x})}\,[U]^n_{\mathbf{i}} = p^n_{\mathbf{i}}(\mathbf{x_i}),\ \text{with}\ \ p^n_{\mathbf{i}} = \underset{p\in P_2}{\arg\min} \sum_{\mathbf{j}\in\mathbb{I}} (p(\mathbf{x_j}) - U^n_{\mathbf{j}})^2 \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{h^2}\right),$$

(2.6)
$$S_{(t)}\,[U]^n_{\mathbf{i}} = p^n_{\mathbf{i}}(t^n),\ \text{with}\ \ p^n_{\mathbf{i}} = \underset{p\in P_2}{\arg\min} \sum_{0\le k\le N} (p(t^k) - U^k_{\mathbf{i}})^2 \exp\left(-\frac{\|t^n - t^k\|^2}{h^2}\right).$$

Here $h > 0$ is a width parameter of the kernel, and P_2 denotes the set of polynomials of degree no more than 2. It is shown in [46, Theorem 4.1] that, for a fixed time index n, if the given data $\{U^n_{\mathbf{i}}\}_{\mathbf{i}}$ are sampled from a C^k function $u(\mathbf{x}, t^n)$, and the $(k-1)$th order polynomials are used in MLS, the output of MLS with a proper choice of the kernel width h gives a kth order approximation of $u(\mathbf{x}, t^n)$. This theory demonstrates that MLS keeps the accuracy of the given data when the data contain no noise and the solution is sufficiently smooth. In practice, the width parameter h is found empirically: as the noise level increases, a larger h is used to address the data variability. In our experiments, we observe that the performance of our method is not sensitive to the choice of h and we use the same h for different noise levels.

We propose a Successively Denoised Differentiation (SDD) procedure to stabilize the numerical differentiation. For every derivative approximation, smoothing is applied as described in Table 1.

The main idea of SDD is to smooth the data at each step (before and after) the numerical differentiation. This simple idea effectively stabilizes numerical differentiation. Figure 2 shows the results of SDD for the same data in Figure 1. The approximations of the first and second order derivatives of u are greatly improved by SDD.

When the given data are noiseless and MLS is used in SDD, the following theorem shows that under appropriate assumptions, the estimated partial derivative $S_{(x)}D_x[U]$

TABLE 1

Examples of SDD: Each (differential) term is approximated by the spatial and time smoothing operators $S_{(\mathbf{x})}$ and $S_{(t)}$ defined in (2.5) and (2.6), respectively. The operator D_t given in (2.3) represents numerical time differentiation by the forward difference scheme, and D_{x_j} for $j = 1, \ldots, d$ represents numerical spatial differentiation with respect to x_j given by the 5-point ENO scheme [14].

Term	Approximation
u	$\approx S_{(\mathbf{x})}[U]$ u is approximated by spatially MLS denoised data U.
$\partial_t u$	$\approx S_{(t)} D_t S_{(\mathbf{x})}[U]$ Time-domain denoising applied after numerical time differentiation.
$\partial^{\mathbf{k}}_{\mathbf{x}} u$	$\approx (S_{(\mathbf{x})} D_{x_1})^{k_1} \cdots (S_{(\mathbf{x})} D_{x_d})^{k_d} S_{(\mathbf{x})}[U]$, where $\mathbf{k} = (k_1, \ldots, k_d)$ Spatial denoising applied after every numerical spatial differentiation.

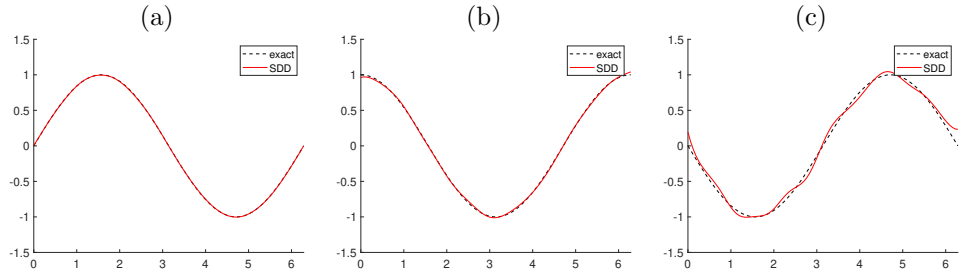

FIG. 2. *Performance of SDD on the data in Figure* 1. (a) *Graph of* $\sin(x)$, $0 \le x \le 2\pi$ *(black) and the denoised data (red) using MLS.* (b) *First order derivatives of the function (black) and the denoised data using SDD (red).* (c) *Second order derivatives of the function (black) and the denoised data using SDD (red). Derivatives are computed by the five-point ENO scheme, and the smoothing operator S is MLS. (Figure in color online.)*

has the same accuracy as the estimated derivative without SDD.

THEOREM 2.1. *Let s be a positive integer. Suppose the given data are noiseless and are sampled from a sufficiently smooth function $u(x)$ with spacing Δx, i.e., $U_i = u(x_i)$ with $x_i = i\Delta x$. Assume polynomials of degree s are used in MLS and the width parameter h is properly chosen. Let D_x be a linear difference scheme satisfying* (a) *at every x_i, $D_x v = dv/dx + O(\Delta x^s)$ for any sufficiently smooth function $v(x)$; and* (b) *$D_x v(x_i) = \sum_{-J \le j \le J} d_j v(x_{i+j})$ for some positive integer J, where d_j depends only on Δx and $d_j = O(1/\Delta x)$ $\forall j$. Then for $k = 0, 1, \ldots, s$, at every x_i*

$$(S_{(x)} D_x)^k S_{(x)}[u] = d^k u/dx^k + O(\Delta x^{s+1-k}) \ .$$

Proof. According to [46, Theorem 4.1], at every x_i

$$S_{(x)}[v] = v + O(\Delta x^{s+1}) \tag{2.7}$$

for any sufficiently smooth function $v(x)$. Therefore, the case $k = 0$ has been proved. Note that $S_{(x)}[v](x_i) = \sum_{-I \le j \le I} a_j v(x_{i+j})$ for some positive integer I and coefficients a_j. Also from [46, Proof of Theorem 4.1], we can deduce that for any function $w(x)$, at every x_i

$$|S_{(x)}[w](x_i)| \le C \max_{-I \le j \le I} |w(x_{i+j})| \tag{2.8}$$

for some constant C. Now assume that at every x_i,

$$(S_{(x)} D_x)^k S_{(x)}[u] = d^k u/dx^k + O(\Delta x^{s+1-k})$$

for some $k \in \{0, 1, \ldots, s-1\}$. We want to show that at every x_i,

$$(S_{(x)}D_x)^{k+1}S_{(x)}[u] = d^{k+1}u/dx^{k+1} + O(\Delta x^{s-k}) \ .$$

Decompose the error at every x_i as

$$\begin{aligned}
&|(S_{(x)}D_x)^{k+1}S_{(x)}[u] - \tfrac{d^{k+1}u}{dx^{k+1}}| \\
&\leq |S_{(x)}D_x[(S_{(x)}D_x)^k S_{(x)}[u] - \tfrac{d^k u}{dx^k}]| + |S_{(x)}D_x \tfrac{d^k u}{dx^k} - \tfrac{d^{k+1}u}{dx^{k+1}}| \\
&= (A) + (B).
\end{aligned}$$

Using the induction assumption and assumption (b) of D_x, we have

$$\left| D_x \left[(S_{(x)}D_x)^k S_{(x)}[u] - \frac{d^k u}{dx^k} \right] \right| = O(\Delta x^{s-k}) \ .$$

Using (2.8), we have $(A) = O(\Delta x^{s-k})$. Using property (a) of D_x, we have

$$D_x \frac{d^k u}{dx^k} = d^{k+1}u/dx^{k+1} + O(\Delta x^s) \ .$$

Combining property (2.8) and (2.7), we have

$$S_{(x)}D_x \frac{d^k u}{dx^k} = d^{k+1}u/dx^{k+1} + O(\Delta x^s) \ ;$$

thus $(B) = O(\Delta x^s)$. Finally, we conclude that

$$(S_{(x)}D_x)^{k+1}S_{(x)}[u] = d^{k+1}u/dx^{k+1} + O(\Delta x^{s-k})$$

and the theorem is proved. ∎

Remark. Suppose $v(x)$ is interpolated by a polynomial $p(x)$ of degree s using Lagrangian interpolation on $s+1$ grid points near x_i; then dp/dx is a linear difference scheme and satisfies assumptions (a) and (b) of Theorem 2.1. In particular, the difference scheme used in this paper is constructed this way.

Theorem 2.1 implies that under proper settings, the estimated derivative by SDD has the same accuracy as the estimated derivative without SDD. Following the proof of Theorem 2.1, one can easily derive similar results for higher order derivatives in multidimensions.

In section 4.9, we explore details of SDD when different smoothing operators are used. We find that MLS has the best performance in terms of preserving the derivative profiles. Therefore, we set S to be MLS in our numerical experiments.

To simplify the notation, in the rest of this paper we use U to denote the denoised data $S_{(\mathbf{x})}[U]$, and $D_t U$ as well as $D^k_{\mathbf{x}} U$ to denote the numerical derivatives with SDD applied as above.

3. Proposed methods: ST and SC. Under the parametric model in (2.2), the PDE identification problem can be reduced to solving the linear system (2.4) for a sparse vector $\mathbf{c}$ with few nonzero entries. Sparse regression can be formulated as the following L_0-minimization:

$$\min \|\mathbf{c}\|_0 \quad \text{subject to } \|F\mathbf{c} - D_t U\| \leq \epsilon \tag{3.1}$$

for some $\epsilon > 0$. However, the L_0-minimization in (3.1) is NP-hard. Its approximate solutions have been intensively studied in the literature. The most popular surrogate

for the L_0 seminorm is the L_1 norm applied in image and signal processing [8, 11]. The L_1-regularized minimization is called Least Absolute Shrinkage and Selection Operator (LASSO) [41], which was used in [17, 34, 35] for PDE identification. The common strategy in these works is to utilize LASSO to select a candidate set, then refine the results with other techniques.

In this paper, we utilize a greedy algorithm called Subspace Pursuit (SP) [10] to select a candidate set. Unlike LASSO, SP takes the sparsity as an input, allowing direct control of the sparsity of the reconstructed coefficient. Let k be a positive integer and denote $\mathbf{b} = D_t U$. For a fixed sparsity level k, $\mathrm{SP}(k; F, \mathbf{b})$ in Algorithm 1 gives rise to a k-sparse vector whose support is selected in a greedy fashion. It was proved that SP gives rise to a solution of the L_0-minimization (3.1) under certain conditions of the matrix F, such as the restricted isometry property [10].

Algorithm 1: Subspace Pursuit $\mathrm{SP}(k; F, \mathbf{b})$.

Input: $F \in \mathbb{R}^{NM^d \times K}$, $\mathbf{b} \in \mathbb{R}^{NM^d}$ and sparsity $k \in \mathbb{N}$.
Initialization: $j = 0$;
$G \leftarrow$ column-normalized version of F;
$\mathcal{I}^0 = \{k$ indices corresponding to the largest magnitude entries in the vector $G^*\mathbf{b}\}$;
$\mathbf{b}^0_{\mathrm{res}} = \mathbf{b} - G_{\mathcal{I}^0} G^\dagger_{\mathcal{I}^0} \mathbf{b}$.
while *True* **do**
  Step 1. $\widetilde{\mathcal{I}}^{j+1} = \mathcal{I}^j \cup \{k$ indices corresponding to the largest magnitude entries in the vector $G^*\mathbf{b}^j_{\mathrm{res}}\}$;
  Step 2. Set $\mathbf{c}_p = G^\dagger_{\widetilde{\mathcal{I}}^{j+1}} \mathbf{b}$;
  Step 3. $\mathcal{I}^{j+1} = \{k$ indices corresponding to the largest elements of $\mathbf{c}_p\}$;
  Step 4. Compute $\mathbf{b}^{j+1}_{\mathrm{res}} = \mathbf{b} - G_{\mathcal{I}^{j+1}} G^\dagger_{\mathcal{I}^{j+1}} \mathbf{b}$;
  Step 5. If $|\mathbf{b}^{j+1}_{\mathrm{res}}\|_2 > \|\mathbf{b}^j_{\mathrm{res}}\|_2$, let $\mathcal{I}^{j+1} = \mathcal{I}^j$ and terminate the algorithm; otherwise set $j \leftarrow j + 1$ and iterate.
Output: $\widehat{\mathbf{c}} \in \mathbb{R}^K$ satisfying $\widehat{\mathbf{c}}_{\mathcal{I}_j} = F^\dagger_{\mathcal{I}_j} \mathbf{b}$ and $\widehat{\mathbf{c}}_{(\mathcal{I}_j)^\complement} = \mathbf{0}$.

We propose two new methods based on SP for PDE identification: Subspace pursuit Time evolution (ST) and Subspace pursuit Cross-validation (SC).

