Sparse discovery of differential equations based on multi-fidelity Gaussian process

Meng, Yuhuang; Qiu, Yue

DOI
10.1016/j.jcp.2024.113651

Publication date
2025

Document Version
Final published version

Published in
Journal of Computational Physics

Journal of Computational Physics 523 (2025) 113651

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Sparse discovery of differential equations based on multi-fidelity Gaussian process ☆

Yuhuang Meng [a], Yue Qiu [b,c,d,*]

[a] Delft Institute of Applied Mathematics, Delft University of Technology, Delft, 2628 CD, the Netherlands
[b] College of Mathematics and Statistics, Chongqing University, Chongqing, 401331, China
[c] Key Laboratory of Nonlinear Analysis and its Applications (Chongqing University), Ministry of Education, Chongqing, 401331 China
[d] Innovation Center for Mathematical Analysis of Fluids and Chemotaxis, Chongqing University, Chongqing, 401331 China

ARTICLE INFO

Keywords:
Sparse discovery
Gaussian process regression
Multi-fidelity data
Uncertainty quantification

ABSTRACT

Sparse identification of differential equations aims to compute the analytic expressions from the observed data explicitly. However, there exist two primary challenges. Firstly, it exhibits sensitivity to the noise in the observed data, particularly for the derivatives computations. Secondly, existing literature predominantly concentrates on single-fidelity (SF) data, which imposes limitations on its applicability due to the computational cost. In this paper, we present two novel approaches to address these problems from the view of uncertainty quantification. We construct a surrogate model employing the Gaussian process regression (GPR) to mitigate the effect of noise in the observed data, quantify its uncertainty, and ultimately recover the equations accurately. Subsequently, we exploit the multi-fidelity Gaussian processes (MFGP) to address scenarios involving multi-fidelity (MF), sparse, and noisy observed data. We demonstrate the robustness and effectiveness of our methodologies through several numerical experiments.

1. Introduction

Nonlinear differential equations are widely prevalent in both science and engineering applications. Nonetheless, these equations are generally unknown in many situations, which makes it difficult to understand and control the systems of interest. Fortunately, amounts of data concerning the underlying systems can be available through experiments or simulations. To address this challenge, data-driven approaches have emerged. For instance, the Koopman operator theory and dynamic mode decomposition (DMD) embed a nonlinear system into a higher-dimensional linear space via a set of observation functions, which provides a powerful and efficient tool for understanding the behavior of the nonlinear dynamic systems, especially in fluid dynamics [1–4].

In recent years, numerous efforts have been dedicated to learn these nonlinear equations from the observed data. Based on the diversity of the final results, these methods can be broadly categorized into two classes. The first develops black-box models to approximate the underlying differential equations. Such techniques rely on deep neural networks and numerical schemes. Raissi et al. [5] combined the multistep method with deep neural networks to discover nonlinear ordinary differential equations (ODEs). They

☆ This work is partially supported by the National Natural Science Foundation of China (NSFC) under grant number 12471404, the Fundamental Research Funds for the Central Universities under grant number 2024CDJYXTD-009 and 2024IAIS-QN010, and the Chongqing Entrepreneurship and Innovation Program for Returned Overseas Scholars under grant number CX2023068.

* Corresponding author at: College of Mathematics and Statistics, Chongqing University, Chongqing, 401331, China.
E-mail addresses: y.meng@tudelft.nl (Y. Meng), qiuyue@cqu.edu.cn (Y. Qiu).

https://doi.org/10.1016/j.jcp.2024.113651
Received 29 January 2024; Received in revised form 7 October 2024; Accepted 2 December 2024

