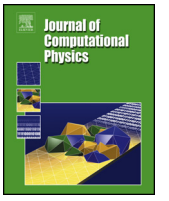
Journal of Computational Physics 483 (2023) 112069

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp



WeakIdent: Weak formulation for identifying differential equation using narrow-fit and trimming ☆

Mengyi Tang *, Wenjing Liao [1], Rachel Kuske [2], Sung Ha Kang [3]

School of Mathematics, Georgia Institute of Technology, 686 Cherry Street, Atlanta, 30032, GA, USA

ARTICLE INFO

Article history:
Received 14 November 2022
Received in revised form 8 March 2023
Accepted 9 March 2023
Available online 16 March 2023

Dataset link: https://github.com/sunghakang/WeakIdent

Keywords:
Partial differential equation identification
Data-driven model selection
Sparse recovery
Subspace persuit

ABSTRACT

Data-driven identification of differential equations is an interesting but challenging problem, especially when the given data are corrupted by noise. When the governing differential equation is a linear combination of various differential terms, the identification problem can be formulated as solving a linear system, with the feature matrix consisting of linear and nonlinear terms multiplied by a coefficient vector. This product is equal to the time derivative term, and thus generates dynamical behaviors. The goal is to identify the correct terms that form the equation to capture the dynamics of the given data. We propose a general and robust framework to recover differential equations using a weak formulation with two new mechanisms, narrow-fit and trimming, for both ordinary and partial differential equations (ODEs and PDEs). The weak formulation facilitates an efficient and robust way to handle noise, and two new mechanisms, narrow-fit and trimming, improve the coefficient support and value recoveries respectively. For each sparsity level, Subspace Pursuit is utilized to find an initial set of support from the large dictionary. Then, we focus on highly dynamic regions (rows of the feature matrix), and error normalize the feature matrix in the narrow-fit step. The support is further updated via trimming the terms that contribute the least. Finally, the support set of features with the smallest Cross-Validation error is chosen as the result. A comprehensive set of numerical experiments are presented for both systems of ODEs and PDEs with various noise levels. The proposed method gives a robust recovery of the coefficients, and a significant denoising effect which can handle up to 100% noise-to-signal ratio for some equations. We compare the proposed method with several state-of-the-art algorithms for the recovery of differential equations.

☆ We provide the code of WeakIdent at http://github.com/sunghakang/WeakIdent.

* Corresponding author.
E-mail addresses: tangmengyi@gatech.edu (M. Tang), wliao60@gatech.edu (W. Liao), rachel@math.gatech.edu (R. Kuske), kang@math.gatech.edu (S.H. Kang).

[1] This work was partially funded by NSF 2145167.
[2] This work was partially funded by NSF-CMMI 2009270.
[3] This work was partially funded by Simons Foundation grant 584960.

https://doi.org/10.1016/j.jcp.2023.112069
0021-9991/

1. Introduction

In recent years, there has been an increasing interest in discovering physical or biological dynamics from complex data. The discovery of differential equations can offer important insights into contemporary neuroscience [1], fluid mechanics, physical systems [2,3], and biology [4].

In this paper, we focus on the inverse problem of identifying a differential equation corresponding to given data corrupted by noise. Given a time-dependent discrete data set, we aim to discover the underlying equation of the form

$$\partial_t u = f(u, \partial_{\boldsymbol{x}} u, \ldots, \partial_{\boldsymbol{x}}^k u, \ldots, u^2, \partial_{\boldsymbol{x}} u^2, ..., \partial_{\boldsymbol{x}}^k u^2, \ldots, u^3, \partial_{\boldsymbol{x}} u^3, ..., \partial_{\boldsymbol{x}}^k u^3, \ldots) \tag{1}$$

where each differential term in the right hand side of (1) is called a feature in this prescribed dictionary. In particular, f is called the governing equation of (1). We assume that f in (1) is a linear combination of the features, so that this inverse problem becomes the identification of a sparse coefficient vector where both the support and the values of this coefficient vector are unknown. Since the features include linear and nonlinear terms, this f in (1) includes nonlinear differential equations. This model identification problem is very challenging when the given data are corrupted by noise.

Parameter identification in differential equations and dynamical systems has been studied by scientists in various fields. Earlier works include [5–7,2,8,3], where the differential equation (1) is considered in [6,8], symbolic regression is used in [2,3], and an optimization approach is taken in [5,6,8]. In recent years, sparse regression is incorporated into the model identification problem to promote sparsity in the coefficient recovery [9–24]. Representative works include Sparse Identification of Nonlinear Dynamics (SINDy) [9–12], Identifying Differential Equations with Numerical Time evolution (IDENT) [16,17], Weak SINDy [23,24], RGG [25] and many others [18,26,19,27]. The PDE and dynamics identification problem is also addressed by deep learning approaches [28–34].

The majority of existing works apply sparse regression on a linear system formed from (1) with differential features [9–12,16–18,20]. From the given data, differential features are approximated via numerical differentiation. When the given data contain noise, a denoising step is applied before numerical differentiation. Least-squares moving average is applied in [16], successively denoised differentiation is proposed in [17] and regularization is used in [35]. In terms of sparse regression, L_1 or regularized L_1 minimization has been widely used [9,16,18,26]; Sequentially thresholded least-squares is used in [11,13–15]; Greedy algorithms are used in [17]. More generally, the coefficients are allowed to be spatially dependent in [36,16], and the Group Lasso is used to promote group sparsity where each group represents a feature, which is also used for varying coefficient case in [16]. While these methods using differential features give good results, numerical differentiation can be unstable for high-order features, and the coefficient recovery may not be robust when the given data is corrupted by noise.

Recent progress using a weak/integral formulation [22,25,23,24] shows improvements in the robustness of the sparse coefficient identification. A weak form for (1) with a set of test functions gives rise to a linear system with integral features instead of differential features. Noise is tackled through the weak form, since a proper test function gives a denoising effect. The test functions are chosen to be localized smooth functions vanishing on the boundaries, thus resembling kernel functions commonly used in kernel denoising methods. It is shown in [23,24] that using the weak form with the standard sequentially threshold least-squares algorithm gives rise to superior numerical performance. Differential equations with high-order derivatives, including the Korteweg–De Vries (KdV) equation, the Kuramoto–Sivashinsky (KS) equation, and 2D reaction-diffusion equations can be recovered even with a significant amount of noise. A related work [37] focuses on the identification of advection-diffusion equations, and shows that a Galerkin-type algorithm using the weak form outperforms the collocation-type algorithm using a differential form.

In this paper, we propose a Weak formulation for Identification of Differential Equations with Narrow-fit and Trimming (WeakIDENT). To recover (1) where f is a linear combination of various differential terms, we construct a linear system: the feature matrix consisting of linear and nonlinear terms called features, multiplied by a coefficient vector, is set equal to the time derivative. We use the term coefficient support to refer to a collection of nonzero components in the coefficient vector, such that the linear system is composed of the collection of features that contribute to the dynamics represented by the data. For the weak formulation, we follow the derivation proposed in [23]. For our sparse coefficient recovery, we perform an iterative greedy support identification scheme as in [17] to find the support which gives the collection of linear and nonlinear differential terms. For each sparsity level, we use the Subspace Pursuit (SP) algorithm [38] to first find the initial guess of the coefficient support. We propose new narrow-fit and trimming steps which improve the support selection as well as coefficient value recovery. Among different sparsity results, we choose the one with the minimum Cross-Validation (CV) error as the final result. For Cross-Validation, we randomly separate the given data in half, use one set to find the coefficients, then use this coefficient vector with the other set of data to compute the error. We provide an error analysis in Theorem 1 to show that the error in the linear system under the weak form is significantly smaller than that under the differential form.

Our contributions can be summarized as follows:

1. Proposing WeakIDENT to robustly identify differential equations in (1) from highly corrupted noisy data. The weak form proposed in [23] allows us to move the derivative to the test functions and facilitates robustness against noise. We propose two new and novel mechanisms, narrow-fit and trimming, to improve the coefficient support value and the

coefficient support recovery, respectively. These mechanisms utilize a column-wise error normalization to improve the robustness of the coefficient recovery. Narrow-fit focuses on highly dynamic regions to reduce the size of the feature matrix, and trimming the features with small contributions to the result further contributes to the improvement.
2. We provide comprehensive numerical experiments for ordinary differential systems (ODEs) and partial differential equations (PDEs), and compare with existing methods such as [16,17,22–24].

We organize this paper as follows. In Section 2, we state the identification problem of differential equations, and give details about the feature formulation in the weak form, the choice of test functions, and provide an error analysis of the weak form. We present our WeakIDENT Algorithm in Section 3 with the details of error normalization, selection of highly dynamic regions with dominant contributions to the identification, and trimming of the features with the least contribution to the support. A comprehensive set of numerical experiments is provided in Section 4, including various comparisons against state-of-the-art algorithms. We conclude the paper in Section 5, and provide additional experiments in Supplementary Material 5.

2. Problem set-up, weak formulation and error analysis

In this section, we state the identification problem for differential equations and formulate a linear system in a weak form. We also discuss the choice of test functions and provide an error analysis of the weak formulation.

2.1. Problem set-up

We present the identification problem with one spatial variable for simplicity. It can be easily extended to multi-variables, and numerical results are provided for the multi-variable case. We consider a spatial-temporal domain $\Omega = [X_1, X_2] \times [0, T]$ with $X_1 < X_2$ and $T > 0$. We assume a set of discrete time-dependent noisy data is given:

$$\mathcal{D} = \{\hat{U}_i^n | i = 1, 2, ..., \mathbb{N}_x; n = 1, ..., \mathbb{N}_t\} \in \mathbb{R}^{\mathbb{N}_x \times \mathbb{N}_t}, \tag{2}$$

where $\mathbb{N}_x$, and $\mathbb{N}_t \in \mathbb{N}$ are the size of discretization in spatial and temporal dimension respectively. The data point $\hat{U}_i^n$ is an approximation to the true solution of a differential equation

$$\hat{U}_i^n \approx u(x_i, t^n) \text{ for } (x_i, t^n) \in \Omega,$$

at the spatial location $x_i = i\Delta x \in [X_1, X_2]$ and $t^n = n\Delta t \in [0, T]$. Here $\Delta x = (X_2 - X_1)/(\mathbb{N}_x - 1)$ and $\Delta t = T/(\mathbb{N}_t - 1)$. In the noisy case, we express the noisy data $\hat{U}_i^n$ in terms of the clean data $U_i^n = u(x_i, t^n)$ as:

$$\hat{U}_i^n = U_i^n + \epsilon_i^n, \tag{3}$$

where ϵ_i^n represents the noise at (x_i, t^n). The objective is to identify a differential equation in the form of (1) from the given data (2).

We assume that the governing equation f in (1) is a linear combination of linear and nonlinear terms including the derivatives of u. This covers a vast range of ODEs and PDEs in applications, e.g., the Lorenz equation, the Lotka-Volterra equation, transport equations, Burgers' equation, the heat equation, the KS equation, the KdV equation, and reaction-diffusion equations. In this paper, to utilize the weak form, we consider the function f to be a linear combination of different derivatives of powers of u:

$$\frac{\partial u}{\partial t}(x, t) = \sum_{l=1}^{L} c_l F_l \quad \text{with} \quad F_l = \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}} f_l, \text{ where } f_l = f_l(u) = u^{\beta_l}. \tag{4}$$

The l^{th} feature $F_l(u)$ represents the α_l^{th} spatial derivative of the monomial $f_l = f_l(u) = u^{\beta_l}$ for some nonnegative integer β_l. Let the highest order of derivative be $\bar{\alpha}$ such that $\alpha_l \in \{0, \ldots, \bar{\alpha}\}$, and the highest order of monomial be $\bar{\beta}$ such that $\beta_l \in \{0, \ldots, \bar{\beta}\}$. We use L to denote the total number of features in the dictionary, which depends on $\bar{\alpha}$ and $\bar{\beta}$, since it includes all combinations. The formulation of (4) has the advantage in accurate feature approximation particularly for the weak form, since integration by parts moves the derivatives to the test function. When the spatial domain is multi-dimensional, we consider f_l as monomials in the multivariable case, and we allow F_l to be partial derivatives of f_l across different spatial dimensions.

In (4), the coefficient can be considered as a sparse vector

$$\boldsymbol{c} = (c_1, ..., c_L)^T \in \mathbb{R}^L \tag{5}$$

which parametrizes the differential equation. The objective of this paper is to recover the differential equation from the given noisy data set $\mathcal{D}$ (2), by finding a sparse coefficient vector $\boldsymbol{c}$ (5) of the linear system (1).

2.2. The weak formulation

The weak formulation of (4) is

$$\int_{\Omega_{h(x_i,t^n)}} \phi_{h(x_i,t^n)}(x,t)\frac{\partial u(x,t)}{\partial t}dxdt = \sum_{l=1}^{L} c_l \int_{\Omega_{h(x_i,t^n)}} \phi_{h(x_i,t^n)}(x,t)F_l dxdt, \tag{6}$$

where the test function $\phi_h(x,t)$ is locally defined on a region $\Omega_{h(x_i,t^n)}$, which is centered at (x_i,t^n) and indexed by h. Specifically, each test function $\phi_h(x,t)$ is a translation of a fixed function $\phi(x,t)$ such that $\phi_{h(x_i,t^n)}(x,t) = \phi(x-x_i, t-t^n)$. Integration by parts of (6) gives rise to

$$-\int_{\Omega_{h(x_i,t^n)}} u(x,t)\frac{\partial \phi_h(x,t)}{\partial t}dxdt = \sum_{l=1}^{L} c_l \int_{\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} u^{\beta_l}\frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}dxdt, \tag{7}$$

as long as ϕ_h and its derivatives up to order $\bar{\alpha}$ vanish on the boundary of $\Omega_{h(x_i,t^n)}$. The l^{th} term

$$\int_{\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l} u^{\beta_l}\frac{\partial^{\alpha_l}\phi_h(x,t)}{\partial x^{\alpha_l}}dxdt$$

is the l^{th} integral feature with the test function ϕ_h. Since the test function is smooth, the numerical integration can be carried out with higher order accuracy. With numerical integration, we obtain the following discrete linear system for WeakIdent:

$$\boldsymbol{W}\boldsymbol{c} = \boldsymbol{b} \tag{8}$$

where

$$\boldsymbol{W} = (w_{h(x_i,t^n),l}) \in \mathbb{R}^{H\times L}, \quad \boldsymbol{c} = (c_l) \in \mathbb{R}^L, \quad \text{and} \quad \boldsymbol{b} = (b_{h(x_i,t^n)}) \in \mathbb{R}^H,$$

for

$$w_{h(x_i,t^n),l} = \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} (-1)^{\alpha_l}\hat{U}_j^k \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi_h(x_j,t^k)\Delta x\Delta t \quad \text{and} \quad b_{h(x_i,t^n)} = -\sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \hat{U}_j^k \frac{\partial \phi_h(x_j,t^k)}{\partial t}\Delta x\Delta t. \tag{9}$$

Here the numerical integration is computed with the data points $(x_j,t^k) \in \Omega_{h(x_i,t^n)}$, and $w_{h(x_i,t^n),l}$ represents an approximation of the integral of the feature F_l in the integral region $\Omega_{h(x_i,t^n)}$ centered at (x_i,t^n). The numerical integration is computed from $\mathbb{N}_x\mathbb{N}_t$ grid points.