3.1. Subspace pursuit Time evolution (ST). We first propose a method combining SP and the idea of time evolution. In [17], Time Evolution Error (TEE) quantifies the mismatch between the solution simulated from a candidate PDE and the denoised data. Any candidate coefficient vector $\widehat{\mathbf{c}} = (\widehat{c}_1, \widehat{c}_2 \dots)$ defines a candidate PDE:

$$u_t = \widehat{c}_1 + \widehat{c}_2 \partial_{x_1} u + \cdots + \widehat{c}_m u \partial_{x_1} u + \cdots .$$

This PDE is numerically evolved from the initial condition U^0 with a smaller time step $\widetilde{\Delta t} \ll \Delta t$. Denote $\widehat{U}^1, \widehat{U}^2, \dots, \widehat{U}^N$ as this numerical solution at the same time-space location as $U^1, U^2, \dots, U^N$. The TEE of the candidate PDE given by $\widehat{\mathbf{c}}$ is

$$\mathrm{TEE}(\widehat{\mathbf{c}}) = \frac{1}{N} \sum_{n=1}^{N} \|\widehat{U}^n - U^n\|_2 \,,$$

where U^n is the denoised data at time t^n. Figures 3 (a) and (b) illustrate the idea of TEE. When there are several candidate PDEs, the one with the least TEE is picked

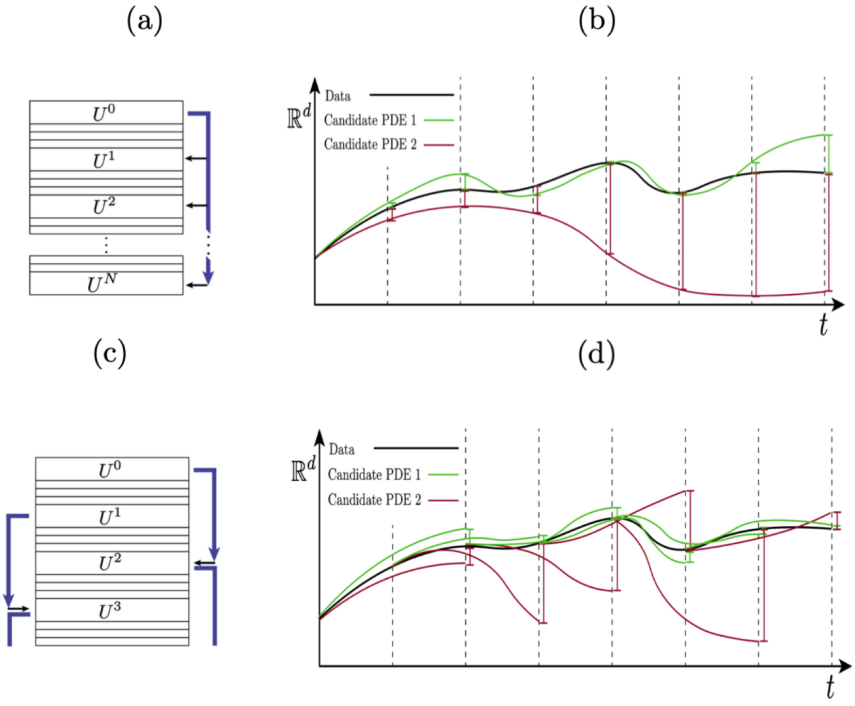

FIG. 3. (a) *and* (b) *illustrate the idea of TEE.* (c) *and* (d) *illustrate MTEE when* $w = 2$. *The blue arrows in* (a) *and* (c) *represent time evolution using the forward Euler scheme on a fine time grid with spacing* $\widetilde{\Delta t} \ll \Delta t$. *In* (b), *two different PDEs (green and red) are evolved, and the green one has a smaller TEE. In* (d), *the candidate PDEs are evolved from multiple time locations, and their numerical solutions are compared with the denoised data after a time length of* $w\Delta t$. *(Figure in color online.)*

[17]. This TEE idea is based on the convergence principle that a correct numerical approximation converges to the true solution as the time step $\widetilde{\Delta t}$ goes to zero. The error from the wrongly identified terms grows during this time evolution process; see more details in [17, section 2.3].

In this paper, we propose a *Multishooting Time Evolution Error* (MTEE). The idea is to evolve a candidate PDE from multiple time locations with a time step $\widetilde{\Delta t} \ll \Delta t$ using the forward Euler scheme for a time length of $w\Delta t$, where w is a positive integer. This scheme is stable as long as the PDE is well posed and the solution is smooth, and when the time step is sufficiently small. Specifically, if r is the highest order of the spatial derivatives, following the CFL condition, we set the time step as $c(\Delta x)^r$ with some constant $c < 1$. Let $\widehat{U}^{(n+w)|n}$ be the numerical solution of the candidate PDE at the time $(n+w)\Delta t$, which is evolved from the initial condition U^n at time $t^n = n\Delta t$. The MTEE is defined as

$$\text{MTEE}(\widehat{\mathbf{c}}; w) = \frac{1}{N-w} \sum_{n=0}^{N-1-w} \|\widehat{U}^{(n+w)|n} - U^{n+w}\|_2 \,. \tag{3.2}$$

Figures 3 (c) and (d) demonstrate the process of multishooting time evolution. While the TEE evolution starts from the initial condition U^0 and ends at T, the MTEE evolution starts from various time locations, such as $t^n, n = 0, \ldots, N-1-w$, and

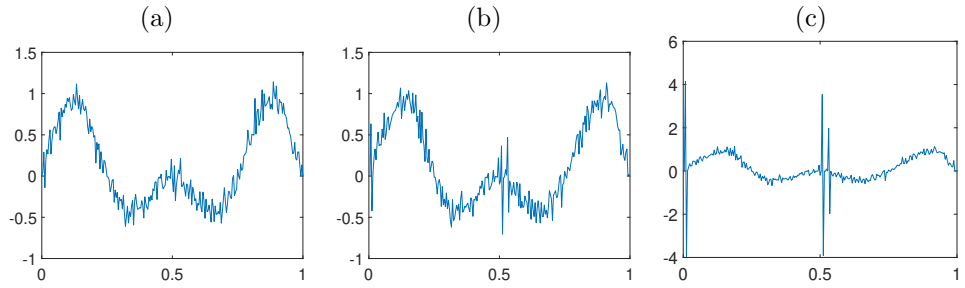

FIG. 4. *Robustness of MTEE over TEE.* (a) *A noisy initial condition for the evolution of the Burgers' equation* $u_t = -uu_x$. *By evolving this noisy initial condition according to* $u_t = -uu_x$, (b) *shows the numerical solution at* $t = 0.02$ *and* (c) *shows the numerial solution at* $t = 0.03$. *The numerical solution blows up at* $t = 0.032$.

lasts for a shorter time, e.g., $w\Delta t$ in our case.

MTEE has two advantages over TEE: (1) MTEE is more robust against noise in comparison with TEE. If $w \ll N$, the noise in the initial condition accumulates for a smaller amount of time in MTEE, which helps to stabilize numerical solvers.

For example, consider identifying the Burgers' equation $u_t = -uu_x$ from a set of noisy data generated with $T = 0.05, \Delta t = 0.001, \Delta x = 1/256$ (see Figure 4). If one evolves the noisy initial condition in Figure 4 (a), using the correct PDE, i.e., $u_t = -uu_x$, the numerical solution blows up at $t = 0.032$. The numerical solutions at $t = 0.02$ and $t = 0.03$ are shown in Figures 4 (b) and (c), respectively. The TEE at $T = 0.05$ is ∞ even for the correct PDE since the numerical solution blows up at $t = 0.032$. On the other hand, MTEE works since we evolve the initial condition for a shorter amount of time, before the numerical solution blows up, such as $t = 0.02$ (corresponding to $w = 20$ in MTEE) in this example.

(2) MTEE is more flexible, and its computation is parallelizable. The flexibility of MTEE comes from two aspects: (i) The error accumulation time can be controlled by the parameter w such that the PDE is evolved for a time length of $w\Delta t$. (ii) One may assign different weights in the calculation of the evolution errors in different periods. Since each time evolution in the multishooting is independent, the computation of MTEE can be parallelized.

The SP algorithm finds a coefficient vector with a specified sparsity, while the correct sparsity is not known from the given data. Based on SP and MTEE, we propose ST, which iteratively refines the selection of features. Figure 5 illustrates the ST iteration: Starting from a large number K (no more than the number of features), each SP(k) coefficient vector is computed for all $k = 0, \ldots, K$. Among these, the k which gives the minimum MTEE is chosen to be K_1. This procedure continues until two consecutive iterations give the same output or only one feature is left. This process will terminate after at most $K - 1$ iterations.

More specifically, as an initial condition, we set $K_0 = K$ and $\mathcal{A}_0 = \{1, \ldots, K\}$. Clearly, this K is bounded by the number of dictionary. At the first iteration, all possible sparsity levels are considered up to K in the SP algorithm. For each $k = 1, \ldots, K$, we run $\mathrm{SP}(k; F, D_t U)$ to obtain a coefficient vector $\widehat{\mathbf{c}}^{(k)} \in \mathbb{R}^K$ such that $\|\widehat{\mathbf{c}}^{(k)}\|_0 = k$, which gives rise to the PDE:

$$u_t = f_{\mathrm{SP}(k)}, \text{ where } f_{\mathrm{SP}(k)} := \widehat{c}_1^{(k)} + \widehat{c}_2^{(k)} \partial_{x_1} u + \cdots + \widehat{c}_m^{(k)} u \partial_{x_1} u + \cdots . \tag{3.3}$$

We then numerically evolve each PDE $u_t = f_{\mathrm{SP}(k)}$ for $k = 1, \ldots, K$ and calculate

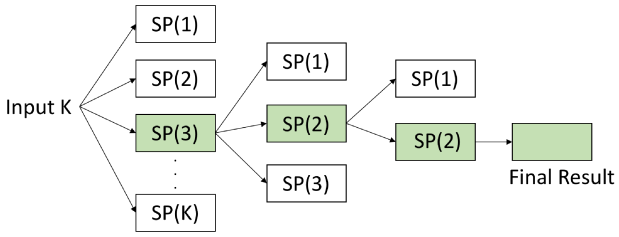

FIG. 5. *An example of the ST iteration. Starting with the large number K, the first iteration gives rise to K candidate coefficients for $k = 1, \ldots, K$. The PDE with the smallest MTEE is picked, e.g., SP(3) with cardinality $K_1 = 3$ and support $\mathcal{A}_1$. The second iteration gives rise to the candidate coefficients only supported on $\mathcal{A}_1$ using SP(k) with $k = 1, 2, 3$. The PDE with the smallest MTEE is found, e.g., SP(2) with cardinality $K_2 = 2$ and support $\mathcal{A}_2$. The third iteration does not change the support, i.e., $\mathcal{A}_3 = \mathcal{A}_2$, so the final output is the coefficient vector of* SP(2).

the corresponding MTEE. Among these PDEs, the one with the smallest MTEE is selected; then let

$$K_1 = \underset{k=1,2,\ldots,K}{\arg\min}\ \mathrm{MTEE}(\widehat{\mathbf{c}}^{(k)}; w) \text{ and } \mathcal{A}_1 = \mathrm{supp}(\widehat{\mathbf{c}}^{(K_1)}) .$$

If $\mathcal{A}_1 = \mathcal{A}_0$, the algorithm is terminated; otherwise, we continue to the second iteration. The proposed method requires solving the sparsity-constrained least-squares problems at least K times. These computations and the evaluation of MTEE can be computed in parallel.

At the second iteration, we refine the selection from the index set $\mathcal{A}_1$ with cardinality K_1. For $k = 0, \ldots, K_1$, we run $\mathrm{SP}(k; [F]_{\mathcal{A}_1}, D_t U)$ to obtain a coefficient vector $\widehat{\mathbf{c}}^{(k)} \in \mathbb{R}^K$ such that

$$\widehat{\mathbf{c}}^{(k)}_{\mathcal{A}_1} = \mathrm{SP}(k; [F]_{\mathcal{A}_1}, D_t U) \text{ and } \widehat{\mathbf{c}}^{(k)}_{\mathcal{A}_1^{\complement}} = \mathbf{0} ,$$

and the associated PDE $u_t = f_{\mathrm{SP}(k)}$ as in (3.3). Among these PDEs, the one with the smallest MTEE is selected, and we denote

$$K_2 = \underset{k=1,2,\ldots,K_1}{\arg\min}\ \mathrm{MTEE}(\widehat{\mathbf{c}}^{(k)}; w) \text{ and } \mathcal{A}_2 = \mathrm{supp}(\widehat{\mathbf{c}}^{(K_1)}) .$$

If $\mathcal{A}_2 = \mathcal{A}_1$, the algorithm is terminated; otherwise, we continue to the next iteration similarly.

The ST iteration will be terminated when the index set remains the same, i.e., $\mathcal{A}_j = \mathcal{A}_{j+1}$. The ST outputs a recovered coefficient vector and the corresponding PDE denoted by $\mathrm{ST}(w)$. A complete description of ST is given in Algorithm 2.