Available online 5 December 2024

represented the right-hand-side of ODEs using deep neural networks. In a similar vein, Raissi [6] harnessed deep neural networks and employed automatic differentiation to compute the partial derivatives. Rudy et al. [7] introduced a robust identification algorithm for dynamic systems that combines neural networks with the Runge-Kutta method to denoise the data and discover the dynamic system simultaneously. Qin et al. [8] used a residual network to approximate the integral form of the underlying ODEs. Conversely, the second approach seeks an analytic expression for the hidden differential equations. Sparse Identification of Nonlinear Dynamic Systems (SINDy), introduced by Brunton et al. [9], has emerged as a significant framework. It assumes the underlying differential equations could be represented by few predefined functions. The core procedure of SINDy involves the computation of derivatives and the solution of linear equations while imposing constraints for a sparse solution. Long et al. [10] employed convolutional neural networks to learn the symbolic representations of partial differential equations (PDEs). Kim et al. [11] pursued symbolic regression using neural networks. Kang et al. [12] proposed the so-called IDENT to obtain sparse solutions through Lasso and validated these solutions by assessing time evolution errors.

SINDy has proven to be a successful tool in the discovery of PDEs, primarily due to its simplicity and efficiency [13]. However, there remain two critical challenges. The first involves robust approximation of the temporal or spatial partial derivatives when the observed data is corrupted by noise. In practice, only the states observed data is available whereas the temporal derivatives or partial derivatives are required to be calculated using numerical differentiation methods. However, these numerical differentiation methods magnify the noise in the observed data, which makes SINDy sensitive to noise. The second revolves around effectively leveraging the multi-fidelity data to reduce the computational cost.

Regarding the first challenge, various robust algorithms have emerged. The first category constructs a local polynomial surrogate model aimed to smooth the noisy observed data and obtain the derivatives analytically. For instance, He et al. [14] employed the moving least squares (MLS) to smooth the observed data and enhance the stability of numerical differentiation when applied to noisy datasets. Similarly, Sun et al. [15] represented the observed data by a set of cubic spline basis functions and the surrogate model and discovered equations are trained simultaneously. Notably, the noisy data is fitted by local quadratic and cubic polynomials in [14] and [15], respectively. Analogous methodologies have appeared in [16–18]. While these approaches show promising results in specific scenarios, they primarily rely on the local regression models that entail manual selection of parameters (such as the window length) and require analysis case-by-case. The second category seeks to recover nonlinear equations from the perspective of sparse Bayesian learning. Relevance vector machine (RVM) was introduced to infer the nonlinear equations [19,20]. Furthermore, Tikhonov regularization is utilized for derivatives computations [13,21], and neural networks were employed as a surrogate model to smooth the data, which transforms numerical differentiation to automatic differentiation to decrease the impact of noise [22,23]. Moreover, in addition to the aforementioned approaches to smooth noisy data, there are several derivative-free methods. Notably, RK4-SINDy eliminates the approximations of derivatives by employing the fourth-order Runge-Kutta method [24]. On the other hand, weak SINDy (WSINDy) avoids the derivative approximations by constructing the weak form of the underlying differential equations [25,26].

As for the second challenge, related work mainly concentrate on the single-fidelity, namely high-fidelity (HF) data. Nevertheless, the accurate simulation of HF data suffers from the large computational cost. To address this issue, the integration of a small amount of HF data with suitable low-fidelity (LF) observed data which may be of lower accuracy but also lower cost, becomes a feasible approach. This is commonly referred to as the multi-fidelity modeling. The problem of harnessing multi-fidelity data for sparse identification remains unresolved. One classical MF modeling technique is the linear autoregressive method known as co-kriging [27,28], where each fidelity level model is represented by a Gaussian process and the relationship between the outputs of the LF and HF model is assumed to be linear. Perdikaris et al. [29] proposed the nonlinear autoregressive multi-fidelity GP regression (MFGP) scheme which extends the capabilities of the linear autoregressive approach. Moreover, recent developments have seen the emergence of deep neural networks for multi-fidelity modeling [30–32].