For the test function, we follow the derivation and use $\phi(x,t)$ as in [23]:

$$\phi(x,t) = \left(1-\left(\frac{x}{m_x\Delta x}\right)^2\right)^{p_x}\left(1-\left(\frac{t}{m^t\Delta t}\right)^2\right)^{p_t}, \quad (x,t)\in\Omega_{h(x_i,t^n)} \tag{10}$$

for $i = 1,..,\mathbb{N}_x, n = 1,...,\mathbb{N}_t$ where p_x and p_t give the smoothness of ϕ in terms of x and t. The test function satisfies $\int_{\Omega_{h(x_i,t^n)}}\phi(x,t)dxdt = 1$ and $\phi(x,t) = 0$ for $(x,t)\in\partial\Omega_{h(x_i,t^n)}$, with $\phi(x,t)$ localized around (x_i,t^n) and is supported on $\Omega_{h(x_i,t^n)} = [x_i - m_x\Delta x, x_i + m_x\Delta x]\times[t^n - m_t\Delta t, t^n + m_t\Delta t]$ for some positive integers m_x and m_t. The weak features $w_{h(x_i,t^n)}$ in (9) can be written into a convolution form $U * \frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi$ and calculated through Fast Fourier Transform in terms of $\mathcal{F}^{-1}\left(\mathcal{F}(U)\circ\mathcal{F}\left(\frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi\right)\right)$, where $\circ$ denotes point-wise multiplication, and p_x, p_t, m_x and m_t are carefully chosen to give a denoising effect depending on the frequency of the given data as in [23]. For the completeness, more details are presented in Supplementary Material 5.3.

The weak form (8) has $\mathbb{N}_x\mathbb{N}_t$ rows. For computational efficiency, we subsample $\boldsymbol{W}$ to

$$H = N_xN_t \le \mathbb{N}_x\mathbb{N}_t, \tag{11}$$

rows by uniformly subsampling N_x and N_t points in space and time respectively. Then, we consider highly dynamic regions to further reduce the size of $\boldsymbol{W}$ and $\boldsymbol{b}$ for an improved coefficient recovery (details in Subsection 3.2). In comparison, random subsampling is used in [25] for sparse regression, and regions with large gradients in time are considered in [24].

2.3. Error analysis of the weak formulation

We next analyze the approximation error of the weak formulation in (7). Suppose the given noisy data $\mathcal{D}$ (2) has mean-zero i.i.d. Gaussian noise, $\mathbb{E}[\epsilon_i^n] = 0$, and $\mathrm{Var}(\epsilon_i^n) = \sigma^2$. Let c_l be the l^{th} true coefficient in the true support Supp^*. The associated integral formulation using the test function (10) with the true coefficients from the true support becomes

$$\int_{\Omega_h} u(x,t)\frac{\partial \phi_h(x,t)}{\partial t}dxdt + \sum_{l\in \mathrm{Supp}^*}(-1)^{\alpha_l}c_l\int_{\Omega_h} f_l(x,t)\frac{\partial^{\alpha_l}\phi_h(x,t)}{\partial x^{\alpha_l}}dxdt = 0. \tag{12}$$

We next analyze the error for the discretized system in (8) using the noisy data $\{\hat{U}_i^n\}$, approximating the true equation (12). The h^{th} row of the linear system (8) is obtained from the weak form with the test function ϕ_h. The error for the discretized system in (8) is defined as

$$\boldsymbol{e} = \boldsymbol{W}\boldsymbol{c} - \boldsymbol{b} \tag{13}$$

where the row-wise error is

$$e_h = \sum_{l\in \mathrm{Supp}^*}\sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}}(-1)^{\alpha_l}c_l\hat{U}_j^k\frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi_h(x_j,t^k)\Delta x\Delta t + \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}}\hat{U}_j^k\frac{\partial\phi_h}{\partial t}(x_j,t^k)\Delta x\Delta t.$$

We decompose the error as

$$\boldsymbol{e} = \boldsymbol{e}^{\text{int}} + \boldsymbol{e}^{\text{noise}} \tag{14}$$

where

$$e_h^{\text{noise}} = e_h - \left(\sum_{l\in \mathrm{Supp}^*}\sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}}(-1)^{\alpha_l}c_lU_j^k\frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi_h(x_j,t^k)\Delta x\Delta t + \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}}U_j^k\frac{\partial\phi_h}{\partial t}(x_j,t^k)\Delta x\Delta t\right)$$

$$e_h^{\text{int}} = \left(\sum_{l\in \mathrm{Supp}^*}\sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}}(-1)^{\alpha_l}c_lU_j^k\frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}\phi_h(x_j,t^k)\Delta x\Delta t + \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}}U_j^k\frac{\partial\phi_h}{\partial t}(x_j,t^k)\Delta x\Delta t\right)$$
$$-\left(\sum_{l\in \mathrm{Supp}^*}c_l\int_{\Omega_{h(x_i,t^n)}}(-1)^{\alpha_l}u^{\beta_l}\frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}dxdt + \int_{\Omega_{h(x_i,t^n)}}u(x,t)\frac{\partial\phi_h(x,t)}{\partial t}dxdt.\right).$$

In this decomposition, $\boldsymbol{e}^{\text{int}}$ represents the numerical integration error of the noise-free data U. It has been shown in [23] that $\boldsymbol{e}^{\text{int}} = O((\Delta x\Delta t)^{q+1})$, where q is the order of the numerical integration as in [23], if the decay of test function ϕ near the boundary of the test region satisfies $\max\{\phi(1-1/m_x,0),\phi(0,1-1/m_t)\} \le (\frac{2\max\{m_x,m_t\}-1}{\max\{m_x,m_t\}^2})^{q+1}$.

The following Theorem 1 provides an estimate of the error $\boldsymbol{e}^{\text{noise}}$ arising from noise.

Theorem 1. *Consider a dynamical system*

$$u_t = \sum_{l\in Supp^*}c_l\frac{\partial^{\alpha_l}}{\partial x^{\alpha_l}}u^{\beta_l}$$

of one spatial variable where $Supp^*$ denotes the true support of the underlying differential equation. Assume the noise ϵ_i^n are i.i.d. and satisfies $\mathbb{E}[\epsilon_i^n] = 0$, $\mathrm{Var}(\epsilon_i^n) = \sigma^2$, and $|\epsilon_i^n| \le \epsilon$ for all i and n. Each test region $\Omega_{h(x_i,t^n)}$ for $i = 1,\ldots,\mathbb{N}_x$, $n = 1,\ldots,\mathbb{N}_t$ has area $|\Omega_h| = m_xm_t\mathbb{N}_x\mathbb{N}_t$. Then,

1. *In (14), the error from noise $\boldsymbol{e}^{\text{noise}}$ for the discretized system satisfies*

$$\|\boldsymbol{e}^{\text{noise}}\|_\infty \le \bar{S}^*|\Omega_h|\epsilon + O\left(\epsilon^2\right) \tag{15}$$

with a constant

$$\bar{S}^* = \max_h \sup_{(x_j,t^k)\in\Omega_h}\left|\sum_{l\in Supp^*}(-1)^{\alpha_l}c_l\beta_l(U_j^k)^{\beta_l-1}\frac{\partial^{\alpha_l}\phi}{\partial x^{\alpha_l}}(x_j,t^k) - \frac{\partial\phi}{\partial t}(x_j,t^k)\right|. \tag{16}$$

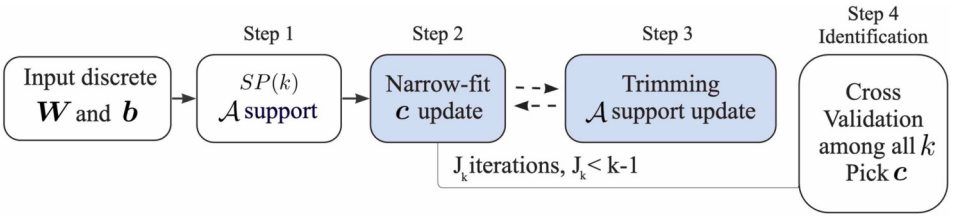


Fig. 1. WeakIdent flowchart: Input weak formulation $\boldsymbol{W}$ and $\boldsymbol{b}$ in (8) subsampled as (11). [Step 1] SP for a given sparsity k gives the first candidate of coefficient support $\mathcal{A}_0^k$. [Step 2] Narrow-fit and [Step 3] Trimming improves the coefficient values $\boldsymbol{c}(k, j)$ and support $\mathcal{A}_j^k$. Steps 2 and 3 are iterated at most $k-1$ times. Finally, in [Step 4] the result $\boldsymbol{c}(k^*, J_{k^*})$ with the minimum Cross Validation among all different sparsity level k give the identification of the differential equation.

2. *The leading error in* e_h^{noise} *(that is linear in noise) for the test function* ϕ_h *has mean* 0 *and variance* $\sigma^2 S_h^*$ *where*

$$S_h^* = \Delta x \Delta t \sum_{(x_j,t^k)\in\Omega_{h(x_i,t^n)}} \left(\sum_{l\in Supp^*} (-1)^{\alpha_l} c_l \beta_l (U_j^k)^{\beta_l - 1} \frac{\partial^{\alpha_l}\phi_h}{\partial x^{\alpha_l}}(x_j,t^k) + \frac{\partial \phi_h}{\partial t}(x_j,t^k) \right)^2 . \tag{17}$$

Theorem 1 is proved in Supplementary Material 5.1. In summary, we prove that the error $\boldsymbol{e}$ in (13) for the discretized linear system under the weak formulation satisfies the following upper bound

$$\|\boldsymbol{e}\|_\infty \le O((\Delta x \Delta t)^{q+1}) + \bar{S}^* |\Omega_h| \epsilon + O(\epsilon^2) \tag{18}$$

where q is the order of the numerical integration as in [23]. By comparison, the error for the discretized system under the differential form [16] is on the order of

$$O\left(\Delta t + \Delta x^{p+1-r} + \frac{\epsilon}{\Delta t} + \frac{\epsilon}{\Delta x^r}\right), \tag{19}$$

where r is the highest order of derivatives for the features in the true support, and the numerical differentiation is carried by interpolating the data by a pth order polynomial. By comparing (18) and (19), we observe that the error for the discretized linear system in the weak form is significantly smaller than the error in the differential form.

3. WeakIdent algorithm

In this section, we present the details of the proposed Weak formulation for Identifying Differential Equation using Narrow-fit and Trimming (WeakIdent) model. There are mainly four steps to the algorithm: After the system is set-up as in (8),

[Step 1] For each sparsity level k, we use Subspace Pursuit (SP) [38] to find an initial choice of support $\mathcal{A}_0^k$ from the dictionary of L features. SP finds the choice with the minimum residual from a column-wise normalized (21) linear system as in [17].

[Step 2] Narrow-fit. To recover the coefficient value using the support $\mathcal{A}_j^k$, we (i) identify highly dynamic regions of certain features of interest; (ii) normalize the reduced feature matrix according to the leading error term, then (iii) determine a coefficient value vector $\boldsymbol{c}(k, j)$ from this reduced narrow system (We set $j = 0$ on the first iteration).

[Step 3] Trimming. With the updated coefficient values $\boldsymbol{c}(k, j)$ in [Step 2], we identify a single feature with the least contribution to f. If the contribution score is less than a preset trimming parameter $\mathcal{T}$, we trim the corresponding coefficient. This trimming yields a new updated support $\mathcal{A}_j^k$. We iterate [Step 2] and [Step 3], with increment j, until no change is made to $\mathcal{A}_j^k$ at $j = J_k$.

[Step 4] Cross Validation. With the final support $\mathcal{A}_{J_k}^k$ and coefficient value vector $\boldsymbol{c}(k, J_k)$ for each different sparsity level k, we select the one $\boldsymbol{c}(k^*, J_{k^*})$ with the minimum Cross-Validation error (30) as the final result.

A schematic of the algorithm is given in Fig. 1. From the weak form input $\boldsymbol{W}$ and $\boldsymbol{b}$, for a fixed sparsity level k, SP is used to find the initial set of support $\mathcal{A}_0^k$. Then [Step 2] Narrow-fit and [Step 3] Trimming are iterated until the support does not change, where the number of iterations is at most $k-1$. Here we use $\boldsymbol{c}(k, j)$ to indicate the coefficient vector for the sparsity level k and j iteration. The cross validation is used to select the optimal solution $\boldsymbol{c}(K, J_K)$ among all $k \le L$.

We present the details in the following subsections. In [Step 2], we normalize each column of the feature matrix according to its leading error term, to balance the effect of noise perturbations across the features. The details for this error

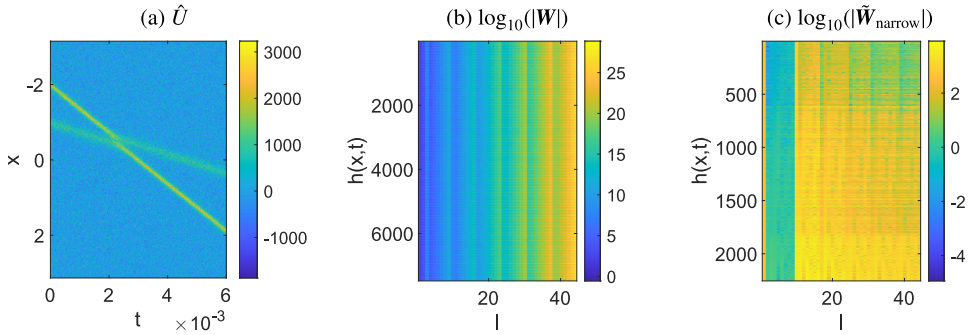


Fig. 2. Error normalization: (a) The given noisy data $\hat{U}$ with $\sigma_{\text{NSR}} = 0.5$ in $x-t$ plane. (b) The entry-wise magnitude of the matrix $\boldsymbol{W}$. (c) The matrix $\tilde{\boldsymbol{W}}_{\text{narrow}}$ in (23). We use log 10 scale in (b) and (c). The difference in scale has been reduced approximately from 10^{29} in the unnormalized matrix (b) to 10^{6} after normalization in (c). Our error normalization results in more uniform entry values with less variance across different columns.

normalization of the feature matrix are given in Subsection 3.1. We detail the implementation of Narrow-fit using the highly dynamic regions in Subsection 3.2. In [Step 3], we trim the support removing features with contributions below a threshold, as described in detail in Subsection 3.3. The algorithm is summarized Subsection 3.4.