3.2. Subspace pursuit Cross-validation (SC). Our second method utilizes the idea of cross-validation for the linear system in (2.4). Cross-validation is commonly used in statistics for the choice of parameters in order to avoid overfitting [15]. We consider the two-fold cross-validation where data are partitioned into two subsets. One subset is used to estimate the coefficient vector, and the other one is used to validate the candidates. If a suitable coefficient vector is found within one subset, it should yield a small validation error for the other subset because of consistency.

For some fixed ratio parameter $\alpha \in (0, 1)$, we split the rows of $D_t U \in \mathbb{R}^{NM^d}$ (and $F \in \mathbb{R}^{NM^d \times K}$) into two groups indexed by $\mathcal{T}_1$ and $\mathcal{T}_2$, such that $\mathcal{T}_1$ consists of the indices of the first $\lfloor \alpha N M^d \rfloor$ rows and $\mathcal{T}_2$ consists of the indices of the rest of the rows. Since we focus on PDEs with constant coefficients, the idea of cross-validation

Algorithm 2: Subspace pursuit Time evolution (ST).

Input: $F \in \mathbb{R}^{NM^d \times K}$, $D_t U \in \mathbb{R}^{NM^d}$ and a positive integer w.
Initialization: $j = 0$, $K_0 = K$ and $\mathcal{A}_0 = \{1, 2, \ldots, K\}$.
while $\mathcal{A}_{j+1} \neq \mathcal{A}_j$ **do**
Step 1. For $k = 1, 2, \ldots, K_j$, run $\mathrm{SP}(k; [F]_{\mathcal{A}_j}, D_t U)$ to obtain a coefficient vector $\widehat{\mathbf{c}}^{(k)} \in \mathbb{R}^K$ such that

$$\widehat{\mathbf{c}}^{(k)}_{\mathcal{A}_j} = \mathrm{SP}(k; [F]_{\mathcal{A}_j}, D_t U) \ \text{ and } \ \widehat{\mathbf{c}}^{(k)}_{\mathcal{A}_j^{\complement}} = \mathbf{0} \, ,$$

and the associated PDE $u_t = f_{\mathrm{SP}(k)}$ given in (3.3).
Step 2. Among all the PDEs $u_t = f_{\mathrm{SP}(k)}$ for $k = 1, \ldots, K_j$, select the one with the minimum $\mathrm{MTEE}(\widehat{\mathbf{c}}^{(k)}; w)$ and update

$$K_{j+1} = \underset{k=1,2,\ldots,K_j}{\arg\min} \ \mathrm{MTEE}(\widehat{\mathbf{c}}^{(k)}; w) \ \text{ and } \ \mathcal{A}_{j+1} = \mathrm{supp}(\widehat{\mathbf{c}}^{(k_{j+1})}) \, .$$

If $\mathcal{A}_{j+1} = \mathcal{A}_j$, terminate the algorithm; otherwise, update $j = j + 1$.
Output: Recovered coefficient $\widehat{\mathbf{c}}^{K_{j+1}}$ and the corresponding PDE, denoted by $\mathrm{ST}(w)$.

is applicable: if a correct support is identified, the coefficient vector obtained from the data in $\mathcal{T}_1$ should be compatible with the data in $\mathcal{T}_2$.

We introduce our SC algorithm where cross-validation is incorporated into the SP algorithm. SC consists of the following three steps:

Step 1. For each sparsity level $k = 1, 2, \ldots, K$, use SP to select a set of active features:

$$\mathcal{A}_k = \mathrm{supp}(\mathrm{SP}(k; F, D_t U)) \, .$$

Step 2. Use the data in $\mathcal{T}_1$ to compute the estimator for the coefficient vector, $\widehat{\mathbf{c}}^{(k)} \in \mathbb{R}^K$, by the following least-squares problem:

$$\widehat{\mathbf{c}}^{(k)} = \underset{\mathbf{c} \in \mathbb{R}^K \text{such that } \mathbf{c}_{\mathcal{A}_k^{\complement}} = 0}{\arg\min} \| [F]^{\mathcal{T}_1}_{\mathcal{A}_k} \mathbf{c}_{\mathcal{A}_k} - [D_t U]^{\mathcal{T}_1} \|_2^2 \, ,$$

and then use the data in $\mathcal{T}_2$ to compute a Cross-validation Estimation Error (CEE)

$$\mathrm{CEE}(\mathcal{A}_k; \alpha, \mathcal{T}_1, \mathcal{T}_2) = \| [D_t U]^{\mathcal{T}_2} - [F]^{\mathcal{T}_2} \widehat{\mathbf{c}}^{(k)} \|_2 \, . \tag{3.4}$$

Step 3. Set $k_{\min} = \arg\min_k \mathrm{CEE}(\mathcal{A}_k; \alpha, \mathcal{T}_1, \mathcal{T}_2)$ and the estimated coefficient vector is given as

$$\widehat{\mathbf{c}} = \underset{\mathbf{c} \in \mathbb{R}^K \text{such that } \mathbf{c}_{\mathcal{A}_k^{\complement}} = 0}{\arg\min} \| [F]^{\mathcal{T}_1}_{\mathcal{A}_{k_{\min}}} \mathbf{c}_{\mathcal{A}_{k_{\min}}} - [D_t U]^{\mathcal{T}_1} \|_2^2 \, .$$

The identified PDE by SC is denoted as $\mathrm{SC}(\alpha)$.

CEE in (3.4) is an effective measure for consistency. If the estimated coefficient vector's support matches that of the true one, CEE is guaranteed to be small provided there is sufficiently high resolution in time and space.

THEOREM 3.1. *Assume that $D_t U \to u_t$ and $F \to F_0$ pointwise as $\Delta t, \Delta x \to 0$. Let $\mathcal{A}_0 = \mathrm{supp}(\mathbf{c}_0)$, where $\mathbf{c}_0$ is the coefficient vector of the true PDE. For any set of*

Algorithm 3: Subspace pursuit Cross-validation (SC) algorithm.

Input: $F \in \mathbb{R}^{NM^d \times K}$ and $D_t U \in \mathbb{R}^{NM^d}$; $0 < \alpha < 1$ ratio of the training data.

Step 1. For $k = 1, 2, \ldots, K$, run $\mathrm{SP}(k; F, D_t U)$ to obtain the support of the candidate coefficients

$$\mathcal{A}_k = \mathrm{supp}(\mathrm{SP}(k; F, D_t U)) \ .$$

Step 2. For each k, compute the averaged cross-validation error

$$\mathrm{CEE}(\mathcal{A}_k, \alpha) = \frac{1}{2}\left(\mathrm{CEE}(\mathcal{A}_k; \alpha, \mathcal{T}_1, \mathcal{T}_2) + \mathrm{CEE}(\mathcal{A}_k; 1-\alpha, \mathcal{T}_2, \mathcal{T}_1)\right) \ .$$

Step 3. Choose the k which gives the smallest cross-validation error and denote it by $k_{\min}$,

$$k_{\min} = \arg\min_k \mathrm{CEE}(\mathcal{A}_k, \alpha) \ .$$

Estimate the coefficients by least squares as

$$\widehat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbb{R}^K \text{such that } \mathbf{c}_{\mathcal{A}_k^{\complement}}=0}{\arg\min} \|[F]^{\mathcal{T}_1}_{\mathcal{A}_{k_{\min}}} \mathbf{c}_{\mathcal{A}_{k_{\min}}} - [D_t U]^{\mathcal{T}_1}\|_2^2 \ .$$

Output: Recovered coefficient $\widehat{\mathbf{c}}$ and the identified PDE denoted by $\mathrm{SC}(\alpha)$.

support $\mathcal{A}$, we have

$$\mathrm{CEE}(\mathcal{A}; \alpha, \mathcal{T}_1, \mathcal{T}_2) \le \left\| \left([F_0]^{\mathcal{T}_2}_{\mathcal{A}_0} \left([F_0]^{\mathcal{T}_1}_{\mathcal{A}_0}\right)^\dagger - [F_0]^{\mathcal{T}_2}_{\mathcal{A}} \left([F_0]^{\mathcal{T}_1}_{\mathcal{A}}\right)^\dagger \right) [u_t]^{\mathcal{T}_1} \right\|_2 + g(\mathcal{A}; \alpha, \mathcal{T}_1, \mathcal{T}_2) \ ,$$

where $g > 0$ is a sum of residual terms of approximating the partial derivatives and feature matrix using data (see (B.1)*), which is independent of $\mathcal{A}_0$, such that $g \to 0$ as $\Delta t, \Delta x \to 0$.*

Proof. See Appendix B for the proof. ∎

In (3.4), the data in $\mathcal{T}_1$ serve as the training set, and the data in $\mathcal{T}_2$ act as the validation set. One can also use the data in $\mathcal{T}_2$ for training and the data in $\mathcal{T}_1$ for validation, which gives rise to the cross-validation estimation error $\mathrm{CEE}(\mathcal{A}_k; 1-\alpha, \mathcal{T}_2, \mathcal{T}_1)$. To improve the robustness of SC, we replace (3.4) with the following averaged cross-validation error:

$$\mathrm{CEE}(\mathcal{A}_k, \alpha) = \frac{1}{2}\left(\mathrm{CEE}(\mathcal{A}_k; \alpha, \mathcal{T}_1, \mathcal{T}_2) + \mathrm{CEE}(\mathcal{A}_k; 1-\alpha, \mathcal{T}_2, \mathcal{T}_1)\right) \ .$$

In general, one can randomly pick part of the data as the training set and use the rest as the validation set. Our numerical experiments in subsection 4.8 demonstrate that the splitting strategy does not affect the results. For simplicity, we split the data according to the row index in this paper.

The proposed SC algorithm is summarized in Algorithm 3. In comparison with ST, SC does not involve any numerical evolution of the candidate PDE, so the computation of SC is faster.

4. Numerical experiments. In this section, we perform a systematic numerical study to demonstrate the effectiveness of ST and SC and compare them to IDENT

[17]. To measure the identification error, we use the following relative coefficient error e_c and grid-dependent residual error e_r:

(4.1)
$$e_c = \frac{\|\widehat{\mathbf{c}} - \mathbf{c}\|_1}{\|\mathbf{c}\|_1}, \quad e_r = \begin{cases} \sqrt{\Delta x \Delta t}\|F(\widehat{\mathbf{c}} - \mathbf{c})\|_2 \text{ for one-dimensional (1D) PDE,} \\ \sqrt{\Delta x \Delta y \Delta t}\|F(\widehat{\mathbf{c}} - \mathbf{c})\|_2 \text{ for two-dimensional (2D) PDE.} \end{cases}$$

The relative coefficient error e_c measures the accuracy in the recovery of PDE coefficients, while the residual error e_r measures the difference between the learned dynamics and the denoised one by SDD. Since each feature vector in F may have different scales, e_r can be different from e_c in some cases. When the given data contain noise, the features containing higher order derivatives have greater magnitude than the features containing lower order derivatives. In this case, a small coefficient error in the high order terms may lead to a large e_r. We use both e_c and e_r to quantify the PDE identification error. To measure how well the solution of the identified PDE matches the dynamics of the correct PDE, we also use the following evolution error:

$$e_e = \Delta x \Delta t \left(\sum_n \sum_{\mathbf{i}} |u(\mathbf{x_i}, t^n) - \hat{u}(\mathbf{x_i}, t^n)| \right), \tag{4.2}$$

where u and $\hat{u}$ denote the solution of the exact and identified PDE from the same initial condition, respectively.

To generate data, we first solve the underlying PDE by forward Euler scheme using time and space step δt and δx (and δy), respectively, then downsample the data with time and space step Δt and Δx (and Δy). In the noisy case, we add Gaussian noise with standard deviation σ to the clean data. We say that the noise is $p\%$ by setting $\sigma = \frac{p}{100}\sqrt{\frac{1}{NM^d}\sum_n \sum_{\mathbf{i}} (u(\mathbf{x_i}, t^n))^2}$. In the computation of $D_t U$ and the feature matrix F, we always use SDD with MLS with $h = 0.04$ as the smoother. When MLS is used to denoise the data of 2D PDEs, one can either fit 2D polynomials or fit 1D polynomials in each dimension. In this work, we use the second approach. In ST, without specification, $\widetilde{\Delta t} = \Delta t/5$ is used.

We first consider PDEs containing partial derivatives up to the second order. Let the governing equation f be a polynomial with degree up to 2. There are 10 features: $1, u, u^2, u_x, u_x^2, uu_x, u_{xx}, u_{xx}^2, uu_{xx}, u_x u_{xx}$ in the dictionary for 1D PDEs. For 2D PDEs, there are 28 features, which contain $1, u, u_x, u_y, u_{xx}, y_{xy}, u_{yy}$ and their pairwise products. In the following examples, without specification, the spatial domain $[0, 1]$ is used for 1D PDEs and $[0, 1]^2$ is used for 2D PDEs. For both cases, the zero Dirichlet boundary condition is used for all examples.