This paper aims to address the aforementioned challenges by introducing two methodologies. Firstly, we propose the Gaussian Process based Sparse Identification of Nonlinear Dynamics (GP-SINDy) to construct a global surrogate model and alleviate the effect of noise in the observed data. GP-SINDy leverages the non-parametric model GPR to effectively smooth the noisy data. Meanwhile, the derivatives are computed analytically within the GP framework. Notably, GPR provides valuable uncertainty quantification (UQ) information for the variables inference. We incorporate this UQ information into the weighted least squares (WLS) problem to ensure the accurate recovery of the potential functions. Furthermore, we introduce the Multi-Fidelity Gaussian Process based Sparse Identification of Nonlinear Dynamics (MFGP-SINDy) to infer the explicit representation of the differential equations using a suitable amount of LF data and limited HF data. To achieve this, MFGP [29] is utilized for the information fusion among different fidelity levels. The main contributions of this paper include:

1. A robust algorithm for sparse identification of differential equation using GP, aka GP-SINDy, is proposed.
2. The uncertainty of time derivative is approximated by its posterior variance in GP and this UQ information are embodied into the process of sparse identification of differential equation.
3. MFGP-SINDy is developed to cope with the case of multi-fidelity, sparse, and noisy observed data. Meanwhile, the partial derivative computations of MFGP kernel are provided, which is the key for MFGP-SINDy.

This paper is organized as follows. We present the problem of sparse discovery for differential equations in Section 2. In Section 3, we briefly review the Gaussian process regression and present our GP-SINDy and MFGP-SINDy algorithms. Several numerical experiments in Section 4 are carried out to demonstrate the efficiency and robustness of our methods, and we summarize our paper in Section 5.

2. Problem statement

Consider a (nonlinear) differential equation (ODE or PDE) described by the following form

$$\frac{d}{dt}\mathbf{u} = \mathbf{f}(\mathbf{u}), \tag{1}$$

where $\mathbf{u} = [u_1(\mathbf{x}), u_2(\mathbf{x}), \cdots, u_d(\mathbf{x})]^T \in \mathbb{R}^d$ represents the state variables, the (nonlinear) evolution $\mathbf{f}(\mathbf{u})$ of the system is unknown, and $\mathbf{x} \in \mathbb{R}^D$ represent the time (and the space location), where $\mathbf{x} = t$ for ODEs, and $\mathbf{x} = (t, x)$ for PDEs, with $t \in [0, T]$. For PDEs, the right-hand-side of (1) also contains the partial derivatives with respect to (w.r.t.) x, *i.e.*, $\mathbf{f}(\mathbf{u}) = \mathbf{f}(\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \cdots)$.

The available observed data is represented by $\mathcal{D} = \{\mathbf{x}_i, \mathbf{u}_i\}$ for $i = 1, \cdots, N$, where $\mathbf{u}_i = \mathbf{u}_i^* + \boldsymbol{\varepsilon}_i$. Here, $\mathbf{u}_i^*$ denotes the clean data sampled from the true system, and $\boldsymbol{\varepsilon}_i$ is independent and identically distributed (i.i.d.) Gaussian white noise with $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}_d, \sigma^2 \mathbf{I}_d)$, where $\mathbf{0}_d$ and $\mathbf{I}_d$ denote the zero vector of size d and the identity matrix of size $d \times d$, respectively. We could assemble the data $\mathcal{D}$ into the matrix form with $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]^T \in \mathbb{R}^{N\times d}$ and

$$\mathbf{U} = \mathbf{U}^* + \mathbf{E},$$

where $\mathbf{U}^*$ represents the corresponding clean data matrix and $\mathbf{E}$ is the noise matrix. Our objective is to discover the explicit expression of $\mathbf{f}(\mathbf{u})$ based on the noisy observed data $\mathbf{U}$.