3.1. Column-wise error normalized matrix

We use least squares for coefficient recovery. The accuracy of least squares is highly dependent on the conditioning of the feature matrix [39,40]. In this paper, we utilize two types of normalization for the columns of the feature matrix to improve the coefficient recovery. For the linear system (8), we introduce a diagonal matrix $\boldsymbol{D} = \text{diag}(d_1, ..., d_L)$ and solve

$$\boldsymbol{W}\boldsymbol{D}^{-1}\tilde{\boldsymbol{c}} = \boldsymbol{b} \quad \text{and then} \quad \boldsymbol{c} = \boldsymbol{D}^{-1}\tilde{\boldsymbol{c}} \tag{20}$$

instead.

The first type of normalization we consider is **column normalization**, which is applied to the feature matrix as an input to SP in [Step 1]. Denote $\boldsymbol{W} = [\boldsymbol{w}_1\ \boldsymbol{w}_2\ \dots\ \boldsymbol{w}_L]$. We let $\boldsymbol{D} = \text{diag}(\|\boldsymbol{w}_1\|, ..., \|\boldsymbol{w}_L\|)$ and each column of $\boldsymbol{W}$ is normalized by its own norm:

$$\boldsymbol{W}^{\dagger} = \left[\frac{\boldsymbol{w}_1}{\|\boldsymbol{w}_1\|}, \frac{\boldsymbol{w}_2}{\|\boldsymbol{w}_2\|}, \dots, \frac{\boldsymbol{w}_L}{\|\boldsymbol{w}_L\|}\right]. \tag{21}$$

We observe that the scale of the columns in the feature matrix usually varies substantially from column to column, which negatively affects the SP step. This column normalization helps to prevent a large difference in the scale among the columns. For example, in Fig. 2 (b) shows that the magnitude of the entries in $\boldsymbol{W}$ vary from 0 to 10^{29}.

In [Step 2], we introduce our second normalization – **error normalization**, which is particularly effective for coefficient recovery. The columns in $\boldsymbol{W}$ are given by certain derivatives of a monomial of u. When we compute the feature matrix with noisy data, the noise has different effects on different features. For the feature $\frac{\partial^\alpha}{\partial x^\alpha}\left(u^\beta\right)$, the noisy data with noise ϵ in (3) give rise to the following integral feature:

$$\int_{\Omega_h} (-1)^\alpha (u+\epsilon)^\beta \frac{\partial^\alpha}{\partial x^\alpha}\left(\phi_h(x,t)\right) dxdt = \sum_{k=0}^{\beta} (-1)^\alpha \binom{\beta}{k} \epsilon^{\beta-k} \int_{\Omega_h} u^k \frac{\partial^\alpha}{\partial x^\alpha}\phi_h(x,t) dxdt.$$

The leading coefficient in the error (that is linear in ϵ) in this integral feature is obtained for $k = \beta - 1$:

$$s(h,l) = \beta \left| \int_{\Omega_h} u^{\beta-1} \frac{\partial^\alpha}{\partial x^\alpha}\left(\phi_h(x,t)\right) dxdt \right|, \quad h = 1, 2, ..., H, \ \beta \geq 1. \tag{22}$$

When $\alpha = \beta = 0$, we set $s(h,l) = 1$. This leading coefficient $s(h,l)$ depends on the row index h and the column index l. For the l^{th} column, we define

$$\langle s(h,l)\rangle_h = \frac{1}{H}\sum_{h=1}^{H} s(h,l)$$

as an average of these leading coefficients over the rows.

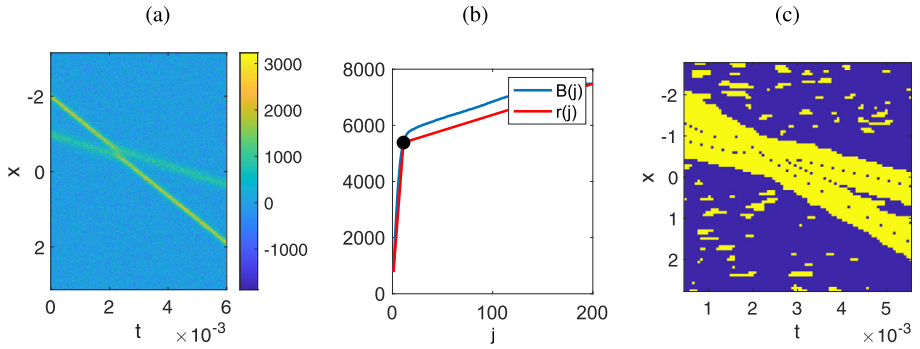


Fig. 3. Highly dynamic regions for an experiment using the KdV equation (33) with $\sigma_{\text{NSR}} = 0.5$. (a) The given noisy data $\hat{U}$ with $\sigma_{\text{NSR}} = 0.5$ in $x-t$ plane. (b) The separation point Γ (black) for $\mathbb{H}$ (24) is found, from the accumulated function $B(j)$ (blue) and the fitted piecewise linear function $r(j)$ with one junction at Γ (red). (c) The location of highly dynamic regions in the $x-t$ plane. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

By error normalization, we normalize $\boldsymbol{W}$ with the diagonal matrix $\boldsymbol{D} = \text{diag}(\langle s(h,1)\rangle_h, \ldots, \langle s(h,L)\rangle_h)$ such that $\boldsymbol{W}$ is normalized to

$$\tilde{\boldsymbol{W}} = \left[\frac{\boldsymbol{w}_1}{\langle s(h,1)\rangle_h}, \frac{\boldsymbol{w}_2}{\langle s(h,2)\rangle_h}, \ldots, \frac{\boldsymbol{w}_L}{\langle s(h,L)\rangle_h}\right] \tag{23}$$

Fig. 2 shows an example, with the given noisy data in (a) and the unnormalized feature matrix $\boldsymbol{W}$ in (b). Fig. 2 (c) shows the normalized matrix $\tilde{\boldsymbol{W}}$ after the error normalization. We use log 10 scale in Fig. 2. The difference in scale has been reduced approximately from 10^{29} in the unnormalized matrix (b) to 10^6 after normalization in (c). Our error normalization results in more uniform entry values with less variation across different columns.

In the following Subsection, we further discuss how error normalization is used to select the highly dynamic regions.

3.2. Highly dynamic regions: choice of the domain $\Omega_{h(x_i,t^n)}$

One of the benefits of using the weak form is to consider the influence of different regions on the integral computation. We take advantage of this and choose a subset of test functions indexed by $\{h|h = 2, \ldots, H\}$ to improve the coefficient recovery. We propose the following Narrow-fit procedure: (i) define the features of interest, (ii) determine the highly dynamic regions of the chosen features, and then (iii) use the subsampled matrix based on the highly dynamic regions for the coefficient recovery. This Narrow-fit procedure focuses on the regions with higher dynamical behaviors for the features of interest, so that these regions play a larger role in the coefficient recovery.

Features of interest: We focus on a small group of features which give the variation information for the differential equation, thus highlighting which rows to choose for the coefficient recovery. In this paper, we choose the features of interest to be the terms corresponding to u and first derivatives consistently for all experiments. We simply utilize the high variance region of the function value u and the first derivative, e.g., a term such as uu_x which gives a combined information, since they would likely represent a broad range of dynamical behavior observed in the data. We explored including other terms as features of interest, but they did not provide consistent improvements.

Details are as follows: In 1D, we choose the features with $(\alpha, \beta) = (1, 2)$ for the case of one variable in 1D which corresponds to $\frac{\partial}{\partial x}u^2$, this term is uu_x. For a system with two variables u, v in 1D, $(\alpha, \beta_u, \beta_v) = (1, 2, 0), (1, 0, 2)$, they are $\frac{\partial}{\partial x}u^2$ and $\frac{\partial}{\partial x}v^2$. In 2D, we choose the features with $(\alpha_x, \alpha_y, \beta) = \{(1, 0, 2), (0, 1, 2), (1, 1, 3)\}$ for a scalar equation in 2D, i.e., the features of interest are $\frac{\partial}{\partial x}u^2$, $\frac{\partial}{\partial y}u^2$ and $\frac{\partial^2}{\partial x \partial y}u^3$. For the case of 2 variables (u and v) in 2D, $(\alpha_x, \alpha_y, \beta_u, \beta_v) = \{(1, 0, 2, 0), (0, 1, 2, 0), (1, 0, 0, 2), (0, 1, 0, 2), (1, 0, 2, 1), (0, 1, 1, 2)\}$, that is there are six features of interest: $\frac{\partial}{\partial x}u^2$, $\frac{\partial}{\partial x}v^2$, $\frac{\partial}{\partial y}u^2$, $\frac{\partial}{\partial y}v^2$, $\frac{\partial}{\partial x}u^2v$, and $\frac{\partial}{\partial y}uv^2$. For each feature of interest, we utilize the leading coefficient error (22) to select highly dynamic regions. For multiple features of interest with indices $l = l_1, l_2, ..., l_{\mathcal{L}}$, we take the average over l, and let

$$\bar{s}(h) = \frac{1}{\mathcal{L}}\sum_{i=1}^{\mathcal{L}} |s(h, l_i)|,$$

with $s = \bar{s}$ for $\mathcal{L} = 1$.

Highly dynamic regions: We consider the set $\mathcal{S} = \{\bar{s}(h)|h = 1, \ldots, H\}$, which is the collection of averaged leading coefficient errors over the features of interest. We divide the set $\mathcal{S}$ into mildly and highly dynamic regions, automatically identifying the transition point Γ between these two types of dynamics as follows.

After partitioning the histogram of $\mathcal{S}$ into $N_{\mathcal{S}}$ bins $(b_1, b_2, ..., b_{N_{\mathcal{S}}})$, we consider the cumulative sum of the bins $B(j) = \sum_{i=1}^{j} b_i$. We used $N_{\mathcal{S}} = 200$ for PDEs and $N_{\mathcal{S}} = 100$ for ODEs in this paper. We fit the function $B(j)$ with a piecewise linear function $r(j)$ with one junction point, using the cost function $\sum_j (B(j) - r(j))^2 / B(j)^2$. The junction point Γ separates the highly dynamic and mildly dynamic regions. Any h with $\bar{s}(h) \geq \Gamma$ gives the highly dynamic region Ω_h which we include for the coefficient recovery. Let the collection of the row indices of highly dynamic regions be an ordered set:

$$\mathbb{H} = \{h_i \mid \bar{s}(h_i) \geq \Gamma,\ h_i < h_j \text{ for } i < j\}. \tag{24}$$

Fig. 3 illustrates how the transition point Γ is computed in (b) from the given data in (a). Fig. 3 (c) shows the locations in $x - t$ plane of the highly dynamics regions with the index set $\mathbb{H}$.

Narrow-fit: We consider a submatrix using only the ordered rows from the highly dynamic region $\mathbb{H}$, indicated by a subscript $\mathbb{H}$, for both $\boldsymbol{W}$ and $\boldsymbol{b}$:

$$\boldsymbol{W}_{\text{narrow}} := \boldsymbol{W}_{\mathbb{H}} \quad \text{and} \quad \boldsymbol{b}_{\text{narrow}} := \boldsymbol{b}_{\mathbb{H}}.$$

We also error normalize this matrix, using the rows in $\mathbb{H}$:

$$\tilde{\boldsymbol{W}}_{\text{narrow}} = \left[\frac{\boldsymbol{w}_{1\mathbb{H}}}{\langle s(h,1)\rangle_{\mathbb{H}}}, \frac{\boldsymbol{w}_{2\mathbb{H}}}{\langle s(h,2)\rangle_{\mathbb{H}}}, \ldots, \frac{\boldsymbol{w}_{L\mathbb{H}}}{\langle s(h,L)\rangle_{\mathbb{H}}} \right], \tag{25}$$

where $\boldsymbol{w}_{i\mathbb{H}}$ represents the i^{th} column with the rows indexed by $\mathbb{H}$, and $\langle s(h,l)\rangle_{\mathbb{H}}$ takes the average of $s(h,l)$ for $h \in \mathbb{H}$. This matrix is represented in Fig. 2 (c). Let $\bar{b} = \langle \boldsymbol{b}_{narrow} \rangle$ be the average of the entries of $\boldsymbol{b}_{\text{narrow}}$. After narrow-fitting, we solve:

$$\tilde{\boldsymbol{W}}_{\text{narrow}} \tilde{\boldsymbol{c}} = \tilde{\boldsymbol{b}}_{\text{narrow}} \quad \text{where} \quad \tilde{\boldsymbol{b}}_{\text{narrow}} = \boldsymbol{b}_{\text{narrow}} / \bar{b}. \tag{26}$$

We then compute the coefficient $\boldsymbol{c}$ by rescaling $\tilde{\boldsymbol{c}}$ back:

$$\boldsymbol{c} = \bar{b}\, \tilde{\boldsymbol{c}}\, \text{diag}\left\{ \frac{1}{\langle s(h,1)\rangle_{\mathbb{H}}}, \frac{1}{\langle s(h,2)\rangle_{\mathbb{H}}}, \ldots, \frac{1}{\langle s(h,L)\rangle_{\mathbb{H}}} \right\}. \tag{27}$$

3.3. Trimming the support

After the coefficient values in $\boldsymbol{c}$ are recovered, some features give very small contributions to u_t. We further trim the support by eliminating these features corresponding to small contributions.

From the solution $\tilde{\boldsymbol{c}}$ of the linear equation (26), we define a **contribution score** a_i of each feature as

$$a_i = \frac{n_i}{\max_{i \leq L} n_i} \quad \text{where} \quad n_i = ||\tilde{\boldsymbol{w}}_i||_2 |\tilde{c}_i|, \quad i = 1, 2, \ldots, L. \tag{28}$$

Here $\tilde{\boldsymbol{w}}_i$ denotes the i^{th} column of $\tilde{\boldsymbol{W}}_{\text{narrow}}$. We consider the L_2 norm of this column multiplied by the coefficient value of the i^{th} component of $\tilde{c}$. Since a_i is normalized by the maximum value of n_i, a_i gives the score of the contribution of the i^{th} feature relative to the contribution of the feature with the largest contribution.

We trim the coefficient, thus the feature, when the contribution score of that feature is below $\mathcal{T}$, i.e. $a_i < \mathcal{T}$. Typically, we set $\mathcal{T} = 0.05$ to trim the features with contributions less than 5% of u_t. Each time [Step 3] is called to trim the support set $\mathcal{A}_j^k$ to the new support set $\mathcal{A}_{j+1}^k$, and [Step 2] narrow-fit is called to find the updated coefficient value $c(k, j+1)$.

Fig. 4 shows the effect of trimming. For each sparsity level k in x-axis, the bar shows the cross validation value (30) of the recovered coefficient $c(k^*, J_{k^*})$. For a large sparsity level, thanks to the trimming step, the correct support and coefficient values are found.