4.1. Transport equation. Our first experiment is a transport equation with zero Dirichlet boundary condition:

$$u_t = -u_x \ , \tag{4.3}$$

with an initial condition of

$$u(x, 0) = \begin{cases} \sin^2(2\pi x/(1-T))\cos(2\pi x/(1-T)) \text{ for } 0 \le x \le 1 - T, \\ 0 \text{ otherwise} \end{cases}$$

for $0 < t \le T$ and $x \in [0, 1]$. The clean data $\mathbf{D}$ is generated by explicitly solving (4.3) with $\delta x = \Delta x = 1/256, \delta t = \Delta t = 10^{-3}$, and $T = 0.05$. In theory, for the

transport equation, the zero boundary condition should only be applied to the inflow boundary. We design our initial condition and choose the evolution time T that the solution value is 0 at the outflow boundary during the evolution. The same setup is considered in the rest of this section.

TABLE 2

Identification of the transport equation (4.3) *with different noise levels. In the noise-free case, applying SDD does not introduce a strong bias. The identification results (second column) by ST and SC are stable even with* 30% *noise. Here* $w = 20$ *for ST, and* $\alpha = 1/200$ *for SC.*

Method	0% noise without SDD	e_c	e_r
ST	$u_t = -0.9994u_x$	6.20×10^{-4}	4.89×10^{-4}
SC	$u_t = -0.9993u_x - 0.0010u_{xx}$	1.65×10^{-3}	1.11×10^{-2}
	0% noise with SDD	e_c	e_r
ST	$u_t = -0.9997u_x$	3.36×10^{-4}	2.64×10^{-4}
SC	$u_t = -0.9997u_x - 0.0010u_{xx}$	1.34×10^{-3}	1.11×10^{-2}
	10% noise without SDD	e_c	e_r
ST	$u_t = -3.028 \times 10^{-4}u_{xx}$	1.00	5.55
SC	$u_t = 9.4224u - 2.9992u_{xx}$	1.04×10	5.62
	10% noise with SDD	e_c	e_r
ST, SC	$u_t = -1.0357u_x$	3.57×10^{-2}	2.67×10^{-2}
	30% noise without SDD	e_c	e_r
ST	$u_t = 8.0587 \times 10u - 2.6316 \times 10^{-4}u_{xx}$	8.16×10	1.88×10
SC	$u_t = 8.2488 \times 10u$	8.25×10	1.86×10
	30% noise with SDD	e_c	e_r
ST, SC	$u_t = -0.9421u_x$	5.79×10^{-2}	4.31×10^{-2}

Table 2 shows the results of ST(20) and SC(1/200) with various noise levels. In practice, we have no a priori knowledge of whether the given data contain noise, so we conduct two experiments with and without SDD to check the effect of SDD on clean data. We observe that SDD makes a small difference in the noise-free case. With clean data, SC identifies an additional u_{xx} term with a small coefficient, while ST can rule out all wrong terms. The corresponding e_c and e_r are both small. For 10% or 30% noise, the results by ST and SC with and without SDD are also shown. With SDD, both ST and SC identify the correct PDE with small e_c and e_r values. SDD significantly improves the results.

To further demonstrate the significance of SDD and the effectiveness of ST and SC, we display the noisy data with 10% and 30% noise, the denoised data, and the recovered dynamics in Figure 6. Even though the given data contain a large amount of noise, the recovered dynamics are close to the clean data. In the rest of the examples, SDD is always used for ST, SC, and IDENT on noisy data.

Figure 7 shows how e_c, e_r, and e_e change when the noise level varies. Each experiment is repeated 50 times and the error is averaged. We test IDENT, ST(20), and SC(1/200). Figure 7 (a) shows that e_c of ST or SC is much smaller than that of IDENT when the noise level is larger than 20%. Figures 7 (b) and (c) show e_r and e_e versus noise, respectively. The coefficient error e_c by ST and SC is significantly smaller than that of IDENT.

In Figure 8, we explore the robustness of SC with respect to the choice of α. We present e_c and e_r versus $1/\alpha$ in (a) and (b), respectively, with $1\%, 5\%, 10\%, 20\%$ noise. Each experiment is repeated 50 times and the error is averaged. The result shows that SC, in this case, is not sensitive to α, and there is a wide range of choices of α that give rise to a small error.

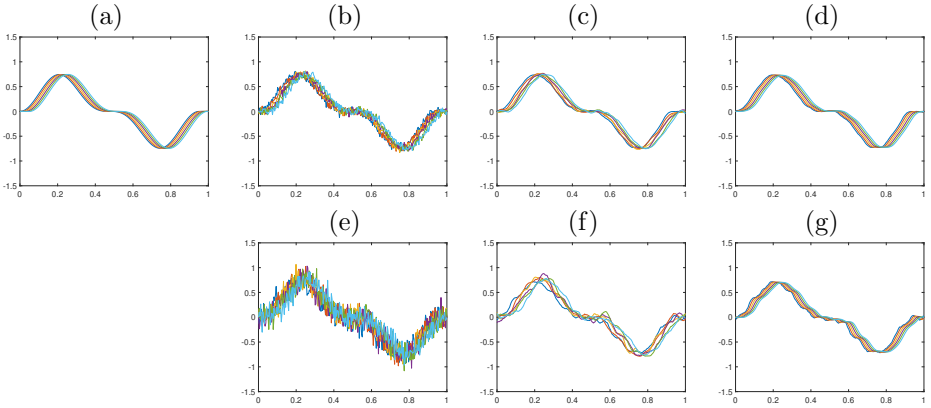

FIG. 6. *Noisy and denoised data of the transport equation* (4.3)*, as well as simulations of the recovered PDE.* (a) *The clean data,* (b) *data with* 10% *noise,* (c) *the denoised data* $S_{\mathbf{x}}[U]$*,* (d) *simulation of the PDE identified by ST and SC (identical).* (e) *Data with* 30% *noise,* (f) *the denoised data* $S_{(\mathbf{x})}[U]$*, and* (g) *simulation of the PDE identified by ST and SC (identical).*

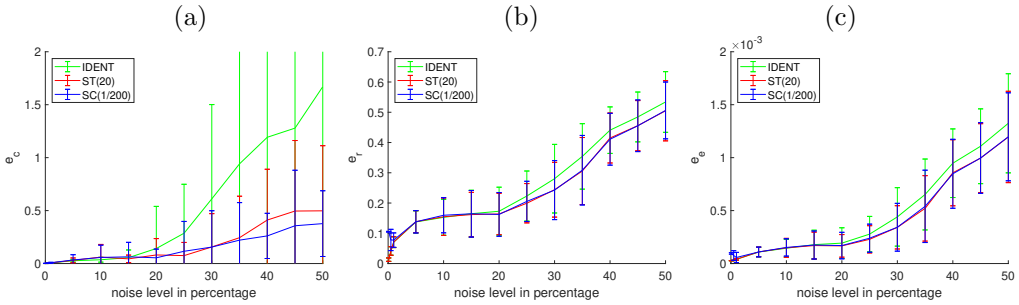

FIG. 7. *The average error* e_c, e_r*, and* e_e *over* 50 *experiments for the transport equation* (4.3) *with respect to various noise levels.* (a) *The curve represents the average* e_c *for IDENT* [17] *(Green), ST (Red), and SC (Blue), and the standard deviation is represented by vertical bars.* (b) *The average and variation of* e_r *for IDENT (Green), ST (Red), and SC (Blue).* (c) *The average and variation of* e_e *for IDENT (Green), ST (Red), and SC (Blue). The coefficient error* e_c *by ST and SC is significantly smaller than that of IDENT. (Figure is in color online.)*

We next test ST and SC on data generated from the transport equation with a discontinuous initial condition. We set the initial condition as

(4.4)
$$u(x,0) = \begin{cases} \sin^2(2\pi x/(1-T))\cos(2\pi x/(1-T)) & \text{for } 0 \le x < (1-T)/3, \\ -\cos^2(2\pi x/(1-T)) + 0.5 & \text{for } (1-T)/3 \le x < 2(1-T)/3, \\ \sin^2(2\pi x/(1-T)) & \text{for } 2(1-T)/3 \le x \le (1-T), \\ 0 & \text{otherwise.} \end{cases}$$

The clean data is generated by explicitly solving (4.3) with $\delta x = \Delta x = 1/256, \delta t = \Delta t = 10^{-3}$, and $T = 0.05$. After adding i.i.d. Gaussian noise, we have the noisy data. We show the clean data and the noisy data in Figure 9. The identification results are shown in Table 3. Even with the existence of discontinuities, ST and SC are stable and can identify the correct PDE with up to 30% noise.

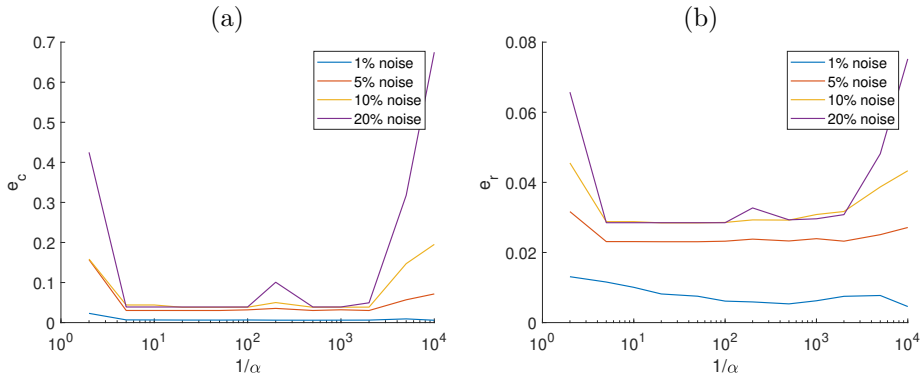

FIG. 8. *Robustness of SC to the choice of α for the recovery of the transport equation* (4.3). (a) *and* (b) *display e_c and e_r versus $1/\alpha$, respectively, with* 1% *(blue),* 5% *(red),* 10% *(orange),* 20% *(purple) noise. Each experiment is repeated* 50 *times, and the errors are averaged. We observe that SC is not sensitive to α, and there is a wide range of values for α that give rise to a small error. (Figure is in color online.)*

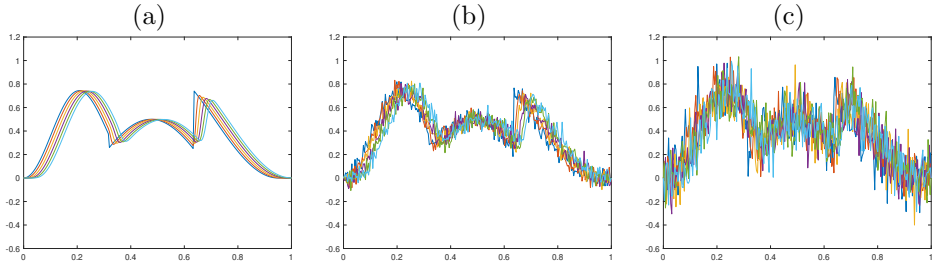

FIG. 9. *Clean and noisy data of the transport equation* (4.3) *with the discontinuous initial condition* (4.4). (a) *Clean data.* (b) *Noisy data with* 10% *noise.* (c) *Noisy data with* 30% *noise.*

TABLE 3

Identification of the transport equation (4.3) *with the discontinuous initial condition* (4.4) *and different noise levels. In the noise-free case, applying SDD does not introduce strong bias. The identification results (second column) by ST and SC are stable even with* 30% *noise. Here $w = 20$ for ST and $\alpha = 1/200$ for SC.*

Method	0% noise without SDD	e_c	e_r
ST	$u_t = -1.0091u_x + 9.65 \times 10^{-4}u_{xx}$	1.01×10^{-2}	1.64×10^{-1}
SC	$u_t = -1.0511u_x$	5.11×10^{-2}	4.43×10^{-2}
	0% noise with SDD	e_c	e_r
ST, SC	$u_t = -1.0274u_x$	2.74×10^{-2}	1.95×10^{-2}
	10% noise	e_c	e_r
ST, SC	$u_t = -0.9913u_x$	8.72×10^{-3}	5.90×10^{-3}
	30% noise	e_c	e_r
ST, SC	$u_t = -0.9239u_x$	7.61×10^{-2}	5.36×10^{-2}

4.2. Burgers' equation. In the second example, we test our methods on the Burgers' equation, which is a first order nonlinear PDE:

$$u_t = -uu_x \tag{4.5}$$

for $0 < t \leq T$. We use the initial condition

$$u(x,0) = \sin(4\pi x)\cos(\pi x) \tag{4.6}$$

and zero Dirichlet boundary condition. Our data is generated by solving (4.5) with $\delta x = \Delta x = 1/256, \delta t = \Delta t = 10^{-3}$, and $T = 0.05$.