Assume that $\mathbf{f}(\mathbf{u})$ can be expressed by a linear combination of as few as predefined functions from a function library denoted by $\boldsymbol{\Phi}(\mathbf{u}) = [\phi_1(\mathbf{u}), \phi_2(\mathbf{u}), \cdots, \phi_{N_f}(\mathbf{u})]$ where each ϕ_i is referred to as a basis function or function feature, and N_f is the size of the function library. The function library $\boldsymbol{\Phi}(\mathbf{u})$ is often selected as the polynomial basis functions or trigonometric functions. For PDE systems, the basis functions also involve the partial derivatives w.r.t. the space variables, such as $\mathbf{u}_x$, $\mathbf{u}_{xx}$, and their combination functions $\mathbf{u}\mathbf{u}_x$, $\mathbf{u}\mathbf{u}_{xx}$, and $\mathbf{u}_x\mathbf{u}_{xx}$. The task of discovering the (nonlinear) system (1) can be reformulated as the problem of solving the following linear equations while enforcing the constraint of sparse solution,

$$\dot{\mathbf{U}} = \boldsymbol{\Phi}\mathbf{C} + \mathbf{E}'. \tag{2}$$

Here, $\dot{\mathbf{U}} \in \mathbb{R}^{N\times d}$ represents the temporal derivatives of the state variables, which needs to be computed from the given data matrix $\mathbf{X}$ and $\mathbf{U}$. $\mathbf{E}' \in \mathbb{R}^{N\times d}$ is the noise matrix. The matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N\times N_f}$ corresponds to the function library, and $\mathbf{C} \in \mathbb{R}^{N_f\times d}$ represents the sparse coefficient matrix to be determined.

The problem of sparse discovery of differential equations described by (2) involves solving a linear least square problem. Partition $\mathbf{U}$, $\dot{\mathbf{U}}$ and $\mathbf{C}$ using $\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_d]$, $\dot{\mathbf{U}} = [\dot{\mathbf{U}}_1, \dot{\mathbf{U}}_2, \cdots, \dot{\mathbf{U}}_d]$, and $\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_d]$, where $\mathbf{U}_i$, $\dot{\mathbf{U}}_i \in \mathbb{R}^N$, and $\mathbf{C}_i \in \mathbb{R}^{N_f}$ represent the i-th column of matrices $\mathbf{U}$, $\dot{\mathbf{U}}$, and $\mathbf{C}$, respectively. We aim to compute a sparse solution $\mathbf{C}_i \in \mathbb{R}^{N_f}$ with

$$\min_{\mathbf{C}_i} \left(\dot{\mathbf{U}}_i - \boldsymbol{\Phi}\mathbf{C}_i\right)^T \mathbf{W}_i \left(\dot{\mathbf{U}}_i - \boldsymbol{\Phi}\mathbf{C}_i\right) + \lambda_0 \|\mathbf{C}_i\|_0, \tag{3}$$

for each $i = 1, 2, \cdots, d$. We transform the problem of sparse discovery of differential equations into a WLS problem.

Here $\mathbf{W}_i$ is the weight matrix and should be positive semi-definite and the ℓ_0 regularization term is employed to promote the sparsity of $\mathbf{C}_i$. If the weight matrix $\mathbf{W}_i$ is chosen as the identity matrix, this problem degenerates to an ordinary least squares formulation which is studied in classical SINDy [9,13].

3. Sparse identification using Gaussian process

In this section, we propose two methods that leverage the GPR and MFGP to recover the nonlinear differential equations, named GP-SINDy and MFGP-SINDy, respectively. First, we briefly review some fundamental concepts of GPR. Subsequently, we present the GP-SINDy approach, which incorporates the inference outcomes of GPR into the sparse discovery of nonlinear systems. Finally, we employ MFGP to fuse the nonlinear information among different fidelity levels for the sparse identification.