3.4. Algorithms

Our WeakIdent algorithm is summarized in Algorithm 1. From the linear system in (8)

$$\boldsymbol{W}\boldsymbol{c} = \boldsymbol{b},$$

we input $\boldsymbol{b}$ and $\boldsymbol{W}$ computed through (9), with subsampling in (11). For each sparsity level $k = 1, 2, \ldots, K$,

[Step 1] First, Subspace Pursuit (SP) [38] is applied to find $\mathcal{A}_0^k = \text{supp}\{SP(\boldsymbol{W}\dagger, \tilde{\boldsymbol{b}}, s)\}$ using the column normalized matrix $\boldsymbol{W}\dagger$ in (21) and $\tilde{\boldsymbol{b}} = \boldsymbol{b}/||\boldsymbol{b}||$.

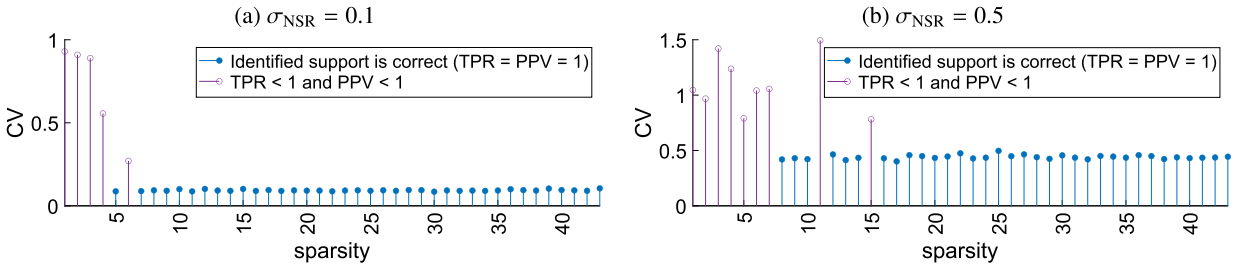


Fig. 4. Trimming is demonstrated in an experiment using the KS equation (34). For each sparsity level k in x-axis, the bar shows the cross validation (30) of the recovered coefficient $c(k^*, J_{k^*})$. Notice for most sparsity levels 5 and above the correct support is found. After SP finds k supports, the trimming step reduces the support until only the correct ones are left. Here σ_{NSR} is the noise-to-signal ratio (44), TPR is true positive rate (47) and PPV is positive prediction value (48).

[Step 2] Narrow-fit. To recover the coefficient values using the support $\mathcal{A}_j^k$, we find the row index set $\mathbb{H}$ of highly dynamic regions in (24), and solve

$$\tilde{\boldsymbol{W}}_{\mathrm{narrow}}\tilde{\boldsymbol{c}} = \tilde{\boldsymbol{b}}_{\mathrm{narrow}}$$

in (26) and get $\boldsymbol{c}(k, j)$ in (27).

[Step 3] Trimming. Update to $\mathcal{A}_{j+1}^k$, if there is any column with the contribution score in (28) below $\mathcal{T}$, i.e. $a_i < \mathcal{T}$. If trimmed, move to [Step 2] to get a new updated $\boldsymbol{c}(k, j+1)$. If no column is trimmed, move to [Step 4] and set $J_k = j$.

[Step 4] Cross Validation. With the support $\boldsymbol{c}(k, J_k)$ computed for each sparsity level $k = 1, \ldots, K$, we select the final support by finding the k^* which gives the minimum cross-validation error. For a sparsity level k, we randomly sample regions from the $N_x N_t$ regions and equally partition these regions into two sets indexed by $\mathbb{A}$ and $\mathbb{B}$ respectively. We consider the linear system in (26):

$$\tilde{\boldsymbol{W}} = \left[\frac{\boldsymbol{w}_1}{\langle s(h,1)\rangle_{\mathbb{H}}}, \frac{\boldsymbol{w}_2}{\langle s(h,2)\rangle_{\mathbb{H}}}, \ldots, \frac{\boldsymbol{w}_L}{\langle s(h,L)\rangle_{\mathbb{H}}}\right], \quad \text{and} \quad \tilde{\boldsymbol{b}} = \boldsymbol{b}/\bar{b} \tag{29}$$

utilizing the highly dynamic region error normalization for the large full matrix. Here $\mathbb{H}$ indicates ordered row index from the set $\mathbb{H}$, and $\langle s(h,l)\rangle_{\mathbb{H}}$ taking the average of $s(h,l)$ for $h \in \mathbb{H}$. We solve least square problems $\tilde{\boldsymbol{W}}_{\mathbb{A}}\tilde{\boldsymbol{c}}_{\mathbb{A}} = \tilde{\boldsymbol{b}}_{\mathbb{A}}$ and $\tilde{\boldsymbol{W}}_{\mathbb{B}}\tilde{\boldsymbol{c}}_{\mathbb{B}} = \tilde{\boldsymbol{b}}_{\mathbb{B}}$, where $\tilde{\boldsymbol{W}}_{\mathbb{A}}$ and $\mathbb{B}$ contain the rows of $\tilde{\boldsymbol{W}}$ indexed by $\mathbb{A}$ and $\mathbb{B}$ respectively. Then, we compute the **cross validation** (CV) error

$$\mathrm{CV}(k) = \lambda||\tilde{\boldsymbol{W}}_{\mathbb{A}}\tilde{\boldsymbol{c}}_{\mathbb{B}} - \tilde{\boldsymbol{b}}_{\mathbb{A}}||_2 + (1-\lambda)||\tilde{\boldsymbol{W}}_{\mathbb{B}}\tilde{\boldsymbol{c}}_{\mathbb{A}} - \tilde{\boldsymbol{b}}_{\mathbb{B}}||_2, \tag{30}$$

where we set $\lambda = 1/100$. In practice, for each k, we generate 30 different random partitions of $\mathbb{H}$ to $\mathbb{A}$ and $\mathbb{B}$, then select the minimum:

$$\boldsymbol{c}(k^*, J_{k^*}) = \arg\min_k \{\mathrm{CV}(k) | k = 1, 2, ..., L\}. \tag{31}$$

Here $K \le L$, since L is the total number of features in the dictionary. In practice, a small K is needed. Fig. 4 illustrates that for (small) values of K around $K = 10$ and below, the correct coefficients are found, thanks to the trimming step.

4. WeakIDENT results and comparisons

In this section, we provide detailed experimental results. We summarize a list of PDEs and ODE systems in Table 1 and Table 2. For the systems of ODEs, we consider features with polynomial order between 3 and 5, with $L \le 21$ for all the cases. For the systems of PDEs, we consider features with both polynomial order and derivative order between 4 and 6, which gives a dictionary of size $L \le 65$ for the 1 spatial dimension and $L \le 190$ for 2 spatial dimensions. Simulation and feature details are presented in Table 1 and 2 for each experiment.

For PDEs, N_x and N_t are chosen such that $N_x N_t \in (1,000, 3,000)$ to reduce the computational cost. In particular, we set

$$N_x = \lceil \frac{\mathbb{N}_x - 2m_x - 1}{\lfloor \mathbb{N}_x/\mathbf{N} \rfloor} + 1 \rceil \quad \text{and} \quad N_t = \lceil \frac{\mathbb{N}_t - 2m_t - 1}{\lfloor \mathbb{N}_t/\mathbf{N} \rfloor} + 1 \rceil, \tag{38}$$

Algorithm 1: WeakIdent Algorithm.

Input: $\boldsymbol{W} \in \mathbb{R}^{H\times L}$, $\boldsymbol{b} \in \mathbb{R}^{H}$, from (8) uniformly subsampled as (11); Parameter $\mathcal{T} = 0.05$
for k = 1,2,...,K **do**
  [Step 1] $\mathcal{A}_0^k = \text{supp}\{\text{SP}(\boldsymbol{W}\dagger, \tilde{\boldsymbol{b}}, s)\}$ use SP [38] and set $j = 0$;
  [Step 2] Find $c(k, j)$ by narrow-fit (26);
    while *there exists* $a_i < \mathcal{T}$ *as in (28)* **do**
      [Step 3] Trim as in Subsection 3.3 and set $j = j + 1$;
      [Step 2] Find $c(k, j + 1)$ by Narrow-fit (26);
    end
  end
Among $k = 1, \ldots, K$, find $\boldsymbol{c}(k^*, J_{k^*})$ by Cross Validation in (31).
Output: $\boldsymbol{c} = \boldsymbol{c}(k^*, J_{k^*}) \in \mathbb{R}^L$ such that $\boldsymbol{W}\boldsymbol{c} \approx \boldsymbol{b}$.

Table 1
A list of PDEs considered in this paper. Here L is the total number of features, $\bar{\alpha}$ is the highest order of partial derivative, $\bar{\beta}$ is the highest degree used in f_l in (4), $[X_1, X_2]$ is the range of the spatial domain, T is the final time for simulation. Δx and Δt are the spatial and temporal increment of the given data. The set up of (33), (34), (35), (36), and (37) are identical to [23].

Equation		Parameters
Transport equation $\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x} + 0.05\frac{\partial^2 u}{\partial x^2}$	(32)	$L = 43$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$, $[X_1, X_2] = [0, 1]$, $\Delta x = 0.039$, $T = 0.3$, $\Delta t = 0.001$ $u(x, 0) = \sin(4\pi/(1-T)x)^3 \cos(\pi/(1-T)x)$ for $x < 1 - T$, and 0 otherwise
Korteweg-de Vires (KdV) $\frac{\partial u}{\partial t} = -0.5u\frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3}$	(33)	$L = 43$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$, $[X_1, X_2] = [-\pi, \pi]$, $\Delta x = 0.0157$, $T = 0.006$, $\Delta t = 10^{-5}$ $u(x, 0) = 3.0 \times 25^2 * \text{sech}(0.5 \times (25 \times (x + 2.0)))^2$ $+3.0 \times 16^2 * \text{sech}(0.5 \times (16 * (x + 1.0)))^2$
Kuramoto-Sivashinsky (KS) $\frac{\partial u}{\partial t} = -u - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}$	(34)	$L = 43$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$, $[X_1, X_2] = [0, 100.53]$, $\Delta x = 0.3927$, $T = 150$, $\Delta t = 0.5$ $u(x, 0) = \cos(x/16)(1 + \sin(x/16))$.
Nonlinear Schrodinger (NLS) (1D) $\begin{cases} \frac{\partial u}{\partial t} = 0.5\frac{\partial^2 v}{\partial x^2} + u^2 v + v^3 \\ \frac{\partial v}{\partial t} = -0.5\frac{\partial^2 u}{\partial x^2} - uv^2 - u^3 \end{cases}$	(35)	$L = 190$, $\bar{\alpha} = 6$, $\bar{\beta} = 6$ $[X_1, X_2] = [-5, 5]$, $\Delta x = 0.0391$ $T = 3.1416$, $\Delta t = 0.0126$
Anisotropic Porous Medium (PM) (2D) $\frac{\partial u}{\partial t} = +0.3\frac{\partial^2 u^2}{\partial^2 y} - 0.8\frac{\partial^2 u^2}{\partial x \partial y} + \frac{\partial^2 u^2}{\partial^2 x}$	(36)	$L = 65$, $\bar{\alpha} = 4$, $\bar{\beta} = 4$ $[X_1, X_2] = [-5, 5]$, $\Delta x = 0.0503$ $T = 5$, $\Delta t = 0.0503$
Reaction-Diffusion (2d) $\begin{cases} \frac{\partial u}{\partial t} = 0.1\frac{\partial^2 u}{\partial^2 x} + 0.1\frac{\partial^2 u}{\partial^2 t} + u + v^3 - uv^2 + u^2 v - u^3 \\ \frac{\partial v}{\partial t} = 0.1\frac{\partial^2 v}{\partial^2 y} + 0.1\frac{\partial v^2}{\partial^2 x} + v - v^3 - uv^2 - u^2 v - u^3 \end{cases}$	(37)	$L = 155$, $\bar{\alpha} = 4$, $\bar{\beta} = 5$, $[X_1, X_2] = [-10, 10]$ $\Delta x = 0.0781$, $T = 9.9219$, $\Delta t = 0.0781$ $u(x, y, 0) = \tanh(\sqrt{x^2 + y^2}\cos(\theta(x + iy) - \pi\sqrt{x^2 + y^2})$, $v(x, y, 0) = \tanh(\sqrt{x^2 + y^2}\sin(\theta(x + iy) - \pi\sqrt{x^2 + y^2})$

with $\mathbf{N} = 50$ as a default choice. Here $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ denotes the ceiling and floor operator. In Table 1, (38) is used for the transport question (32), the KS equation (34) and the nonlinear Schrodinger equation (35). For certain cases such as the KdV equation (33) where $|\mathbb{H}|$ is very small, we increase N_x and N_t, e.g., using $\mathbf{N} = 70$, such that $|\mathbb{H}| > 800$. For the spatially 2 dimensional cases, we use $\mathbf{N} = (25, 25)$ for the anisotropic porous medium equation (PM) (36), and $\mathbf{N} = (19, 16)$ for the 2D reaction-diffusion equation (37) to reduce the time of computation. For the ODEs listed in Table 2, we choose $N_t \approx 1000$ by default with $\mathbf{N} = 1000$. Since we use different subsampling, we present additional comparisons in Section 4.5 to demonstrate that the effect of subsampling on the result is minimal.

The experiments are performed on both clean data and noisy data with various Noise-to-Signal Ratio, σ_{NSR} defined as follows:

Table 2
A list of ODEs considered in this paper. This table includes the initial condition, the temporal increment Δt, the total simulation time T, the total number of features L and the highest degree of polynomials $\bar{\beta}$ in (4) for each equation. The Solution is simulated with RK45 with tolerance 10^{-10}.

Name	Equation		Parameters
2D Linear System	$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.15 & 2.5 \\ -2.5 & -0.15 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$	(39)	$(x_0, y_0) = (2, 50)$, $\Delta t = 0.01, T = 10$ $L = 21, \bar{\beta} = 5$
2D Nonlinear (Van der Pol)	$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 4 & -1 & -4 \end{bmatrix}\begin{bmatrix} x \\ y \\ x^2y \end{bmatrix}$	(40)	$(x_0, y_0) = (0, 1)$ $\Delta t = 0.001, T = 15$ $L = 21, \bar{\beta} = 5$
2D Nonlinear (Duffing)	$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -0.2 & -0.05 & -1 \end{bmatrix}\begin{bmatrix} x \\ y \\ x^3 \end{bmatrix}$	(41)	$(x_0, y_0) = (0, 2)$ $\Delta t = 0.01, T = 10$ $L = 21, \bar{\beta} = 5$
2D Nonlinear (Lotka-Volterra)	$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.67 & 0 & -1.33 \\ 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ xy \end{bmatrix}$	(42)	$(x_0, y_0) = (10, 10)$ $\Delta t = 0.05, T = 50$ $L = 21, \bar{\beta} = 5$
3D Nonlinear (Lorenz)	$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -10.2 & 10.2 & 0 & 0 & 0 \\ 29 & -1 & 0 & 0 & -1 \\ 0 & 0 & -2 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ xy \\ xz \end{bmatrix}$	(43)	$(x_0, y_0, z_0) = (-8, 7, 10)$ $\Delta t = 0.001, T = 15$ $L = 20, \bar{\beta} = 3$

$$\sigma_{\text{NSR}} = \frac{\epsilon_i^n}{\frac{1}{\mathbb{N}_t\mathbb{N}_x}\sum_{i,n}|U_i^n - (\max_{i,n} U_i^n + \min_{i,n} U_i)/2|^2} \tag{44}$$

for $i = 1, ..., \mathbb{N}_x$, $n = 1, ..., \mathbb{N}_t$. Note that our definition of NSR reflects the local variation of the given data. This is different from the absolute variation (absolute root mean squared of U_i^n) σ_{NR} used in [23], and this σ_{NSR} value tends to be smaller than the σ_{NR} value. We also mention the σ_{NR} value in the following experiments when it is relevant. We use Gaussian noise, such that $\epsilon_i^n \sim \mathcal{N}(0, \sigma_{\text{NSR}})$ for ϵ_i^n, and $\hat{U}_i^n$ in (3). For the case of multiple variables, we compute (44) for each variable.