TABLE 4

Identification of the Burgers' equation (4.5) *with initial condition* (4.6) *and different noise levels. The identification results (second column) by ST and SC are good with small* e_c *and* e_r *for a noise level up to* 40%. *Here* $w = 20$ *for ST and* $\alpha = 1/500$ *for SC.*

Method	0% noise without SDD	e_c	e_r
ST	$u_t = -1.0023uu_x - 2.38 \times 10^{-5}u_xu_{xx}$	2.35×10^{-3}	5.07×10^{-3}
SC	$u_t = -0.9960uu_x$	4.01×10^{-3}	2.58×10^{-3}
	0% noise with SDD	e_c	e_r
ST	$u_t = -1.0079uu_x - 0.0001u_xu_{xx}$	7.97×10^{-3}	1.43×10^{-2}
SC	$u_t = -0.9888uu_x$	1.12×10^{-2}	7.20×10^{-3}
	10% noise	e_c	e_r
ST, SC	$u_t = -1.0246uu_x$	2.46×10^{-2}	1.52×10^{-2}
	40% noise	e_c	e_r
ST, SC	$u_t = -0.7366uu_x$	2.63×10^{-1}	1.64×10^{-1}

Table 4 shows the results of ST(20) and SC(1/500) with various noise levels. With clean data, ST identifies an additional term, but its coefficient is very small, and the corresponding e_c and e_r are small. SC works very well on clean data. With 10% and 40% noise, both methods identify the same PDE with small e_c and e_r.

Figure 10 shows how e_c, e_r, and e_e change when the noise level varies. Each experiment is repeated 50 times and the errors are averaged. We test IDENT, ST(20), and SC(1/500). The results in Figure 10 show that ST and SC perform better than IDENT.

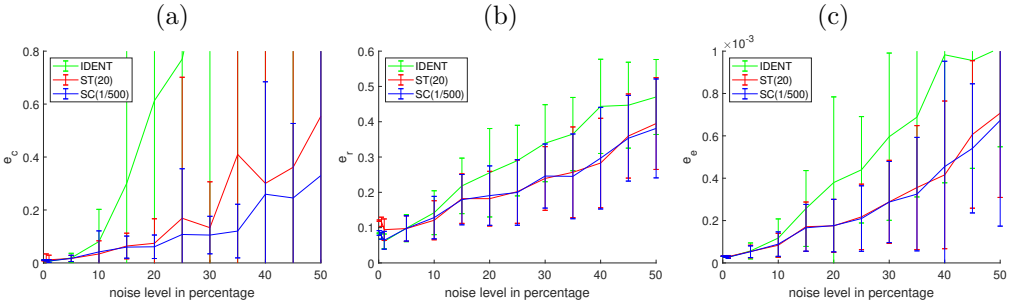

FIG. 10. *The average error* e_c, e_r, *and* e_e *over* 50 *experiments for the Burgers' equation* (4.5) *with respect to various noise levels, where the initial condition is* (4.6). (a) *The curve represents the average* e_c *for IDENT* [17] *(green), ST (red), and SC (blue), and the standard deviations are represented by vertical bars.* (b) *The average and variation of* e_r *for IDENT (green), ST (red), and SC (blue).* (c) *The average and variation of* e_e *for IDENT (green), ST (red), and SC (blue). The* e_c, e_r, *and* e_e *of ST and SC are much smaller than those of IDENT. (Figure in color online.)*

In Table 5, we compare SC, ST from this paper with IDENT in [17], the methods proposed in [34, 35]. The method from [35] uses the spectral method to compute the spatial derivatives, which requires periodic boundary conditions. For a fair compari-

son, we use the initial condition

$$u(x,0) = \sin(4\pi x)\cos(2\pi x) \tag{4.7}$$

and the periodic boundary condition (in which the boundary values are always 0). Our data is generated by solving (4.5) with $\delta x = \Delta x = 1/256, \delta t = \Delta t = 10^{-3}$, and $T = 0.05$. We set $w = 20$ for ST, and $\alpha = 1/500$ for SC. For IDENT, we use SDD to denoise the data and compute the partial derivatives, which improves the original IDENT in [17]. For the method in [35], we use the denoising method specified in [35, Example 3.9]. The identification results are shown in Table 5. Table 5 shows that ST, SC, and IDENT are more robust than the method in [35] at various noise levels. The errors given by ST, SC, and IDENT are also smaller. The results by the method in [34] are similar to those of ST and SC when the noise level is low. For a large level of noise, for example 40%, ST and SC are more robust than the method in [34]. ST and SC can still identify the correct PDE with 40% noise.

TABLE 5

Comparison of ST, SC with IDENT in [17] *and the methods in* [35] *and* [34] *for the identification of the Burgers' equation* (4.5) *with the initial condition* (4.7)*, and various noise levels. In this table, we only include the reconstructed terms with the coefficient magnitudes above* 10^{-2}*. ST and SC are very stable compared to IDENT and the methods in* [35] *and* [34]*. The coefficient error* e_c (4.1) *and the time evolution error* e_e (4.2) *are presented. With large noise, the errors given by ST, SC are smaller than the errors by other methods.*

Method	0% noise	e_c	e_e
[35]	$u_t = -0.01u - 0.95uu_x$	6.49×10^{-2}	1.56×10^{-4}
[34]	$u_t = -0.99uu_x$	1.0×10^{-2}	3.46×10^{-5}
ST, SC, IDENT	$u_t = -0.97uu_x$	2.75×10^{-2}	8.01×10^{-5}
	1% noise	e_c	e_2
[35]	$u_t = -0.14u + 0.01u^2 - 0.89uu_x$	2.82×10^{-1}	3.42×10^{-4}
[34]	$u_t = -0.99uu_x$	1.0×10^{-2}	3.46×10^{-5}
ST, SC, IDENT	$u_t = -0.98uu_x$	1.87×10^{-2}	5.50×10^{-5}
	10% noise	e_c	e_e
[35]	$u_t = -0.07 + 0.4u + 0.44u^2 - 0.15uu_x$	1.76	2.59×10^{-3}
[34]	$u_t = -0.94uu_x$	6.0×10^{-2}	1.77×10^{-4}
IDENT	$u_t = 0.03u_x - 1.00uu_x$	3.0×10^{-2}	2.25×10^{-4}
ST, SC	$u_t = -1.00uu_x$	1.74×10^{-3}	2.88×10^{-5}
	40% noise	e_c	e_e
[35]	$u_t = -1 + 10.67u + 1.84u^2 - 0.02uu_x$	13.59	7.16×10^{-3}
[34]	$u_t = -0.93u - 0.38uu_x$	1.56	1.84×10^{-3}
ST, SC, IDENT	$u_t = -1.02uu_x$	2.39×10^{-2}	8.27×10^{-5}

4.3. Burgers' equation with diffusion. Our third example is the Burgers' equation with diffusion, which is a second order nonlinear PDE:

$$u_t = -uu_x + 0.1u_{xx}\ . \tag{4.8}$$

We use the initial condition $u(x,0) = \sin(3\pi x)\cos(\pi x)$ and zero Dirichlet boundary condition. We first solve (4.8) with $\delta x = 1/256, \delta t = 10^{-5}$, and $T = 0.05$. The

TABLE 6

Identification of the Burgers' equation with diffusion (4.8) *with different noise levels. The identification results (second column) by ST and SC are good with small* e_c *and* e_r *for a noise level up to* 5%. *Here* $w = 20$ *for ST and* $\alpha = 1/10$ *for SC.*

Method	0% noise without SDD	e_c	e_r
ST, SC	$u_t = -1.0018uu_x + 0.1001u_{xx}$	1.67×10^{-3}	8.14×10^{-4}
	0% noise with SDD	e_c	e_r
ST, SC	$u_t = -0.9994uu_x + 0.1009u_{xx}$	1.36×10^{-3}	7.68×10^{-3}
	1% noise	e_c	e_r
ST, SC	$u_t = -0.9901uu_x + 0.1013u_{xx}$	1.02×10^{-2}	1.19×10^{-2}
	5% noise	e_c	e_r
ST, SC	$u_t = -1.0170uu_x + 0.0976u_{xx}$	1.77×10^{-2}	2.21×10^{-2}

given data is downsampled from the numerical solution such that $\Delta x = 1/64$ and $\Delta t = 10^{-4}$.

Table 6 shows the results of ST(20) and SC(1/10) with various noise levels. With clean data, 1% and 5% noise, both methods identify the PDE with small e_c and e_r.

Figure 11 shows how e_c, e_r, and e_e change when the noise level varies from 0.1% to 10%. Each experiment is repeated 50 times, and the error is averaged. We test IDENT, ST(20), and SC(1/10). Among the three methods, ST is the best. SC does not perform as well as ST and IDENT when the noise level is large. For high order PDEs, the high order derivatives are heavily contaminated by noise, even with SDD, which affects the accuracy of cross-validation. While ST and IDENT use time evolution, it is easier to pick correct features. In general, ST performs better than SC for high order PDEs when the given data contain heavy noise.

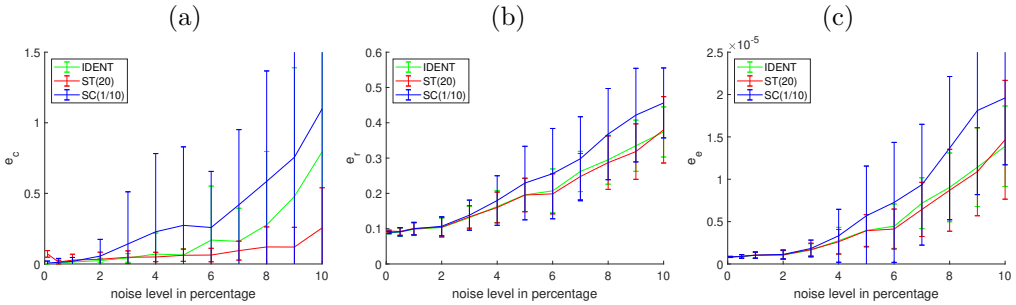

FIG. 11. *The average error* e_c, e_r *and* e_e *over* 50 *experiments of the Burgers' equation with diffusion* (4.8) *with respect to various noise levels.* (a) *The curve represents the average* e_c *for IDENT* [17] *(green), ST (red), and SC (blue), and the standard deviations are represented by vertical bars.* (b) *The average and variation of* e_r *for IDENT (green), ST (red), and SC (blue).* (c) *The average and variation of* e_e *for IDENT (green), ST (red), and SC (blue). Among the three methods, ST gives the best result. (Figure in color online.)*

In Figure 12, we explore the effect of α in SC on the Burgers' equation with diffusion. Figures 12 (a) and (b) show e_c and e_r versus $1/\alpha$, respectively, with 0.5%, 1%, 3%, and 5% noise. When the noise level is low, such as 0.5% and 1%, we have a wide range of good choices of α which give rise to a smaller error. As the noise level increases, the range of the optimal α becomes narrow.

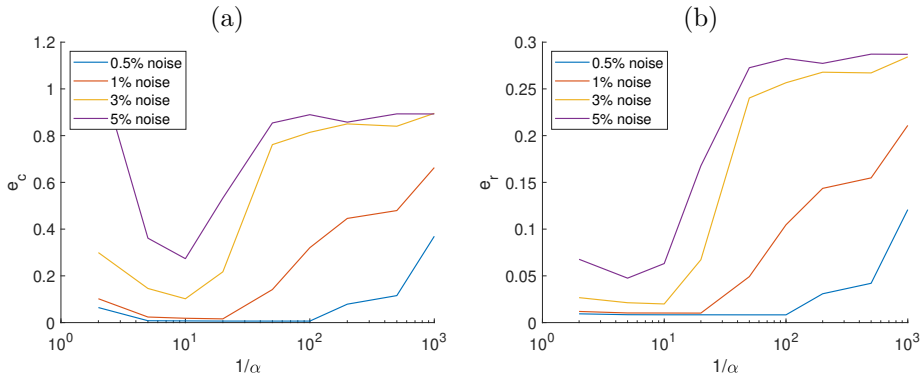

FIG. 12. *Robustness of SC to the choice of α for the recovery of the Burgers' equation with diffusion* (4.8). (a) *and* (b) *display e_c and e_r versus $1/\alpha$, respectively, with* 0.5% *(blue),* 1% *(red),* 3% *(orange),* 5% *(purple) noise. Each experiment is repeated* 50 *times, and the errors are averaged. When the noise level is low, such as* 0.5% *and* 1%*, there is a wide range of values for α, which give a small error. As the noise level increases, the range of the optimal α becomes narrow.*

4.4. The KdV equation. We test our algorithms on the KdV equation

$$u_t + 6uu_x + u_{xxx} = 0 \tag{4.9}$$

on the spatial domain $[-10, 10]$ and the time domain $0 \le t \le T$ with $T = 0.4$. We use the initial condition $u(x, 0) = 5\,\mathrm{sech}^2(1.2x)$ and zero Dirichlet boundary condition. The data is generated with $\delta x = \Delta x = 0.1, \delta t = 10^{-5}$. Data are downsampled in the time domain with $\Delta t = 10^{-3}$. Our dictionary contains $1, u, u_x, u_{xx}$, and u_{xxx} and their pairwise products. There are 15 terms in the dictionary. The identified PDE by ST and SC from clean data is shown in Table 7. In this example, $w = 20, \widetilde{\Delta t} = \Delta t/100$ is used in ST, and $\alpha = 1/1000$ is used in SC. Our results show that both ST and SC can identify the correct PDE.