3.1. Gaussian process regression

Assume that the observation $\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) \middle| , i = 1, \cdots, N \right\}$ satisfies the following model,

$$y_i = f(\mathbf{x}_i) + \varepsilon_i.$$

Here, $\mathbf{x}_i \in \Omega \subset \mathbb{R}^D$, and the scalar y_i is contaminated by i.i.d. noise $\varepsilon_i \sim \mathcal{N}(0, \sigma_0^2)$. The function f could be characterized by a Gaussian process, *i.e.*, $f(\mathbf{x}) \sim \mathcal{GP}\left(\overline{f(\mathbf{x})}, k\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}\right)\right)$, where $\boldsymbol{\theta}$ is the hyperparameters to be determined, the mean function $\overline{f(\mathbf{x})}$ and the covariance function (kernel function) $k\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}\right)$ are defined by,

$$\begin{cases} \overline{f(\mathbf{x})} = \mathbb{E}(f(\mathbf{x})), \\ k\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}\right) = \mathbb{E}\left[\left(f(\mathbf{x}) - \overline{f(\mathbf{x})}\right)^T \left(f\left(\mathbf{x}'\right) - \overline{f\left(\mathbf{x}'\right)}\right)\right]. \end{cases}$$

In this paper, we choose the squared exponential (SE) kernel, *i.e.*, $k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\boldsymbol{\theta}\right)=\theta_0^2\exp\left(-\sum_{s=1}^{D}\frac{\left(\mathbf{x}_s-\mathbf{x}'_s\right)^2}{2\theta_s^2}\right)$. For simplicity, we denote the training data by $\mathbf{X}=[\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_N]\in\mathbb{R}^{D\times N}$ and $\mathbf{y}=[y_1,y_2,\cdots,y_N]^T\in\mathbb{R}^N$.

3.1.1. Inferring the state variables

GPR provides a non-parameter model that allows one to infer the quantities of interest. Given the training data $\mathbf{X}$ and $\mathbf{y}$, our objective is to infer the value and uncertainty of $f^*=f(\mathbf{x}^*)$ where $\mathbf{x}^*\in\mathbb{R}^D$ is called the test data. The joint Gaussian distribution can be expressed by,

$$\begin{bmatrix}\mathbf{y}\\ f^*\end{bmatrix}\sim\mathcal{N}\left(\mathbf{0}_{N+1},\begin{bmatrix}\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N & \mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{x}^*)\\ \mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{X}) & \mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{x}^*)\end{bmatrix}\right). \tag{4}$$

Here, the covariance matrix of the training data denoted by $\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})\in\mathbb{R}^{N\times N}$ is defined such that $\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})_{i,j}=k\left(\mathbf{x}_i,\mathbf{x}_j;\boldsymbol{\theta}\right)$, where the subscript "SS" represents the covariance between states. $\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{x}^*)\in\mathbb{R}^N$ represents the cross-covariance matrix between the training data $\mathbf{X}$ and the test data $\mathbf{x}^*$ with $\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{x}^*)_i=k\left(\mathbf{x}_i,\mathbf{x}^*;\boldsymbol{\theta}\right)$. $\mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{X})=\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{x}^*)^T$ and $\mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{x}^*)=k\left(\mathbf{x}^*,\mathbf{x}^*;\boldsymbol{\theta}\right)$.

The posterior distribution, which represents the conditional distribution of f^* given $\mathbf{X}$, $\mathbf{y}$, $\mathbf{x}^*$, and $\boldsymbol{\theta}$, can be expressed by

$$f^*\Big|\mathbf{X},\mathbf{y},\mathbf{x}^*,\boldsymbol{\theta}\sim\mathcal{N}\left(\overline{f^*},\mathrm{var}(f^*)\right),$$

where the mean and variance of the posterior distribution are calculated by

$$\begin{cases}\overline{f^*}=\mathbb{E}[f^*|\mathbf{X},\mathbf{y},\mathbf{x}^*,\boldsymbol{\theta}]=\mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{X})\left[\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right]^{-1}\mathbf{y},\\ \mathrm{var}(f^*)=\mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{x}^*)-\mathbf{K}_{\mathrm{SS}}(\mathbf{x}^*,\mathbf{X})\left[\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right]^{-1}\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{x}^*).\end{cases} \tag{5}$$