Error measures: To quantify the quality of the recovery, we utilize different error measurements listed in Table 3. The relative coefficient errors E_2 in (45) and E_∞ in (46) measure the accuracy of the recovered coefficients $\boldsymbol{c}$ against the true coefficients $\boldsymbol{c}^*$ in terms of the l_2 and the infinity norm, respectively. We introduce two new measures to quantify the accuracy of the support recovery. The True Positive Rate (TPR)[4] (47) measures the fraction of features that are found out of all features in the true equation, and is defined as the ratio of the cardinality of the correctly identified support over the cardinality of the true support. The TPR is 1 if all the true features are found. The Positive Predictive Value (PPV) (48) indicates the presence of false positives: it is the ratio of the cardinality of the correctly identified support over the total cardinality of the identified support. The PPV is 1 if the recovered support is also in the true support. The residual error E_{res} in (49), which is also used in [17], measures the relative difference between the learned differential equation and the given data. To show the effectiveness of WeakIdent in the recovery of the dynamics, we define the dynamical error E_{dyn} in (50) to measure the difference between the true dynamics and the expected dynamics simulated from the recovered equation. In (50), we use $U^n_{\mathbf{i},\text{forward}}$ and $U^n_{\mathbf{i},\text{clean}}$ to denote the simulated data and the true data without noise. We simulate ODEs using RK45 with the relative error tolerance to be 10^{-10}. This is measured for ODEs only, due to restricted stability conditions for PDEs. If the identified equation blows up before the final time T is reached, we compare $U^n_{\mathbf{i},\text{forward}}$ and $U^n_{\mathbf{i},\text{clean}}$ just before the blow-up.

4.1. WeakIdent results and comparisons for PDEs

We present the WeakIdent and comparisons in this subsection for PDEs, and in subsection 4.2 for ODEs. We compare with existing methods, such as the IDENT in [16], the Robust Ident, with Subspace pursuit Cross validation (SC) and Subspace pursuit Time evolution (ST) in [17], SINDy [9], and methods using the weak form such as RGG [25], Weak SINDy for first order dynamical systems (WODE) [24], and Weak SINDy for PDEs (WPDE) [23].

[4] The definition of TPR in (47) is different from that used in [23].

Table 3
Error measurements used for comparisons.

Relative coefficient Error l_2	$E_2 = \|\|\boldsymbol{c}^* - \boldsymbol{c}\|\|_2 / \|\|\boldsymbol{c}^*\|\|_2$	(45)
Relative coefficient Error l_∞	$E_\infty = \max_l \{\|\boldsymbol{c}^*(l) - \boldsymbol{c}(l)\| / \|\boldsymbol{c}^*(l)\| : \boldsymbol{c}^*(l) \neq 0\}$	(46)
True Positive Rate	$\text{TPR} = \|\{l : \boldsymbol{c}^*(l) \neq 0,\ \boldsymbol{c}(l) \neq 0\}\| / \|\{l : \boldsymbol{c}^*(l) \neq 0\}\|$	(47)
Positive Predictive Value	$\text{PPV} = \|\{l : \boldsymbol{c}^*(l) \neq 0,\ \boldsymbol{c}(l) \neq 0\}\| / \|\{l : \boldsymbol{c}(l) \neq 0\}\|$	(48)
Residual Error	$E_{\text{res}} = \|\|\boldsymbol{W}\boldsymbol{c} - \boldsymbol{b}\|\|_2 / \|\|\boldsymbol{b}\|\|_2$	(49)
Dynamic Error	$E_{\text{dyn}} = \sum_{1 \le i \le \mathbb{N}_x, 1 \le n \le \mathbb{N}_t} (\|U^n_{\mathbf{i},\text{forward}} - U^n_{\mathbf{i},clean}\|^2) / (\mathbb{N}_x \mathbb{N}_t)$	(50)

For fair comparisons, when available, we used the same underlying equations provided by SINDy [9], WODE [24], WPDE [23] or RGG [25] provided in their respective Githubs.[5] WeakIdent and WPDE use the same dictionary of features as well as the same parameters for the weak form (a system with the same number of variables and dimensions in the spatial domain) to each other. RGG [25] uses a subset of features (e.g. 8-14 features), which is different from other methods which use the full feature matrix ($L = 21$ to 190 features). For each experiment in the comparison, we specify which features are used for RGG. In many of the PDE experiments in this section, we show comparisons only between our proposed WeakIdent and WPDE [23], since these two methods give the best results compared to others, based on the error measures in Table 3.

4.1.1. Transport equation

The first set of results in Fig. 5 shows results for the transport equation (32) with clean and noisy data. (a), (b) and (c) compare the recovery results with clean data, and (d), (e) and (f) compare the results with highly corrupted data where $\sigma_{\text{NSR}} = 100\%$. For the case of clean data, RGG [25], WPDE [23] and the proposed WeakIdent find the correct support u_x, u_{xx}, while the latter two methods have higher accuracy. In the noisy case of $\sigma_{\text{NSR}} = 100\%$, only WeakIdent is able to identify the correct support with the E_2 value as low as 0.008.

In Fig. 6, we provide statistical comparisons between our proposed WeakIdent and WPDE [23] applied to the transport equation (32) for different levels of σ_{NSR}. We show box-plots for the distribution of the identification errors E_2, E_∞, TPR and PPV over 50 experiments for each level of $\sigma_{\text{NSR}} \in \{0.01, 0.1, 0.2, ..., 0.9\}$. The WeakIdent results are robust even for large noise levels: Panels (a3) and (a4) show that in the majority ($> 75\%$) of the cases, a correct support is found by WeakIdent with low E_2 error in Panel (a1).

4.1.2. Anisotropic Porous Medium (PM) equation

In Fig. 7, we compare the recovery results for the 2D anisotropic porous medium equation (PM) (36), which includes a feature with the cross-dimensional derivative u_{xy}. Fig. 7 (a) shows $\hat{U}(\boldsymbol{x}, 0)$ and (b) shows $\hat{U}(\boldsymbol{x}, T)$, where the given noisy data has noise-to-signal ratio $\sigma_{\text{NSR}} = 0.08$. This noise level is equivalent to $\sigma_{\text{NR}} = 0.4139$ as defined in WPDE [23]. We show different recovered equations with the identification error E_2 in (c). WeakIdent is able to identify the correct support with the coefficient error $E_2 = 0.0056$, demonstrating WeakIdent's capability to identify features across multiple dimensions on 2D spatial domain.

4.1.3. Reaction-diffusion equation

In Fig. 8, we compare the recovery results for the 2D reaction-diffusion equation (37). These systems can generate a variety of patterns such as dots, strips, waves and hexagons. The Laplacian (diffusion) features $\Delta u, \Delta v$ in this equation may be difficult to identify in general, particularly in the case where the diffusion coefficients are small compared to those of other features, and accumulated noise can be emphasized. We use the spiral pattern data set from [23]. Fig. 8 (a) shows $\hat{U}(\boldsymbol{x}, 0)$ and (b) shows $\hat{U}(\boldsymbol{x}, T)$, where the given noisy data has $\sigma_{\text{NSR}} = 0.08$ (equivalent to $\sigma_{\text{NR}} = 0.08$ defined in [23]). We show different recovered equations with the E_2 identification error in Fig. 8 (c). WeakIdent finds the correct terms with a small coefficient error.

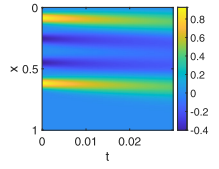
In Fig. 9, we present the statistical results of WeakIdent over 50 experiments for the 2D reaction-diffusion equation (37).

4.1.4. PDEs and systems of PDEs with higher order features

In Fig. 10, we show the average errors of WeakIdent and WPDE over 50 experiments on the PDEs and systems of PDEs in Table 1 with different noise levels. Each column gives the E_2 error, TPR and PPV respectively. In each row, we present the

[5] SINDy and WODE at https://github.com/dm973/WSINDy_ODE, WPDE at https://github.com/dm973/WSINDy_PDE, and RGG at https://github.com/pakreinbold/PDE_Discovery_Weak_Formulation.

(a) U



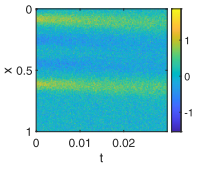
(b) $\sigma_{\text{NSR}} = 0$

	WeakIdent	WPDE [23]	RGG[25]	IDENT[16]	SC[17]	ST[17]
E_2	**0.001**	**0.001**	**0.001**	-	2.26	2.24
E_∞	**0.001**	**0.001**	**0.001**	-	-	3.08
E_{res}	**0.001**	**0.001**	**0.001**	0.98	0.03	0.03
TPR	**1.0**	**1.0**	**1.00**	-	0.00	0.50
PPV	**1.0**	**1.0**	**1.00**	0.00	0.00	0.20

(c) $\sigma_{\text{NSR}} = 0$

True equation	$u_t = -1.00000u_x + 0.05000u_{xx}$
WeakIdent	$\mathbf{u_t = -1.00145u_x + 0.04999u_{xx}}$
WPDE [23]	$\mathbf{u_t = -1.00144u_x + 0.05000u_{xx}}$
RGG [25]	$\mathbf{u_t = -1.00119u_x + 0.04999u_{xx}}$
IDENT[16]	$u_t = -0.0006 + 0.0036u + 0.0244u^2 - 0.9992u_x + 0.0004(u_x)^2 + ...$
SC[17]	$u_t = +1.74039u^2 - 1.03236u_x + 0.05168u_{xx} + 0.00298uu_{xx}$
ST[17]	$u_t = +1.73061u^2 - 1.01121u_x - 0.10390uu_x + 0.05167u_{xx} + 0.00298uu_{xx}$

(d) $\hat{U}$



(e) $\sigma_{\text{NSR}} = 1$

	WeakIdent	WPDE [23]	RGG [25]	IDENT [16]	SC [17]	ST[17]
E_2	**0.008**	0.184	135.33	-	17.43	20.32
E_∞	**0.008**	1.129	0.13	-	-	18.23
E_{res}	**0.811**	0.830	0.95	0.82	0.91	0.89
TPR	**1.0**	**1.0**	0.50	0.00	0.00	0.50
PPV	**1.0**	0.5	0.25	0.00	0.00	0.20

(f) $\sigma_{\text{NSR}} = 1$

True equation	$u_t = -1.00000u_x + 0.05000u_{xx}$
WeakIdent	$\mathbf{u_t = -1.00792u_x + 0.05029u_{xx}}$
WPDE [23]	$u_t = -1.02983u_x + 0.10647u_{xx} - 0.15741(u^3)_{xx} + 0.07197(u^6)_{xx}$
RGG [25]	$u_t = -0.00003u_{xxxx} - 0.87531u_x + 44.70146u^2 - 127.91622u^3$
IDENT[16]	$u_t = -0.2710 + 6.1120u - 3.4346u^2 - 0.0000u_x + 0.0000(u_x)^2 + ...$
SC[17]	$u_t = -3.35704u - 17.09628u^2 - 0.26659u_x$
ST[17]	$u_t = +0.60609 - 4.44906u - 19.79311u^2 - 0.17997u_x - 0.86156uu_x$

For RGG [25], we use 8 default features $\{uu_x, u_{xx}, u_{xxxx}, u, u_x, u_{xxx}, u^2, u^3\}$ and the parameters $p_x = 4, p_t = 3, N_d = 100, D = (40, 20)$ are used. For IDENT [16], we use $\lambda = 200$ for the sparse regression algorithm, and the dictionary is set to be $\{1, u, u^2, u_x, u_x^2, uu_x, u_{xx}, u_{xx}^2, uu_{xx}, u_xu_{xx}\}$. SC and ST [17] use the same dictionary as IDENT. For SC, we use $\alpha = 100$ and for ST, we use $s = 20$ and $n = 5$. These parameters are from the original papers.

Fig. 5. Transport equation with diffusion (32): clean data case in (a), (b) and (c), and noisy data with $\sigma_{\text{NSR}} = 100\%$ in (d), (e) and (f). WeakIdent is compared with WPDE [23], RGG [25], IDENT [16], SC [17], and ST [17]. The error measures are in Table (b) and (e) and the recovered equations are in (c) and (f).

results from the transport equation (32), KdV equation (33), the KS (34), the nonlinear Schrodinger (35), the anisotropic PM equation (36), and the 2D reaction-diffusion equation (37). In the first column, we present the ratio $\tilde{\sigma} = \sigma_{\text{NSR}}/\sigma_{\text{NR}}$ where σ_{NR} denotes the noise ratio in WPDE [23]. (The upper bounds of the noise ratio σ_{NR} [23] are 1.07, 0.78, 0.9, 0.81, 0.78, 0.1 for each equation.) Here the KdV (33) and KS equations (34) include higher order derivative features u_{xxx} and u_{xxxx}. These features are in general difficult to recover, especially from highly corrupted noisy data. Each plot gives comparisons between WeakIdent (Red) and WPDE (blue), with σ_{NSR} on the x-axis. The y-axis is the E_2 error, TPR, or PPV averaged over 50 experiments for a given σ_{NSR}. According to the E_2 error shown in the first column, WeakIdent has smaller E_2 errors than other methods, showing that WeakIdent is more accurate in the coefficient recovery. According to the TPR and PPV in the second and third column, WeakIdent is more accurate in support recovery since the TPR and PPV values of WeakIdent are closer to 1.