TABLE 7
Identification of the KdV equation (4.9)*. Both ST and SC can identify the correct PDE.*

Method	Identified PDE	e_c	e_r
ST, SC	$u_t = -6.135uu_x - 1.0580u_{xxx}$	2.77×10^{-2}	1.21

4.5. A larger dictionary. The examples above involve a dictionary which consists of the leading terms in the Taylor expansion of the governing equation $f(u, \partial_{\mathbf{x}} u, \partial^2_{\mathbf{x}} u)$. Our method is general and can be applied to other dictionaries.

We next test ST and SC on a larger dictionary, which includes $1, u, u_x, u_{xx}$ and $\sin(2\pi u), \cos(2\pi u)$ and their pairwise products. Since $\sin^2(2\pi u) + \cos^2(2\pi u) = 1$, we exclude the term $\cos^2(2\pi u)$ to guarantee a set of linearly independent features. This dictionary contains 20 features. We consider the following PDE:

$$u_t = u - 0.1u_x \sin(2\pi u) \tag{4.10}$$

with the initial condition $u(x, 0) = 0.8\sin(3\pi x)\cos(\pi x)$ and zero Dirichlet boundary condition. The data are generated by solving (4.10) with $\delta x = \Delta x = 1/256, \delta t = \Delta t = 4 \times 10^{-3}$, and $T = 0.2$. The identified PDEs by ST and SC with various noise levels are shown in Table 8. On the clean data without SDD, ST identifies an additional

term whose coefficient is very small. The corresponding e_c and e_r are very small. With up to 10% noise, both ST and SC identify the correct PDE with a small e_c and e_r.

TABLE 8
Identification of the (4.10) *with different noise levels. The results (second column) by ST and SC are good with small e_c and e_r for up to 5% noise. Here $w = 20$ for ST and $\alpha = 1/500$ for SC.*

Method	0% noise without SDD	e_c	e_r
ST	$u_t = 0.9994u - 0.0995\sin(2\pi u)u_x - 2.90 \times 10^{-5}\cos(2\pi u)u_{xx}$	1.01×10^{-3}	1.73×10^{-3}
SC	$u_t = 0.9987u - 0.0992\sin(2\pi u)u_x$	1.88×10^{-3}	1.13×10^{-3}
	0% noise with SDD	e_c	e_r
ST, SC	$u_t = 0.9903u - 0.0895\sin(2\pi u)u_x$	1.83×10^{-2}	1.49×10^{-2}
	5% noise	e_c	e_r
ST, SC	$u_t = 0.9909u - 0.0887\sin(2\pi u)u_x$	1.85×10^{-2}	1.56×10^{-2}
	10% noise	e_c	e_r
ST, SC	$u_t = 1.0646u - 0.1026\sin(2\pi u)u_x$	6.11×10^{-2}	1.33×10^{-2}

4.6. Two-dimensional PDEs. We next apply our methods to identify PDEs in a 2D space. The PDEs are solved with $\delta x = \delta y = 0.02$ and $\delta t = 8 \times 10^{-4}$. Data are downsampled from the numerical solution with $\Delta x = 0.04$ and $\Delta t = 8 \times 10^{-3}$. We fix $w = 10$ for ST and $\alpha = 3/200$ for SC.

The identification of 2D PDEs is more challenging and more sensitive to noise. There are more features in two dimensions, and the directional variation of the data adds complexity to the problem. We will show that both ST and SC are robust against noise.

We first consider the following PDE:

$$\text{(4.11)} \quad \begin{cases} u_t = 0.02u_{xx} - uu_y \text{ for } (x, y, t) \in [0, 1]^2 \times [0, 0.1], \\ u(x, y, 0) = \sin^2(\frac{3\pi x}{0.9})\sin^2(\frac{2\pi x}{0.9}) \text{ when } (x, y) \in [0, 0.9]^2 \text{ and 0 otherwise,} \end{cases}$$

which has different dynamics along the x and y directions. Table 9 shows the identification results of ST(10) and SC(3/200) with noise level 0%, 5%, and 10%. Both methods identify the same features with small e_c and e_r.

TABLE 9
Identification of the PDE (4.11) *with different noise levels. The results (second column) by ST and SC have small e_c and e_r for up to 10% noise. Here $w = 10$ for ST, and $\alpha = 3/200$ for SC.*

Method	0% noise	e_c	e_r
ST, SC	$u_t = 0.0189u_{xx} - 0.9525uu_y$	4.75×10^{-2}	2.48×10^{-2}
	5% noise	e_c	e_r
ST, SC	$u_t = 0.0178u_{xx} - 0.9362uu_y$	8.43×10^{-2}	7.45×10^{-2}
	10% noise	e_c	e_r
ST, SC	$u_t = 0.0134u_{xx} - 0.8674uu_y$	1.33×10^{-1}	1.79×10^{-1}

4.7. Identifiability based on the given data. For the PDE identification, especially in high dimensions, the given data U plays an important role. When the initial condition has sufficient variations in each dimension, the correct PDE can be identified. Otherwise, there may be multiple PDEs which generate the same dynamics.

For example, we consider the following transport equation:

$$\begin{cases} u_t = -0.5u_x + 0.5u_y, \ (x,y) \in [0,1] \times [0,1], \ t \in [0,0.1], \\ u(x,y,0) = f(x,y), \ (x,y) \in [0,1] \times [0,1], \end{cases} \tag{4.12}$$

where f denotes the initial condition.

We first choose the initial condition $f(x,y) = \sin(2\pi x/0.9)^2 \sin(2\pi y/0.9)^2$ for $(x,y) \in [0,0.9] \times [0,0.9]$ and 0 otherwise. The noise-free data are generated with $\delta x = \delta y = 0.02$ and $\delta t = 7 \times 10^{-4}$, and downsampled in space by a factor of 2 and in time by a factor of 10. The identified PDE by SC(1/200) is

$$u_t = -0.5001u_x + 0.4800u_y \ ,$$

where the recovered coefficients are very close to the true coefficients. The same result is identified by using ST(20).

We next choose $f(x,y) = \sin(2\pi x/0.9)^2$ for $(x,y) \in [0,0.9] \times \mathbb{R}$ and 0 otherwise. Our methods SC(1/200) and ST(20) both identify

$$u_t = -0.4992u_x \ . \tag{4.13}$$

With this initial condition, the PDE in (4.12) has the exact solution:

$$u(x,y,t) = \begin{cases} \sin(\frac{2\pi(x-0.5t)}{0.9})^2, & x \in [0.5t, 0.9+0.5t], \ (x,y) \in \mathbb{R} \times [0,1], \ t \in [0,0.1], \\ 0 \quad \text{otherwise}, \end{cases}$$

which also satisfies $u_t = -0.5u_x$. The identified PDE in (4.13) approximates this simpler equation. Since the given data only vary along the x direction, the columns in the feature matrix related to y, e.g., u_y, u_xu_y, and u_{yy}, are mostly 0. This explains why our method identifies the PDE in (4.13), instead of (4.12).

In this problem, the original PDE can be identified if the initial condition has sufficient variations. The identifiability of a PDE for a given dictionary under sparsity constraints can be defined as follows: Suppose the original PDE is associated with the coefficient vector $\mathbf{c}_0$ with sparsity S. This PDE is identifiable if there is a unique coefficient vector with sparsity no more than S, such that the evolution of the PDE associated with this coefficient vector, starting from the given initial condition, matches the given data. We believe it is an open question to investigate the theoretical conditions under which the PDE is identifiable. Roughly speaking, the PDE problem is identifiable if the PDE solution with a given initial condition gives rise to the feature matrix F, which has a small pairwise coherence, in the sense that any two columns of F have a small correlation. We refer to [17, Theorem 1] for an identifiability condition in LASSO.

4.8. SC comparison. Our SC strategy is two-fold: In the first fold, we first choose the α fraction of the rows for training and the rest for testing. In the second fold, we choose the last alpha fraction of rows for training and the rest for testing. Then we take the average of the two testing errors. We next compare the identification results using our current strategy, the random selection, K-fold cross-validation, and Monte-Carlo cross-validation:

- SC (our current strategy): Without changing the time order of the data, for a fixed $0 < \alpha < 1$, we select the PDE by minimizing the average testing error of two types. (1) Head: use the first α of the data for training and the rest for testing. (2) Tail: use the last α of the data for training and the rest for testing.

TABLE 10
The PDE identification of (4.14) *by SC with different sampling strategies.*

With 0.5% noise	
Sampling strategy	Identified PDE
SC: $\alpha = 1/400$	$u_t = -0.5000uu_x + 0.5002uu_y$
RSC: $\alpha = 1/400$	$u_t = -0.5000uu_x + 0.5002uu_y$
K-CV: $K = 400$	$u_t = -0.5000uu_x + 0.5002uu_y$
MC-CV: $N = 100, \alpha = 1/400$	$u_t = -0.5000uu_x + 0.5002uu_y$

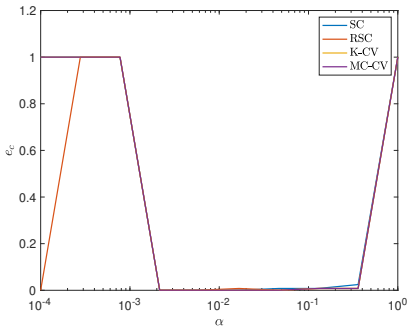

FIG. 13. *The coefficient errors e_c for different sampling strategies for the identification of the PDE in* (4.14) *as the parameter α varies. This shows that, in general, different sampling strategies in SC lead to similar identification results.*

- Random SC (RSC): Randomly permute the data in time, then the remaining procedure is the same as SC.
- K-fold cross-validation (K-CV): Randomly permute the data in time, then uniformly split the data into K groups. The error for a candidate PDE is evaluated by taking the average of K testing errors: for $k = 1, 2, \ldots, K$, while the kth group data is used for training, and the rest is for testing.
- Monte-Carlo cross-validation (MC-CV): Fix the number of simulation N and a coefficient $0 < \alpha < 1$. For $n = 1, \ldots, N$, randomly permute the data and use the first α of the data for training, and the rest for testing.

We consider the following underlying PDE:

$$u_t = -0.5uu_x + 0.5uu_y \tag{4.14}$$

with the initial condition $f(x, y) = \sin(2\pi(x + y))$ filtered by the Tukey window to comply with our zero-boundary requirement. We add 0.5% noise to the data set. The methods above identify the same correct model, as shown in Table 10. Moreover, the effective ranges for α (or, equivalently, $1/K$) for these sampling schemes are similar. This is demonstrated in Figure 13, where we vary α and record the coefficient errors e_c of the identified PDEs, respectively.

4.9. Choice of smoother in SDD. In this paper, we use Moving Least Squares (MLS) as the denoising in SDD. To numerically justify this choice among Moving Average (MA) [40], cubic spline interpolation [9], and diffusion smoothing [47], we present the SDD results with these smoothers in Figure 14. We first solve the PDE

$$u_t = -0.4uu_x - 0.2uu_y, \ (x, y) \in [0, 1] \times [0, 1], \ t \in [0, 0.15] \ , \tag{4.15}$$

with $\Delta t = 0.005$ and $\Delta x = \Delta y = 0.01$, where the initial condition is $u(x, y, 0) = \sin(3\pi x)\sin(5\pi y)$. Then 5% Gaussian noise is added to the numerical solution. Given the noisy data, we perform SDD denoising with different smoothers to obtain various partial derivatives. In MLS, we take the bandwidth $h = 0.04$. For MA, the window size for averaging is fixed to be 3. For cubic spline (CS), we use the MATLAB function *csaps* with $p = 0.5$. For the diffusion (DF) denoising, we evolve the noisy surface following the heat equation $u_t = u_{xx} + u_{yy}$ with a time step size $(\Delta x)^2/4$ for 5 iterations. Figure 14 shows the SDD results of u, u_x, u_{yy}, uu_x at $t = 0.15$ when different smoothers are used in SDD. All of them recover U (the first row), while MLS preserves the underlying dynamics the best, i.e., the first and second order derivatives.