3.1.2. Inferring the partial derivatives of state variables

Since differentiation is a linear operator, the derivative of a Gaussian process remains a Gaussian process [33]. Beyond inferring the mean and variance of the states data $f^*=f(\mathbf{x}^*)$, GPR enables the derivation of the partial derivatives, such as the first-order partial derivative with respect to the j-th component $\mathbf{x}_j^*$, denoted by $(\partial f^*)_j=\frac{\partial f(\mathbf{x}^*)}{\partial\mathbf{x}_j^*}$. Similar to the formulas in (4), a joint Gaussian distribution encompassing the derivatives and states data can be expressed by,

$$\begin{bmatrix}\mathbf{y}\\ (\partial f^*)_j\end{bmatrix}\sim\mathcal{N}\left(\mathbf{0},\begin{bmatrix}\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N & \mathbf{K}_{\mathrm{SD}}(\mathbf{X},\mathbf{x}^*)\\ \mathbf{K}_{\mathrm{DS}}(\mathbf{x}^*,\mathbf{X}) & \mathbf{K}_{\mathrm{DD}}(\mathbf{x}^*,\mathbf{x}^*)\end{bmatrix}\right),$$

where the subscript "S" and "D" represent "State" and "Derivative", respectively. $\mathbf{K}_{\mathrm{SD}}(\mathbf{X},\mathbf{x}^*)\in\mathbb{R}^N$ and $\mathbf{K}_{\mathrm{DS}}(\mathbf{x}^*,\mathbf{X})=\mathbf{K}_{\mathrm{SD}}(\mathbf{X},\mathbf{x}^*)^T$ represent two covariance matrices between the training data $\mathbf{X}$ and the test data $\mathbf{x}^*$, where $\mathbf{K}_{\mathrm{SD}}(\mathbf{x},\mathbf{x}^*)_{i,1}=\left.\frac{\partial k(\mathbf{x},\mathbf{x}';\boldsymbol{\theta})}{\partial\mathbf{x}'_j}\right|_{\mathbf{x}=\mathbf{x}_i,\mathbf{x}'=\mathbf{x}^*}$. Similarly, $\mathbf{K}_{\mathrm{DD}}(\mathbf{x}^*,\mathbf{x}^*)=\left.\frac{\partial^2 k(\mathbf{x},\mathbf{x}';\boldsymbol{\theta})}{\partial\mathbf{x}_j\partial\mathbf{x}'_j}\right|_{\mathbf{x}=\mathbf{x}^*,\mathbf{x}'=\mathbf{x}^*}$.

The conditional distribution of $(\partial f^*)_j$ given $\mathbf{X},\mathbf{y},\mathbf{x}^*,\boldsymbol{\theta}$, can be given by,

$$(\partial f^*)_j\Big|\mathbf{X},\mathbf{y},\mathbf{x}^*,\boldsymbol{\theta}\sim\mathcal{N}\left(\overline{(\partial f^*)}_j,\mathrm{var}((\partial f^*)_j)\right),$$

where the mean and variance of the posterior distribution are calculated by

$$\begin{cases}\overline{(\partial f^*)}_j=\mathbb{E}[(\partial f^*)_j|\mathbf{X},\mathbf{y},\mathbf{x}^*,\boldsymbol{\theta}]=\mathbf{K}_{\mathrm{DS}}(\mathbf{x}^*,\mathbf{X})\left[\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right]^{-1}\mathbf{y},\\ \mathrm{var}((\partial f^*)_j)=\mathbf{K}_{\mathrm{DD}}(\mathbf{x}^*,\mathbf{x}^*)-\mathbf{K}_{\mathrm{DS}}(\mathbf{x}^*,\mathbf{X})\left[\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right]^{-1}\mathbf{K}_{\mathrm{SD}}(\mathbf{X},\mathbf{x}^*).\end{cases} \tag{6}$$