4.2. WeakIdent results and comparisons for ODEs

Since ODE systems do not include spatial derivatives, they have lower computational cost in feature computation. We consider polynomial terms with the highest order being 5. Table 2 presents details of the parameters used for simulation. In Fig. 11, we show the identified dynamics and various identification errors obtained from WeakIdent on the 5 ODE systems listed in Table 2. The noise-to-signal ratio is $\sigma_{\text{NSR}} = 0.2$ for the linear system (39), the Van der Pol nonlinear system (40) and $\sigma_{\text{NSR}} = 0.1$ for the rest of the systems. Fig. 11 (a)-(e) show the phase portraits of the given noisy data for the different

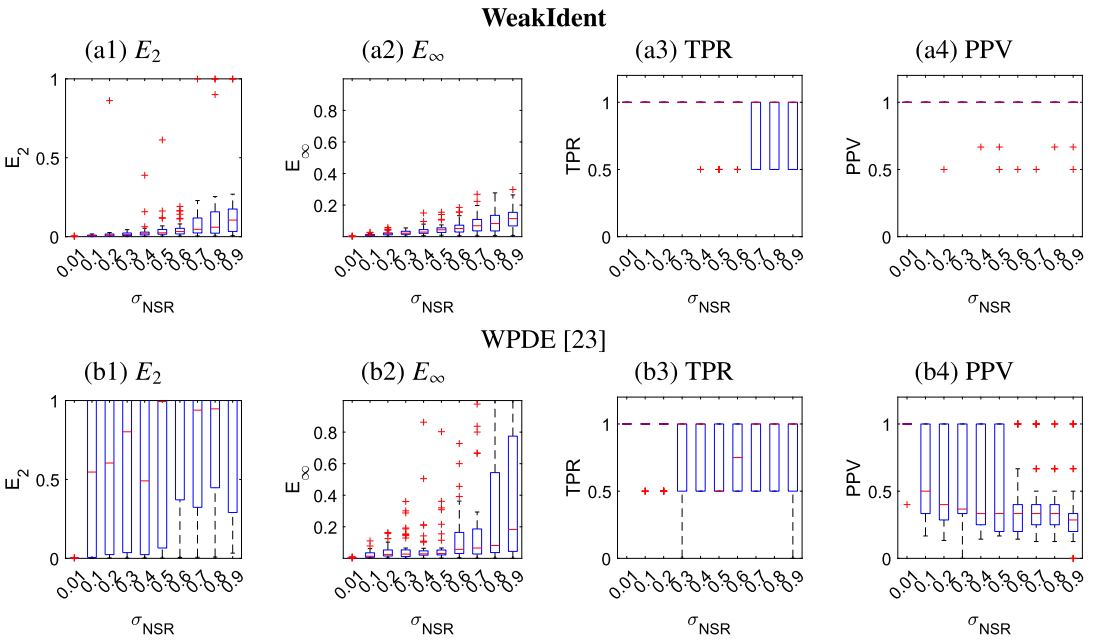
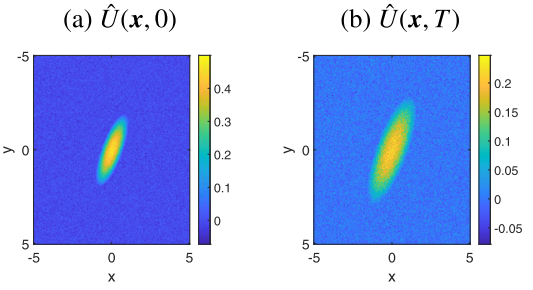


Fig. 6. Transport equation (32), statistical comparison between WeakIdent (the top row) and WPDE [23] (the second row). The errors E_2, E_∞, TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.1, 0.2, ..., 0.9\}$ using box-plots. The E_2 and E_∞ errors by WeakIent are lower than the errors of WPDE, with less variations. The TPR and PPV by WeakIdent are closer to 1 with less variations as well.



(c) $\sigma_{\text{NSR}} = 0.08$

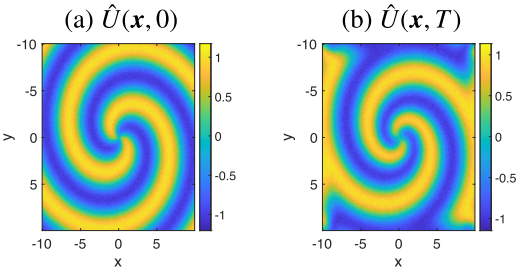
True equation	$u_t = +0.30000(u^2)_{yy} - 0.80000(u^2)_{xy} + 1.00000(u^2)_{xx}$	
WeakIdent	$\mathbf{u_t = +0.29912(u^2)_{yy} - 0.79416(u^2)_{xy} + 0.99568(u^2)_{xx}}$	$E_2 = \mathbf{0.0056}$
WPDE	$\mathbf{u_t = +0.29928(u^2)_{yy} - 0.79362(u^2)_{xy} + 0.99512(u^2)_{xx}}$	$E_2 = \mathbf{0.0061}$
RGG[25]	$u_t = +0.00028 + 0.23457u - 6.28949u^2 + 21.31151u^3 - 0.36341(u^2)_x + 0.48573(u^2)_y - 0.12914(u^2)_{xx} + 1.34146(u^2)_{yy} - 1.18102(u^2)_{xy} + 0.03796u_x - 0.03006u_y + 0.06174u_{xx} - 0.04775u_{yy} + 0.06238u_{xy}$	$E_2 = 8.9376$

For RGG [25], we use a dictionary of 14 features $\{1, u, u^2, u^3, (u^2)_x, (u^2)_y, (u^2)_{xx}, (u^2)_{yy}, (u^2)_{xy}, u_x, u_y, u_{xx}, u_{yy}, u_{xy}\}$ adding the true features, and the parameters $p_x = 2, p_t = 1, N_d = 100$, and $D = (20, 10)$.

Fig. 7. Anisotropic Porous Medium (PM) equation (36) on a 2-D spatial domain with cross derivative feature. We set $\sigma_{\text{NSR}} = 0.08$, which is equivalent to $\sigma_{\text{NR}} = 0.4139$ in WPDE [23]. (a) Given noisy data $\hat{U}(\boldsymbol{x}, 0)$ and (b) $\hat{U}(\boldsymbol{x}, T)$. (c) Identified equations with the E_2 error.

ODEs (red) superimposed on the simulated true data (black). Fig. 11 (f)-(j) show the WeakIdent results (green) compared to the true solution (black). WeakIdent is able to find the correct support in the majority of the cases with $E_2 \leq 0.088$.

Fig. 12 compares the recovery results for the Lotka-Volterra (LV) system (42) across different methods, showing results for the given data sets with various noise levels. The methods we compare include WODE [24], SINDy [9], Robust IDENT SC [17] and ST [17]. Each column is associated with an error type and each row gives results from one method. WeakIdent is able to capture the correct support with a low coefficient error in the last rows. WODE, SINDy, SC and ST has larger coefficient errors with incorrect support in many cases. A similar statistical comparison between these methods on the Lorenz system (43) is shown in Figure 19 in the Supplementary Material 5.2.2. We refer to Table 5 and Table 6 for the



(c) $\sigma_{\text{NSR}} = 0.08$

True equation	$u_t = +v^3 + u + 0.1u_{yy} + 0.1u_{xx} - uv^2 + u^2v - u^3$ $v_t = v + 0.1v_{yy} + 0.1v_{xx} - v^3 - uv^2 - u^2v - u^3$	
WeakIdent	$u_t = +0.99213v^3 + 0.98572u + 0.09660u_{yy} + 0.09695u_{xx} - 0.93229uv^2 + 0.97678u^2v - 0.99018u^3$ $v_t = +0.97792v + 0.09662v_{yy} + 0.09636v_{xx} - 0.97161v^3 - 0.96468uv^2 - 0.95572u^2v - 0.99605u^3$	E_2 =**0.0316**
WPDE	$u_t = +1.34525v^3$ $v_t = -1.34499u^3$	E_2 =0.9081
RGG[25]	$u_t = +0.10204\nabla u + 1.02296u - 1.01966u^3 + 1.01341v^3 + 1.03003u^2v - 1.01767uv^2$ $v_t = +0.09244\nabla v - 0.07400u - 0.93640u^3 + 0.95099v - 0.95370v^3 - 0.95450u^2v - 0.93750uv^2$	E_2 =0.0793

For RGG [25], the provided default features for reaction-diffusion type equation in [25] is used: for u, the dictionary is $\{\nabla u, u, u^2, u^3, v, v^2, v^3, uv, u^2v, uv^2\}$ and for v, the dictionary is $\{\nabla v, u, u^2, u^3, v, v^2, v^3, uv, u^2v, uv^2\}$, and parameters $p_x = 2, p_t = 1, N_d = 100, D = (20, 10)$.

Fig. 8. Reaction-diffusion equation (37) on a 2D spatial domain with $\sigma_{\text{NSR}} = 0.08$ (equivalent to $\sigma_{\text{NR}} = 0.08$ defined in [23]). (a) Given noisy data $\hat{U}(\boldsymbol{x}, 0)$ and (b) $\hat{U}(\boldsymbol{x}, T)$. (c) The identified equations and the E_2 errors. WeakIdent finds the correct terms with a small coefficient error.

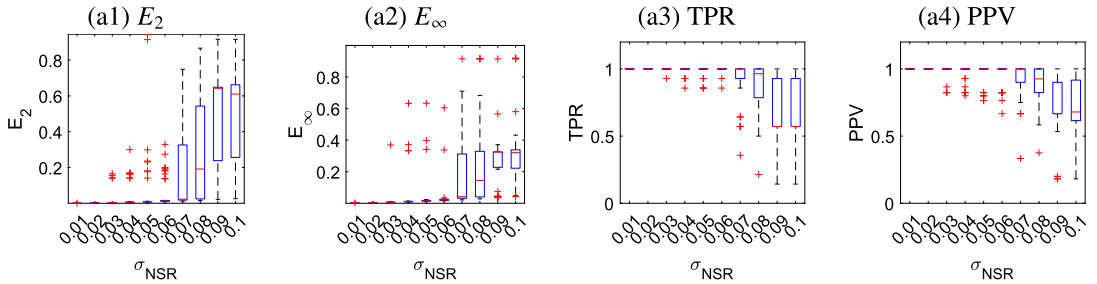


Fig. 9. The Identification results from WeakIdent for the reaction diffusion equation (37): The E_2, E_∞ errors, TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.02, ..., 0.1\}$ using box-plots.

recovery results of the Lotka-Volterra system (42) and the Lorenz system (43) from two noisy data sets with $\sigma_{\text{SNR}} = 0.1$. We also provide a comprehensive comparison on all ODE systems listed in Table 2 in Supplementary Material 5.2.2 (See Figure 18 for the details).

4.3. Influence of the initial condition in WeakIdent

Fig. 13 shows comparisons of WeakIdent and WPDE for the KS equation (34) on noisy data with $\sigma_{\text{NSR}} = 0.6$, using 5 different initial conditions: (1) $u(x, 0) = \cos(x/16). * (1 + \sin(x/16))$, (2) $u = \cos(x/4). * (1 + \sin(x/5))$, (3) $u = \cos(x/10). * (1 + \cos(x/5))$, (4) $u = \sin(x/4). * (1 + \cos(x/5))$, (5) $u = \sin(x.^2/4)$. The top row illustrates the given clean data from the different initial conditions yielding different pattern evolution. In each box plot, the x-axis gives the indices of the initial condition (1)-(5). WeakIdent recovery is robust across these different patterns in recovering this system with higher order features.

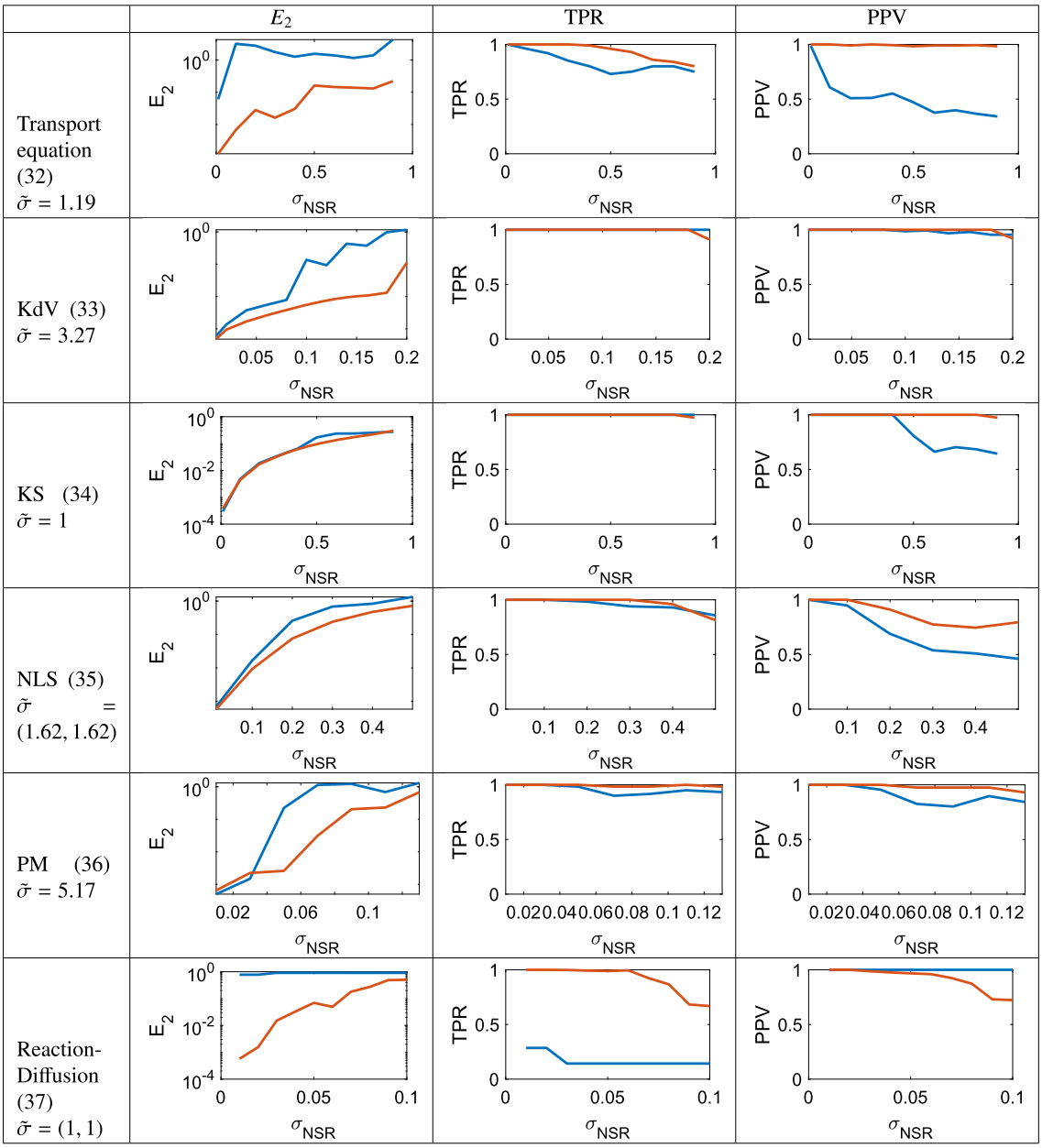
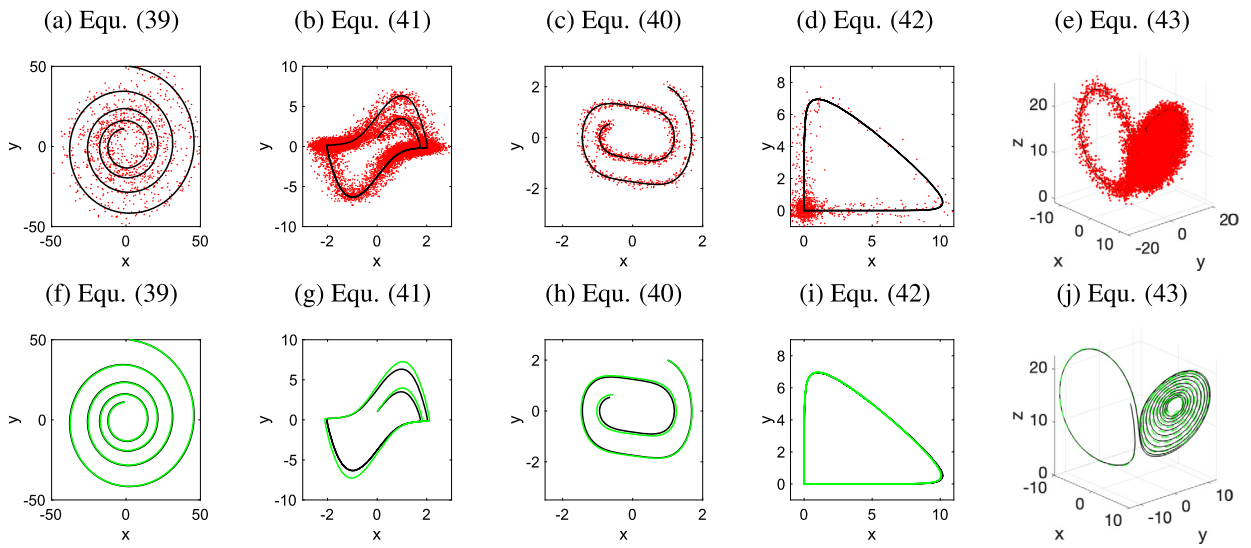