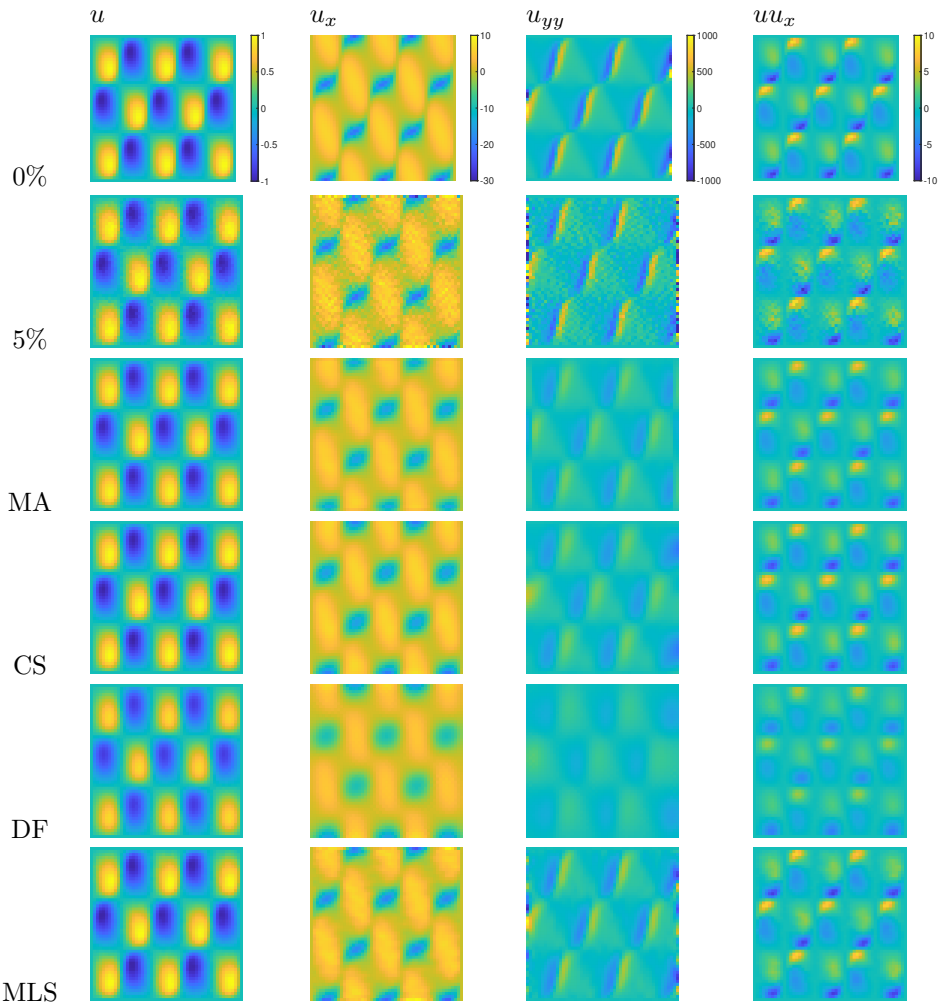

FIG. 14. *SDD results with different smoothers. The first row is the numerical solution of* (4.15) *at* $t = 0.15$ *(0% noise) with the initial condition* $u_0(x, y) = \sin(3\pi x)\sin(5\pi y)$ *and its various partial derivatives. The second row shows the noisy data and its numerical derivatives when* 5% *Gaussian noise is added to the clean data. The bottom four rows are the SDD results at* $t = 0.15$ *using MA, CS, DF, and MLS in order. While all methods recover U (the first row), the dynamics of the derivatives, especially in the third and fourth rows, are best preserved by MLS.*

5. Conclusion. This paper developed two robust methods for PDE identification from a single set of noisy data. First, we proposed a Successively Denoised Differentiation (SDD) procedure to stabilize numerical differentiation, which significantly improves the accuracy in the computation of the feature matrix from noisy data. We then proposed two new robust PDE identification algorithms called ST and SC. These algorithms utilize the Subspace Pursuit (SP) greedy algorithm to select a candidate set and then refine the results by time evolution or cross-validation. We presented various numerical experiments to demonstrate the effectiveness of both methods. SC is more computationally efficient, while ST performs better for PDEs with high order derivatives.

Appendix A. Objectives of minimization. We discuss the error representation to compare different objectives of PDE identification approaches. We consider two ways to measure errors in PDE identification. The first one is the error between the identified numerical solution $\widehat{U}$ and the exact solution u, which is given by $e(u) := \widehat{U} - u$. The second error is $e(u_t) := D_t\widehat{U} - u_t$, which measures the difference between the numerical time derivative of $\widehat{U}$ and the ground truth u_t. These two errors, $e(u)$ and $e(u_t)$, are closely related, which relations are shown below (after Table 11).

Many existing methods for the identification of PDEs or dynamical systems involve a minimization of $e(u)$ or $e(u_t)$. Consider the following decomposition of $e(u)$:

$$e(u) = \underbrace{\widehat{U} - U}_{\text{Data fidelity}} + \underbrace{U - u}_{\text{Measurement error}}, \tag{A.1}$$

where U is the given data. In (A.1), the *Data fidelity* $\widehat{U} - U$ represents the accuracy of the identified PDE in comparison with the given data U. In literature, a class of dynamic-fitting approaches such as [1, 4, 28, 38] focus on controlling the data fidelity error in order to ensure whether the numerical prediction is consistent with the evolution of the given data. The *Measurement error* $U-u$ comes from data acquisition where the given data are contaminated by noise. Denoising is an important step to reduce the measurement error.

TABLE 11

Comparison of the objectives of PDE identification. For parameter estimation problems (Type I*), the feature variables of the underlying PDEs are known. For model identification problems (Type* II*), such active set is unknown; hence sparsity is often imposed or neural network is designed.*

Problems	Objectives in minimization	Methods
Type I	Data fidelity	[1, 4, 28, 29, 38, 42]
	Regression error	[2, 3, 20, 30, 44]
	Regression error, Data fidelity	[48]
Type II	Data fidelity	[23]
	Regression error	[34, 35]
	Regression error, Data fidelity	ST (section 3.1) [17]
	Regression error, Coefficient error	SC (section 3.2)

The second error $e(u_t)$ can be expressed as

$$e(u_t) = \underbrace{D_t\widehat{U} - D_tU}_{\text{Response error}} + \underbrace{D_tU - F\widehat{\mathbf{c}}}_{\text{Regression error}} + \underbrace{F(\widehat{\mathbf{c}} - \mathbf{c}_0)}_{\text{Coefficient error}} + \underbrace{(F - F_0)\mathbf{c}_0}_{\text{System error}}, \tag{A.2}$$

where $\widehat{\mathbf{c}}$ is the estimated coefficient. The first term $D_t\widehat{U} - D_tU$ is called the *Response error*, which is the difference between the numerical derivatives of the identified PDE

and the given data. The L_2 norm of the *Regression error* $D_tU - F\widehat{\mathbf{c}}$ is the most frequently used objective function in PDE identification for the regression-based methods [2, 3, 20, 30, 44]. In addition, one can introduce various types of regularization, such as the L_1 regularization [17, 34, 35] to induce sparsity. The *coefficient error* $F(\widehat{\mathbf{c}}-\mathbf{c}_0)$ compares $\widehat{\mathbf{c}}$ and $\mathbf{c}_0$. This term vanishes when $\widehat{\mathbf{c}}-\mathbf{c}_0$ lies in the null space of F, which can occur even when $\widehat{\mathbf{c}} \neq \mathbf{c}_0$. If the initial condition of the PDE is too simple, the null space of F is very large, which makes the PDE identification problem ill-posed; see (4.12) and (4.13) for an example. In order to guarantee a successful identification, the initial condition should have sufficient variations so that F satisfies an incoherence or null space property [12]. The final term $(F-F_0)\mathbf{c}_0$ represents the *System error*, which is due to the numerical differentiation in the computation of F. Our SDD denoising technique can effectively reduce the system error.

We summarize the objectives considered by many existing methods in the literature in Table 11. These methods are categorized according to which error term(s) they aim at minimizing. As for our proposed methods, ST minimizes the data fidelity, and SC focuses on the coefficient error and the regression error.

If the numerical scheme for the computation of $D_t\widehat{U}$ is consistent, then $\|e(u)\|_\infty \to 0$ and $\|e(u_t)\|_\infty \to 0$ are equivalent as $\Delta t, \Delta x \to 0$. For $n = 0, 1, \ldots, N$, we denote $e(u)^n$ and $e(u_t)^n$ as the values of $e(u)$ and $e(u_t)$ that occurred at time $n\Delta t$, respectively. For $j = 1, 2, \ldots, N$, we have

$$\begin{aligned}\frac{e(u)^j - e(u)^{j-1}}{\Delta t} &= \frac{\widehat{U}^j - \widehat{U}^{j-1}}{\Delta t} - u_t^{j-1} + \mathbf{r}' \\ &= e(u_t)^{j-1} + \left(\frac{\widehat{U}^j - \widehat{U}^{j-1}}{\Delta t} - [D_t\widehat{U}]^{j-1}\right) + \mathbf{r}' \,,\end{aligned}$$

where $\|\mathbf{r}'\|_\infty = O(\Delta t)$. By induction, we obtain the following connection between $e(u)$ and $e(u_t)$:

$$e(u)^n = e(u)^0 + \sum_{j=0}^{n-1} e(u_t)^j \Delta t + \sum_{j=0}^{n-1}\left(\frac{\widehat{U}^{j+1} - \widehat{U}^j}{\Delta\, t} - [D_t\widehat{U}]^j\right)\Delta t + n\mathbf{r}\,, \tag{A.3}$$

where the remainder $\|\mathbf{r}\|_\infty = O(\Delta t^2)$. Equation (A.3) suggests that if the approximation $D_t\widehat{U}$ is consistent and $\|e(u)^0\|_\infty$ converges to 0 as $\Delta x \to 0$, $\|e(u)\|_\infty \to 0$ is equivalent to $\|e(u_t)\|_\infty \to 0$. Therefore, the PDE identification methods with the goal of having $\|e(u)\|_\infty$ or $\|e(u_t)\|_\infty$ approach 0 are equivalent.

It is often practical to consider a grid-dependent L_2-norm of the errors, i.e., $\|\cdot\|_{2,\Delta} = \|\cdot\|_2\sqrt{\Delta x \Delta t}$, where $\|\cdot\|_2$ denotes the ordinary L_2 vector norm. We provide an upper bound for $\|e(u)\|_{2,\Delta}$.

THEOREM A.1. *Suppose $D_t\widehat{U}$ is computed using the forward difference. Then*

$$\|e(u)\|_{2,\Delta}^2 \leq X^d T^3 \|e(u_t)\|_\infty^2 + O(\|e(u_t)\|_\infty + \Delta t) + O(\Delta t)\,. \tag{A.4}$$

Proof. Recall that $U \in \mathbb{R}^{M^d N}$ is the vectorization of the data. By the definition of the grid-dependent norm, $\|U\|_{2,\Delta}^2 = \Delta x^d \Delta t \|U\|_2^2 = \frac{X^d T}{M^d N}\|U\|_2^2$. Using (A.3), we

have

$$\begin{aligned}
\|e(u)\|_2^2 &= \|e(u)^0\|_2^2 + \sum_{n=1}^{N} \|e(u)^n\|_2^2 \\
&\le \|e(u)^0\|_2^2 + \sum_{n=1}^{N} \left(\sum_{j=0}^{n-1} \|e(u_t)^j\|_2 \right)^2 \Delta t^2 + M^d \sum_{n=1}^{N} n^2 O(\Delta t^4) \\
&+ \sum_{n=1}^{N} \|e(u)^0\|_2 \sum_{j=0}^{n-1} \|e(u_t)^j\|_2 \Delta t + M^{d/2} \sum_{n=1}^{N} \|e(u)^0\|_2 n O(\Delta t^2) \\
&+ M^{d/2} \sum_{n=1}^{N} \sum_{j=0}^{n-1} \|e(u_t)^j\|_2 n O(\Delta t^3) \\
&\le \|e(u)^0\|_2^2 + \sum_{n=1}^{N} \left(\sum_{j=0}^{n-1} \|e(u_t)^j\|_2 \right)^2 \Delta t^2 + M^d O(T^3 \Delta t) \\
&+ \|e(u)^0\|_2 \sum_{n=1}^{N} \sum_{j=0}^{n-1} \|e(u_t)^j\|_2 \Delta t + M^{d/2} \|e(u)^0\|_2 O(T^2) \\
&+ M^{d/2} \sum_{n=1}^{N} \sum_{j=0}^{n-1} \|e(u_t)^j\|_2 n O(\Delta t^3) \, .
\end{aligned}$$

Since $\|e(u_t)^j\|_2 \le M^{d/2} \|e(u_t)\|_\infty$, we can simplify the expression above as

$$\begin{aligned}
\|e(u)\|_2^2 \le{}& \|e(u)^0\|_2^2 + M^d T^2 N \|e(u_t)\|_\infty^2 + M^d O(T^3 \Delta t) \\
&+ T M^{d/2} N \|e(u)^0\|_2 \|e(u_t)\|_\infty + M^{d/2} \|e(u)^0\|_2 O(T^2) + M^d \|e(u_t)\|_\infty O(T^3) \, .
\end{aligned}$$

Thus

$$\begin{aligned}
\|e(u)\|_{2,\Delta}^2 &= \Delta x^d \Delta t \|e(u)\|_2^2 \\
&\le \Delta t \|e(u)^0\|_2^2 + X^d T^3 \|e(u_t)\|_\infty^2 + O(X^d T^3 \Delta t^2) \\
&(\|e(u_t)\|_\infty + \Delta t) \|e(u)^0\|_2 O(T^2 X^{d/2}) + X^d \|e(u_t)\|_\infty O(T^3 \Delta t) \, . \qquad \square
\end{aligned}$$

The upper bound expressed in (A.4) depends on several properties of the computational domain Ω and the sampling grid: the resolution Δt and the domain size X, T. To derive useful information from Theorem A.1, we assume that $\|e(u_t)\|_\infty = O(\Delta t)$. This condition holds, for example, when we use first order forward difference and the underlying data is noiseless.