Fig. 10. The identified PDEs in Table 1 for different noise levels. We compare WeakIdent (Red) and WPDE (Blue). The x-axis is σ_{NSR}, while the y-axis is the average E_2 error, TPR and PPV over 50 experiments. The relative noise ratio $\tilde{\sigma} = \sigma_{NSR}/\sigma_{NR}$ compares our noise level σ_{NSR} vs. σ_{NR} in [23]. We present results for the transport equation (32), the KdV equation (33), the KS equation (34), the NLS equation (35), the PM equation (36), and the reaction-diffusion (2D) equation (37). The noise-to-signal ratio σ_{NSR} ranges in $\{0, 0.1, 0.2, ..., 0.9\}$, $\{0.01, 0.02, 0.04, ..., 0.24\}$, $\{0, 0.1, 0.2, ..., 0.9\}$, $\{0.01, 0.1, 0.2, ..., 0.5\}$, $\{0.01, 0.03, 0.05, ..., 0.15\}$, and $\{0.01, 0.02, ..., 0.1\}$ for each equation respectively.

4.4. The choice of the trimming parameter $\mathcal{T}$

In Fig. 14, we present the coefficient E_2 error (y-axis) against different values of the trimming parameter $\mathcal{T}$ (x-axis) for different noise-to-signal ratios (different color curves) for (a) the KdV equation (33) and (b) the KS equation (34). In general, we use $\mathcal{T} = 0.05$ as a default for all equations in Table 1 and Table 2, except for the KS equation (34) and the PM equation (36) for which we use $\mathcal{T} = 0.2$. Our experiments use the same distribution of seeds for the noise with different variances. Different color curves represent the different values of noise-to-signal ratio $\sigma_{NSR} \in \{0, 0.1, ..., 1\}$. For example, when there is no noise, $\sigma_{NSR} = 0$ (the lowest blue curve), it gives the lowest recovery error (compared to other colored curves) over the widest range of allowable $\mathcal{T}$. There is a wide range of $\mathcal{T}$ that yields the same recovery. We use $\mathcal{T} = 0.2$ for the KS and PM equations, by choosing a value of $\mathcal{T}$ from a large plateau. This makes the algorithm more robust. In general, since the



	σ_{NSR}	E_2	E_∞	E_{res}	E_{dyn}	TPR	PPV
(f) 2D Linear System (39)	0.2	0.006	0.063	0.061	0.484	1	1
(g) Van der Pol (41)	0.2	0.055	0.080	0.042	0.120	1	1
(h) Duffing (40)	0.2	0.088	0.204	0.048	0.156	0.75	1
(i) Lotka-Volterra (42)	0.1	0.013	0.023	0.120	0.365	1	1
(j) Lorenz (43)	0.2	0.011	0.041	0.032	2.380	1	1

Fig. 11. WeakIdent results for ODE systems in Table 2. (a)-(e): Given noisy data compared to the true dynamics. (f)-(j): Recovered systems via WeakIdent using true initial conditions. WeakIdent recovers the dynamics close to the true dynamics with a small identification error.

Table 4
Typical examples of the feature matrix size and the reduction in narrow-fit. The given data is of size $\mathbb{N}_x\mathbb{N}_t$ and it is subsampled to $H = N_x N_t$ rows for $\boldsymbol{W}$. We use $\sigma_{\text{NSR}} = 0.1$ for the RD equation (37) and $\sigma_{\text{NSR}} = 0.2$ for the rest of the equations. For systems of equations, the size of the feature matrix for each dependent variable is identical.

Equation	$\mathbb{N}_x$	$\mathbb{N}_t$	N_x	N_t	$\boldsymbol{W}$ size$(H \times L)$	$\tilde{\boldsymbol{W}}_{\text{narrow}}$ size
Linear Equ. (32)	257	300	36	39	1404 × 43	824 × 43
KdV Equ. (33)	400	601	71	65	4615 × 43	1367 × 43
KS Equ. (34)	256	301	46	43	1935 × 43	916 × 43
NLS Equ. (35)	256	251	39	42	1225 × 190	159 × 190
PM Equ. (36)	200 × 200	128	14 × 14	16	3136 × 65	1349 × 65
RD Equ. (37)	256 × 256	201	13 × 13	14	2366 × 155	2271 × 155
Linear Equ. (39)	-	1001	-	851	877 × 21	127 × 21
VdP Equ. (40)	-	15001	-	958	958 × 21	295 × 21
Duffing Equ. (41)	-	1001	-	915	915 × 21	57 × 21
LV Equ. (42)	-	1001	-	947	947 × 10	338 × 10
Lorenz Equ. (43)	-	15001	-	983	983 × 20	930 × 20

colored curves are decreasing functions in terms of $\mathcal{T}$, if the given data is highly corrupted by noise, using a larger $\mathcal{T}$ can help with the identification.

4.5. Effects of subsampling in data acquisition and the feature matrix W

In Fig. 15, we show the effects of changing the final time T (the top row), and of changing Δx and Δt for $\mathbb{N}_x\mathbb{N}_t$ (the second row), and of changing the uniform subsampling in (11), i.e., Δt^* and Δx^* for the generation of the feature matrix (the third row). We compare for the KS equation (34), the 2D linear ODE system (39), the Van der Pol equation (40), and the Duffing equation (41) to illustrate the effects. The noise level is $\sigma_{\text{NSR}} = 0.1$ for each example. We present the average of the E_2 error, the TPR and PPV values from 20 independent experiments for one varying variable among the variables $\{T, \Delta t, \Delta x, \Delta t^*, \Delta x^*\}$ while fixing the rest. The first row shows that the recovery by WeakIdent is robust as long as T is above a sufficiently large value (e.g. 100 or 10), which indicates that there is a time T such that the solution of the differential equations contains enough dynamics up to time T. The second row shows that WeakIdent gives a smaller error

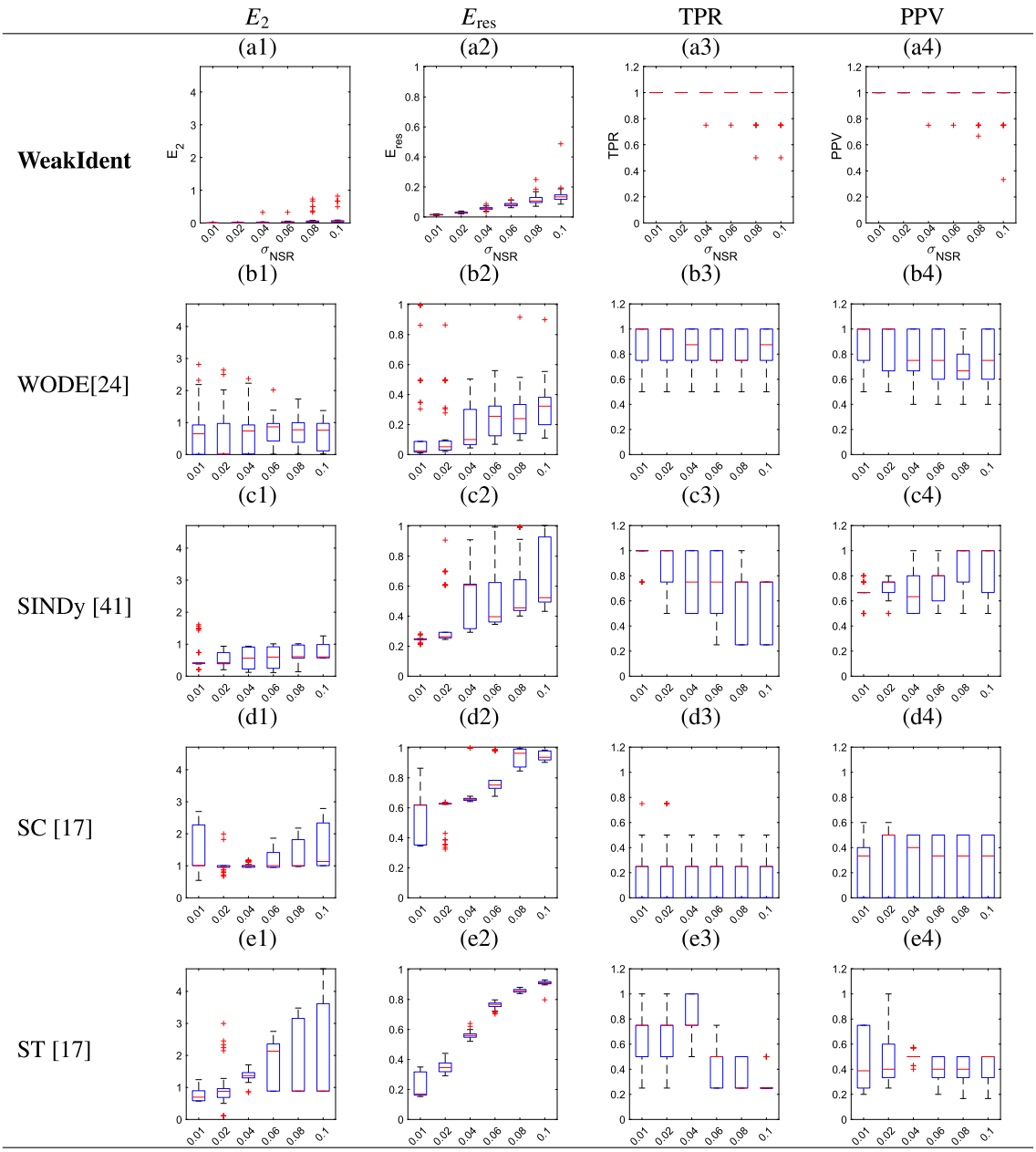


Fig. 12. The Lotka-Volterra equation (42). Statistical comparisons between (a1)-(a4) WeakIdent, (b1)-(b4) WODE [24], (c1)-(c4) SINDy [41], (d1)-(d4) SC [17] and (e1)-(e4) ST [17]. The E_2, E_{res} errors, TPR and PPV are shown from 50 experiments for each $\sigma_{\text{NSR}} \in \{0.01, 0.02, ..., 0.1\}$ using box-plots. Notice that for WeakIdent, the E_2 error is lower with less variations, and the TPR and PPV are closer to 1 as compared with that obtained from other methods.

with smaller Δx and Δt. The bottom row shows that the size of uniform subsampling in space and time of the feature matrix does not affect the recovery.

In Table 4, we show an example of the size reduction from $\mathbf{W}$ to $\mathbf{W}_{\text{narrow}}$ for the PDEs and ODEs considered in this paper. We use $\sigma_{\text{NSR}} = 0.1$ for the RD equation (37) and $\sigma_{\text{NSR}} = 0.2$ for the rest of the equations. The given data is of size $\mathbb{N}_x\mathbb{N}_t$ and it is subsampled to $H = N_x N_t$ number of rows for $\boldsymbol{W}$. The narrow-fit further reduces the feature matrix to $\tilde{\mathbf{W}}_{\text{narrow}}$ for computational accuracy.

4.6. Speed of WeakIdent

We perform experiments using Matlab on the Apple M1 processor with 8-core CPU and 16 GB of RAM. The computational cost of WeakIdent is typically about 1-5 seconds for an ODE system or a PDE with one dependent variable in a 1D spatial

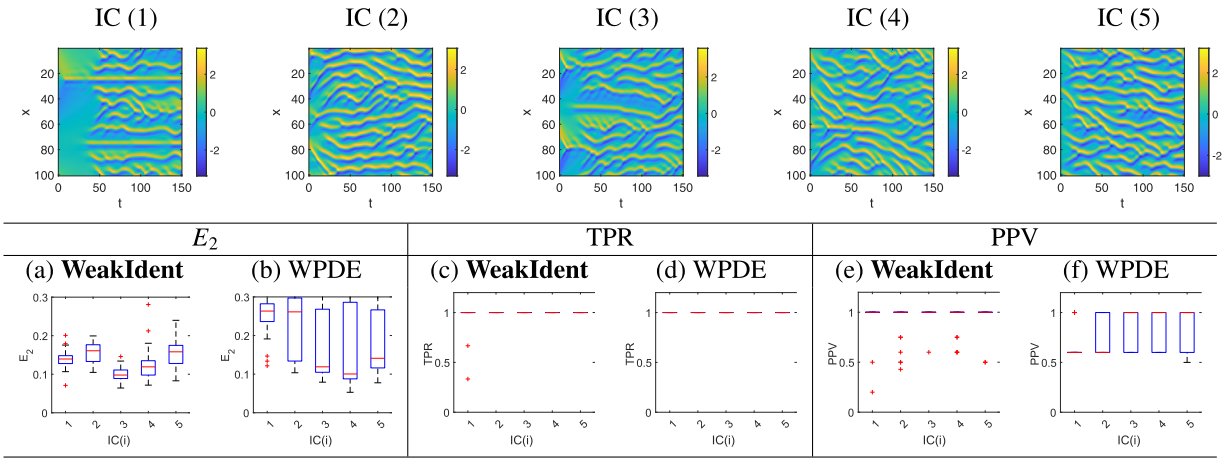


Fig. 13. The KS equation (34) using five different initial conditions (1)-(5) with the noisy level of $\sigma_{\text{NSR}} = 0.6$. In (a)-(f) the x-axis is the index of initial conditions (1)-(5). For each initial condition, the box plot represents the statistical results over 50 experiments. WeakIdent gives a smaller E_2 error, and PPV is closer to 1 with less variations.

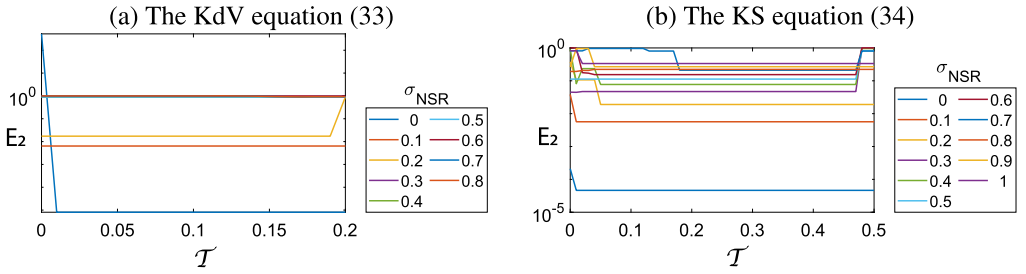


Fig. 14. The coefficient E_2 error (y-axis) versus the trimming parameter $\mathcal{T}$ (x-axis) for the identification of (a) the KdV equation (33) and (b) the KS equation (34). Different color curves represent results for various noise-to-signal ratios $\sigma_{\text{NSR}} \in \{0, 0.1, ..., 1\}$. Notice a wide range of $\mathcal{T}$ gives the same recovery.

domain. For example, the cpu times to recover the Lotka-Volterra system (42) and the KdV equation (33) are 1.11 and 0.63 seconds, respectively. For the cases in 2D spatial domains, such as the anisotropic PM equation (36) with one variable, and the 2D reaction-diffusion equation (37) with two variables, the recovery can take about 3 and 35 seconds, respectively. The speed is comparable with WPDE [23], which takes 16 and 75 seconds for these two examples. We note that the main difference in computation comes from using modified sequential thresholding least-squares (MSTLS) and Subspace Pursuit as in this paper. For the methods using MSTLS, thresholding lease square is performed for a large number (e.g., 50) of different λ (a parameter in MSTLS) to seek for a good threshold, so that the solutions are computed many times, while SP doesn't require to do this. For the computational cost scaling as the number of feature increases, with the trimming step, as in the case of Fig. 4, typical examples converged to the correct support for a smaller sparsity then L. One may be able to stop SP after a reasonable sparsity k is reached to reduce unnecessary computation.

In Supplementary Material 5, we present additional results and more comparisons. The additional results for PDEs are in Subsection 5.2.1 and additional results for ODEs are in Subsection 5.2.2. Details about how to construct test functions are given in Supplementary Material 5.3.

5. Conclusion and discussion

We propose a new method WeakIdent for identifying both PDEs and ODE systems from noisy data using a weak formulation. The proposed WeakIdent does not require prior knowledge of the governing features, but uses all features up to certain polynomial order, and up to certain order of derivatives. We first use Subspace Pursuit to find a candidate support, then propose two novel techniques called narrow-fit and trimming to improve both the support identification and the coefficient recovery. A careful design of the test functions helps with the recovery, and a proper normalization of the columns in the feature matrix improves the results in the implementation of least-squares. The proposed WeakIdent requires at most L sparsity iterations (or including the sub-iteration of narrow-fit and trimming, at most $\frac{L^2}{2}$ iterations), where L is the number of features. At the same time the trimming step improves the recovery and gives good results after a fraction of L is used to identify the correct support, as shown in Fig. 4. Narrow-fit based on highly dynamic regions also makes the computation

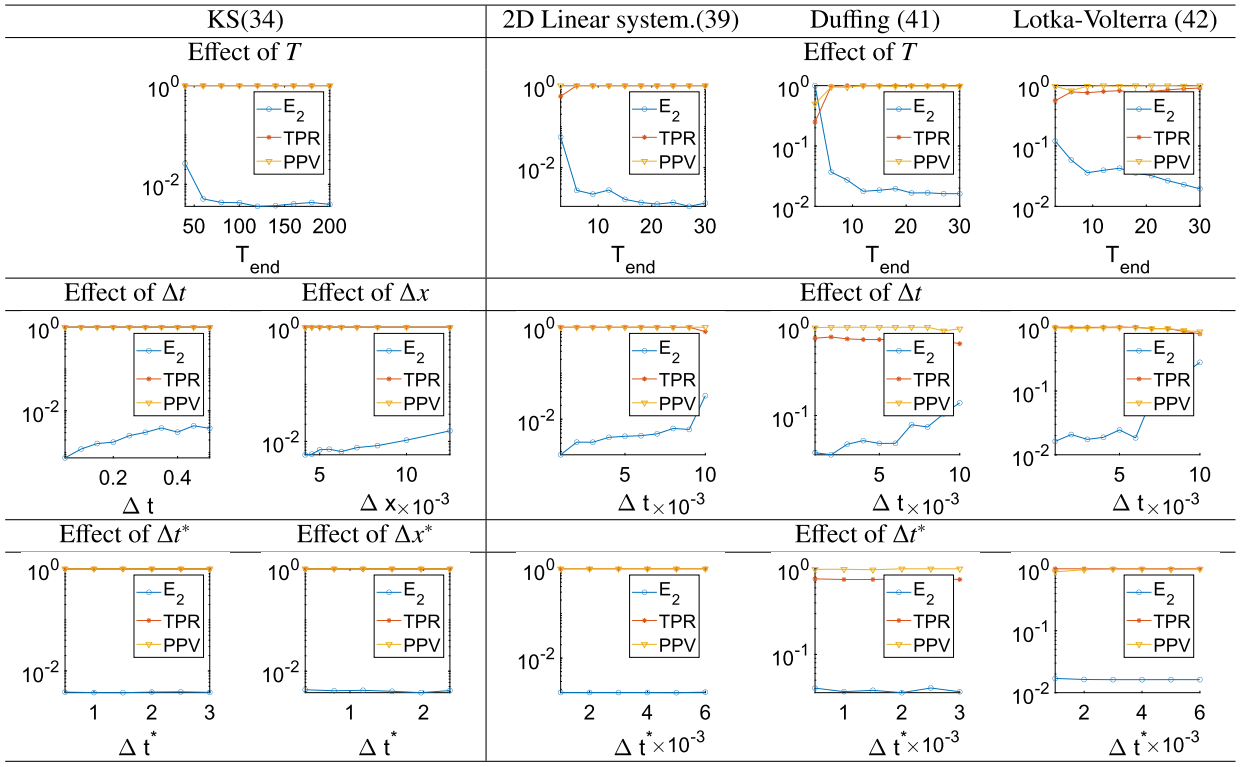


Fig. 15. Effects of the final time T (the top row), $\Delta t, \Delta x$ for $\mathbb{N}_x\mathbb{N}_t$ of the given data (the second row), and subsampling $\Delta t^*, \Delta x^*$ in (11) in the third row. Each graph shows the average of the E_2 error, the TPR and PPV values over 20 experiments for one varying variable among the variables in $\{T, \Delta x, \Delta t, \Delta x^*, \Delta t^*\}$ while the rest is fixed. The noise level is $\sigma_{\text{NSR}} = 0.1$. The left column gives the PDE results for the KS equation while both $\Delta t, \Delta x$ are shown. The right columns show Δt only for ODEs, including the 2D Linear system (39), the Duffing equation (41) and the Lotka-Volterra equation (42). There is a transition point in T such that the given data up to T contain enough dynamics. The recovery is in general better with smaller Δt and Δx, and the rate of uniform subsampling has a minimal effect on the results.

more efficient, and with error normalization of the feature matrix, the coefficient recovery is improved. Comprehensive numerical experiments on various equations/systems are provided, showing the robust performance of WeakIdent compared to other state-of-the-art methods. The Weak form in general is effective when the noise level is high, at the same time, to take advantage of the weak form, the possible features in the differential equation must be in a specific form for the integration of parts.

CRediT authorship contribution statement

Mengyi Tang: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft. **Wenjing Liao:** Conceptualization, Funding acquisition, Methodology, Writing – review & editing. **Rachel Kuske:** Conceptualization, Funding acquisition, Methodology, Writing – review & editing. **Sung Ha Kang:** Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to our code in the paper (https://github.com/sunghakang/WeakIdent).

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcp.2023.112069.

References

[1] L.H. Favela, The dynamical renaissance in neuroscience, Synthese 199 (2021) 2103–2127.
[2] J. Bongard, H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 104 (2007) 9943–9948.
[3] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (2009) 81–85.
[4] J.T. Nardini, J.H. Lagergren, A. Hawkins-Daarud, L. Curtin, B. Morris, E.M. Rutter, K.R. Swanson, K.B. Flores, Learning equations from biological data with limited time samples, Bull. Math. Biol. 82 (2020) 1–33.
[5] E. Baake, M. Baake, H. Bock, K. Briggs, Fitting ordinary differential equations to chaotic data, Phys. Rev. A 45 (1992) 5524.
[6] M. Bär, R. Hegger, H. Kantz, Fitting partial differential equations to space-time dynamics, Phys. Rev. E 59 (1999) 337.
[7] H.G. Bock, Recent advances in parameter identification techniques for ode, in: Numerical Treatment of Inverse Problems in Differential and Integral Equations, 1983, pp. 95–121.
[8] T.G. Müller, J. Timmer, Parameter identification techniques for partial differential equations, Int. J. Bifurc. Chaos Appl. Sci. Eng. 14 (2004) 2053–2060.
[9] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (2016) 3932–3937.
[10] L. Zhang, H. Schaeffer, On the convergence of the sindy algorithm, Multiscale Model. Simul. 17 (2019) 948–972.
[11] K. Kaheman, J.N. Kutz, S.L. Brunton, Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics, Proc. Math. Phys. Eng. Sci. 476 (2020).
[12] S.H. Rudy, A. Alla, S.L. Brunton, J.N. Kutz, Data-driven identification of parametric partial differential equations, SIAM J. Appl. Dyn. Syst. 18 (2019) 643–660.
[13] J.-C. Loiseau, B.R. Noack, S.L. Brunton, Sparse reduced-order modelling: sensor-based dynamics to full-state estimation, J. Fluid Mech. 844 (2018) 459–490.
[14] Y. Guan, S.L. Brunton, I.V. Novosselov, Sparse nonlinear models of chaotic electroconvection, R. Soc. Open Sci. 8 (2021).
[15] K.P. Champion, S.L. Brunton, J.N. Kutz, Discovery of nonlinear multiscale systems: sampling strategies and embeddings, SIAM J. Appl. Dyn. Syst. 18 (2019) 312–333.
[16] S.H. Kang, W. Liao, Y. Liu, Ident: identifying differential equations with numerical time evolution, J. Sci. Comput. 87 (2021) 1–27.
[17] Y. He, S.H. Kang, W. Liao, H. Liu, Y. Liu, Robust pde identification from noisy data, arXiv preprint, arXiv:2006.06557, 2020.
[18] G. Tran, R. Ward, Exact recovery of chaotic systems from highly corrupted data, Multiscale Model. Simul. 15 (2017) 1108–1129.
[19] H. Schaeffer, G. Tran, R. Ward, L. Zhang, Extracting structured dynamical systems using sparse optimization with very few samples, Multiscale Model. Simul. 18 (2020) 1435–1461.
[20] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (2017) e1602614.
[21] K. Wu, D. Xiu, Numerical aspects for approximating governing equations using data, J. Comput. Phys. 384 (2019) 200–221.
[22] D. Gurevich, P.A.K. Reinbold, R.O. Grigoriev, Robust and optimal sparse regression for nonlinear pde models, Chaos 29 (10) (2019) 103113.
[23] D.A. Messenger, D.M. Bortz, Weak sindy for partial differential equations, J. Comput. Phys. (2021) 110525.
[24] D.A. Messenger, D.M. Bortz, Weak sindy: Galerkin-based data-driven model selection, Multiscale Model. Simul. 19 (2021) 1474–1497.
[25] P.A. Reinbold, D.R. Gurevich, R.O. Grigoriev, Using noisy or incomplete data to discover models of spatiotemporal dynamics, Phys. Rev. E 101 (2020) 010203.
[26] H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proc. R. Soc. A, Math. Phys. Eng. Sci. 473 (2017) 20160446.
[27] S. Zhang, G. Lin, Robust data-driven discovery of governing physical laws with error bars, Proc. R. Soc. A, Math. Phys. Eng. Sci. 474 (2018) 20180305.
[28] Z. Chen, V. Churchill, K. Wu, D. Xiu, Deep neural network modeling of unknown partial differential equations in nodal space, J. Comput. Phys. 449 (2022) 110782.
[29] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[30] B. Lusch, J.N. Kutz, S.L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nat. Commun. 9 (2018) 1–10.
[31] M. Raissi, G.E. Karniadakis, Hidden physics models: machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[32] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.
[33] H. Xu, H. Chang, D. Zhang, Dl-pde: deep-learning based data-driven discovery of partial differential equations from discrete and noisy data, arXiv preprint, arXiv:1908.04463, 2019.
[34] H. Xu, H. Chang, D. Zhang, Dlga-pde: discovery of pdes with incomplete candidate library via combination of deep learning and genetic algorithm, J. Comput. Phys. 418 (2020) 109584.
[35] Y. He, S.H. Kang, W. Liao, H. Liu, Y. Liu, Numerical identification of nonlocal potential in aggregation, Commun. Comput. Phys. (2022).
[36] S. Rudy, A. Alla, S.L. Brunton, J.N. Kutz, Data-driven identification of parametric partial differential equations, SIAM J. Appl. Dyn. Syst. 18 (2019) 643–660.
[37] Z. Chen, K. Wu, D. Xiu, Methods to recover unknown processes in partial differential equations using data, ArXiv abs, arXiv:2003.02387, 2020.
[38] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, IEEE Trans. Inf. Theory 55 (2009) 2230–2249.
[39] Å. Björck, Least squares methods, Handb. Numer. Anal. 1 (1990) 465–652.
[40] Å. Björck, Error analysis of least squares algorithms, in: Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, Springer, 1991, pp. 41–73.
[41] S.L. Brunton, J.L. Proctor, J.N. Kutz, Sparse identification of nonlinear dynamics with control (sindyc), IFAC-PapersOnLine 49 (2016) 710–715.