COROLLARY A.2. *When the time-space domain is fixed, i.e., $T > 0$ and $X > 0$, if $||e(u_t)||_\infty = O(\Delta t)$, we have*

$$\|e(u)\|_{2,\Delta} \to 0 \, , \quad \Delta t, \Delta x \to 0 \, . \tag{A.5}$$

This result suggests that, with the assumptions satisfied, increasing both the time and space resolutions is a sufficient condition for controlling $\|e(u)\|_{2,\Delta} \to 0$. The convergence of $\|e(u)\|_{2,\Delta}$ as $\Delta t, \Delta x \to 0$ guarantees the success of the methods which minimize the data fidelity term, e.g., ST and IDENT in [17].

Appendix B. Proof of Proposition 3.1.

Proof. The proof is as follows:

$$
\begin{aligned}
&[D_tU]^{\mathcal{T}_2} - [F]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger [D_tU]^{\mathcal{T}_1} \\
&= [D_tU]^{\mathcal{T}_2} - [u_t]^{\mathcal{T}_2} + [u_t]^{\mathcal{T}_2} - [F]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger [D_tU]^{\mathcal{T}_1} \\
&= \underbrace{[D_tU]^{\mathcal{T}_2} - [u_t]^{\mathcal{T}_2}}_{E_1} + [u_t]^{\mathcal{T}_2} - [F]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger [u_t]^{\mathcal{T}_1} \underbrace{- [F]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger \big([D_tU]^{\mathcal{T}_1} - [u_t]^{\mathcal{T}_1}\big)}_{E_2} \\
&= [u_t]^{\mathcal{T}_2} - \big([F_0]_{\mathcal{A}}^{\mathcal{T}_2} + [F]_{\mathcal{A}}^{\mathcal{T}_2} - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big)\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger [u_t]^{\mathcal{T}_1} + E_1 + E_2 \\
&= [u_t]^{\mathcal{T}_2} - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger [u_t]^{\mathcal{T}_1} \underbrace{- \big([F]_{\mathcal{A}}^{\mathcal{T}_2} - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big)\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger [u_t]^{\mathcal{T}_1}}_{E_3} + E_1 + E_2 \\
&= \underbrace{[u_t]^{\mathcal{T}_2} - [F_0]_{\mathcal{A}_0}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_1}\big)^\dagger [u_t]^{\mathcal{T}_1}}_{=0} + \big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_1}\big)^\dagger - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big)[u_t]^{\mathcal{T}_1} \\
&+ E_1 + E_2 + E_3 \\
&= \big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_1}\big)^\dagger - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big)[u_t]^{\mathcal{T}_1} + E_1 + E_2 + E_3 \\
&= \big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_1}\big)^\dagger - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big)[u_t]^{\mathcal{T}_1} \\
&\underbrace{- [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big(\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger - \big([F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big)[u_t]^{\mathcal{T}_1}}_{E_4} + E_1 + E_2 + E_3 \\
&= \big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_1}\big)^\dagger - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big)[u_t]^{\mathcal{T}_1} + E_1 + E_2 + E_3 + E_4\ .
\end{aligned}
$$

Then we have

$$
\begin{aligned}
\mathrm{CEE}(\mathcal{A}_k;\alpha,\mathcal{T}_1,\mathcal{T}_2) \le\ & \big\|\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}_0}^{\mathcal{T}_1}\big)^\dagger - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big([F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big)[u_t]^{\mathcal{T}_1}\big\|_2 \\
&+ \big\|[D_tU]^{\mathcal{T}_2} - [u_t]^{\mathcal{T}_2}\big\|_2 + \big\|\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big\|_2 \Big(\big\|[F]_{\mathcal{A}}^{\mathcal{T}_2}\big\|_2\, \big\|[D_tU]^{\mathcal{T}_1} - [u_t]^{\mathcal{T}_1}\big\|_2 \\
&+ \big\|[F]_{\mathcal{A}}^{\mathcal{T}_2} - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big\|_2\, \big\|[u_t]^{\mathcal{T}_1}\big\|_2\Big) \\
&+ \big\|[F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big\|_2\, \big\|\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big\|_2\, \big\|\big([F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big\|_2\, \big\|[F]_{\mathcal{A}}^{\mathcal{T}_1} - [F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big\|_2\, \big\|[u_t]^{\mathcal{T}_1}\big\|_2\ .
\end{aligned}
$$

In the last term on the right-hand side of the inequality, we applied the norm bound in Theorem 4.1 of [45]. Then by setting

$$
\begin{aligned}
g(\mathcal{A};\alpha,\mathcal{T}_1,\mathcal{T}_2) = \ & \big\|[D_tU]^{\mathcal{T}_2} - [u_t]^{\mathcal{T}_2}\big\|_2 + \big\|\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big\|_2 \Big(\big\|[F]_{\mathcal{A}}^{\mathcal{T}_2}\big\|_2\, \big\|[D_tU]^{\mathcal{T}_1} - [u_t]^{\mathcal{T}_1}\big\|_2 \\
&+ \big\|[F]_{\mathcal{A}}^{\mathcal{T}_2} - [F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big\|_2\, \big\|[u_t]^{\mathcal{T}_1}\big\|_2\Big) \\
&+ \big\|[F_0]_{\mathcal{A}}^{\mathcal{T}_2}\big\|_2\, \big\|\big([F]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big\|_2\, \big\|\big([F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big)^\dagger\big\|_2\, \big\|[F]_{\mathcal{A}}^{\mathcal{T}_1} - [F_0]_{\mathcal{A}}^{\mathcal{T}_1}\big\|_2\, \big\|[u_t]^{\mathcal{T}_1}\big\|_2,
\end{aligned} \tag{B.1}
$$

we prove the theorem. ∎

REFERENCES

[1] E. Baake, M. Baake, H. Bock, and K. Briggs, *Fitting ordinary differential equations to chaotic data*, Phys. Rev. A, 45 (1992), 5524.

[2] M. Bär, R. Hegger, and H. Kantz, *Fitting partial differential equations to space-time dynamics*, Phys. Rev. E, 59 (1999), 337.

[3] H. G. Bock, *Numerical treatment of inverse problems in chemical reaction kinetics*, in Modelling of Chemical Reaction Systems, Springer, Berlin, Heidelberg 1981, pp. 102–125.

[4] H. G. Bock, *Recent advances in parameter identification techniques for ODE*, in Numerical Treatment of Inverse Problems in Differential and Integral Equations, Birkhäuser Boston, Boston, 1983, pp. 95–121.

[5] J. Bongard and H. Lipson, *Automated reverse engineering of nonlinear dynamical systems*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 9943–9948.

[6] M. Bongini, M. Fornasier, M. Hansen, and M. Maggioni, *Inferring interaction rules from observations of evolutive systems* I: *The variational approach*, Math. Models Methods Appl. Sci., 27 (2017), pp. 909–951.

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

[8] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[9] P. Craven and G. Wahba, *Smoothing noisy data with spline functions*, Numer. Math., 31 (1978), pp. 377–403.

[10] W. Dai and O. Milenkovic, *Subspace pursuit for compressive sensing signal reconstruction*, IEEE Trans. Inform. Theory, 55 (2009), pp. 2230–2249.

[11] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.

[12] D. L. Donoho and X. Huo, *Uncertainty principles and ideal atomic decomposition*, IEEE Trans. Inform. Theory, 47 (2001), pp. 2845–2862.

[13] D. R. Gurevich, P. A. Reinbold, and R. O. Grigoriev, *Robust and optimal sparse regression for nonlinear PDE models*, Chaos, 29 (2019), 103113.

[14] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes,* III, in Upwind and High-resolution Schemes, Springer, Berlin, Heidelberg, 1987, pp. 218–290.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, 2009.

[16] E. Kaiser, J. N. Kutz, and S. L. Brunton, *Sparse identification of nonlinear dynamics for model predictive control in the low-data limit*, Proc. A., 474 (2018), 20180335.

[17] S. H. Kang, W. Liao, and Y. Liu, *IDENT: Identifying Differential Equations with Numerical Time Evolution*, preprint, https://arxiv.org/abs/1904.03538, 2019.

[18] Y. Khoo and L. Ying, *SwitchNet: A Neural Network Model for Forward and Inverse Scattering Problems*, preprint, https://arxiv.org/abs/1810.09675, 2018.

[19] P. Lancaster and K. Salkauskas, *Surfaces generated by moving least squares methods*, Math. Comp., 37 (1981), pp. 141–158.

[20] H. Liang and H. Wu, *Parameter estimation for differential equation models using a framework of measurement error in regression models*, J. Amer. Statist. Assoc., 103 (2008), pp. 1570–1583.

[21] J.-C. Loiseau and S. L. Brunton, *Constrained sparse Galerkin regression*, J. Fluid Mech., 838 (2018), pp. 42–67.

[22] Z. Long, Y. Lu, and B. Dong, *PDE-Net* 2.0*: Learning PDEs from data with a numeric-symbolic hybrid deep network*, J. Comput. Phys., 399 (2019), 108925.

[23] Z. Long, Y. Lu, X. Ma, and B. Dong, *PDE-Net: Learning PDSs from Data*, preprint, https://arxiv.org/abs/1710.09668, 2017.

[24] F. Lu, M. Zhong, S. Tang, and M. Maggioni, *Nonparametric inference of interaction laws in systems of agents from trajectory data*, Proc. Natl. Acad. Sci. USA, 116 (2019), pp. 14424–14433.

[25] B. Lusch, J. N. Kutz, and S. L. Brunton, *Deep learning for universal linear embeddings of nonlinear dynamics*, Nat. Commun., 9 (2018), 4950.

[26] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor, *Model selection for dynamical systems via sparse regression and information criteria*, Proc. A, 473 (2017), 20170009.

[27] D. A. Messenger and D. M. Bortz, *Weak SINDy for Partial Differential Equations*, preprint, https://arxiv.org/abs/2007.02848, 2020.

[28] T. Müller and J. Timmer, *Parameter identification techniques for partial differential equations*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 14 (2004), pp. 2053–2060.

[29] T. G. Müller and J. Timmer, *Fitting parameters in partial differential equations from partially observed noisy data*, Phys. D, 171 (2002), pp. 1–7.

[30] U. Parlitz and C. Merkwirth, *Prediction of spatiotemporal time series based on reconstructed local states*, Phys. Rev. Lett., 84 (2000), 1890.

[31] T. Qin, K. Wu, and D. Xiu, *Data driven governing equations approximation using deep neural networks*, J. Comput. Phys., 395 (2019), pp. 620–635.

[32] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, J. Comput. Phys., 357 (2018), pp. 125–141.
[33] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations*, preprint, https://arxiv.org/abs/1711.10561, 2017.
[34] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Data-driven discovery of partial differential equations*, Sci. Adv., 3 (2017), e1602614.
[35] H. Schaeffer, *Learning partial differential equations via data discovery and sparse optimization*, Proc. A, 473 (2017), 20160446.
[36] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, *Sparse dynamics for partial differential equations*, Proc. Natl. Acad. Sci. USA, 110 (2013), pp. 6634–6639.
[37] H. Schaeffer, G. Tran, and R. Ward, *Extracting sparse high-dimensional dynamics from limited data*, SIAM J. Appl. Math., 78 (2018), pp. 3279–3295, https://doi.org/10.1137/18M116798X.
[38] M. Schmidt and H. Lipson, *Distilling free-form natural laws from experimental data*, Science, 324 (2009), pp. 81–85.
[39] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, San Diego, 1997.
[40] M. Tham, *Dealing with Measurement Noise: Moving Average Filter*, Chemical Engineering and Advanced Materials, University of Newcastle upon Tyne, Tyne, UK, 1998.
[41] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Statist. Soc. Ser. B, 58 (1996), pp. 267–288.
[42] J. Timmer, T. Müller, and W. Melzer, *Numerical methods to determine calcium release flux from calcium transients in muscle cells*, Biophys. J., 74 (1998), pp. 1694–1707.
[43] G. Tran and R. Ward, *Exact recovery of chaotic systems from highly corrupted data*, Multiscale Model. Simul., 15 (2017), pp. 1108–1129, https://doi.org/10.1137/16M1086637.
[44] H. U. Voss, P. Kolodner, M. Abel, and J. Kurths, *Amplitude equations from spatiotemporal binary-fluid convection data*, Phys. Rev. Lett., 83 (1999), 3422.
[45] P.-Å. Wedin, *Perturbation theory for pseudo-inverses*, BIT Numer. Math., 13 (1973), pp. 217–232.
[46] H. Wendland, *Local polynomial reproduction and moving least squares approximation*, IMA J. Numer. Anal., 21 (2001), pp. 285–300.
[47] A. P. Witkin, *Scale-space filtering*, in Readings in Computer Vision, Morgan Kaufmann, San Francisco, 1987, pp. 329–332.
[48] X. Xun, J. Cao, B. Mallick, A. Maity, and R. J. Carroll, *Parameter estimation of partial differential equation models*, J. Amer. Statist. Assoc., 108 (2013), pp. 1009–1020.
[49] M. M. Zhang, H. Lam, and L. Lin, *Robust and parallel Bayesian model selection*, Comput. Statist. Data Anal., 127 (2018), pp. 229–247.