The inference of the partial derivatives is utilized for the terms such as u_t in ODEs, as well as $u_t,u_x,u_y,u_{xx},u_{yy}$ in PDEs. Here, we focus solely on demonstrating how to infer the first-order partial derivatives. Further details regarding the derivation of the first-order and higher-order partial derivatives of the SE kernel are given in Appendix A.1.

3.1.3. Training the hyperparameters in GP

The marginal likelihood at $\mathbf{X}$ can be described by

$$p(\mathbf{y}|\mathbf{X})=\int p(\mathbf{y}|\mathbf{f},\mathbf{X})p(\mathbf{f}|\mathbf{X})\mathrm{d}\mathbf{f},$$

where the priori $p(\mathbf{f}|\mathbf{X})$ is a Gaussian distribution, *i.e.*, $\mathbf{f}|\mathbf{X}\sim\mathcal{N}(\mathbf{0},\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X}))$ [33]. Meanwhile, due to the existence of noise in GP and $p(\mathbf{y}|\mathbf{f},\mathbf{X})=p(\mathbf{y}|\mathbf{f})$, we have $\mathbf{y}|\mathbf{f},\mathbf{X}\sim\mathcal{N}\left(\mathbf{f},\sigma_0^2\mathbf{I}_N\right)$. Then, we obtain $\mathbf{y}|\mathbf{X}\sim\mathcal{N}\left(\mathbf{0},\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right)$. The value of hyperparameters $\boldsymbol{\theta}$ in the kernel of GP, as well as the noise variance σ_0^2 are determined by maximizing the likelihood function of $\mathbf{y}|\mathbf{X}$, which is equivalent to minimizing the negative log marginal likelihood given by,

$$\mathcal{L}_{\mathrm{GP}}\left(\boldsymbol{\theta},\sigma_0^2;\boldsymbol{y},\mathbf{X}\right)=\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\left(\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right)^{-1}\boldsymbol{y}+\frac{1}{2}\log\left|\mathbf{K}_{\mathrm{SS}}(\mathbf{X},\mathbf{X})+\sigma_0^2\mathbf{I}_N\right|+\frac{N}{2}\log(2\pi). \tag{7}$$

3.2. GP-SINDy for single-fidelity data

In this part, we propose the GP-SINDy algorithm, which utilizes $\mathbf{X}$ and the noisy observed data and $\mathbf{U}$ to compute the sparse solution $\mathbf{C}$ for the WLS problem given by (3). GP-SINDy involves four steps,

1. Constructing the GP surrogate model based on observed data $\mathbf{X}, \mathbf{U}$;
2. Inferring the states variable, and their partial derivatives;
3. Assembling the derivatives matrix $\dot{\mathbf{U}}_i$ and function library matrix $\mathbf{\Phi}$;
4. Resolving the WLS problem (3).

Since the dimension of the state variable $\mathbf{u}$ is d and the single output GP is employed, it becomes imperative to compute the sparse coefficients for each individual dimension of $\mathbf{u}$. Now we need to construct d independent GPR models in order to extract information for each dimension, including the states and the associated derivatives. The i-th GPR model is constructed based on training data $\{\mathbf{X}, \mathbf{U}_i\}$ and the kernel function $k\left(\mathbf{x}, \mathbf{x}'; \theta_i\right)$, where the optimal hyperparameters θ_i and noise variance σ_i^2 are determined by minimizing equation (7).

Then, we perform the inference at N' test inputs (denoted by $\mathbf{X}_p \in \mathbb{R}^{D\times N'}$) using equation (5) and (6). This inference involves three parts: the states, the posterior mean and variance of partial derivatives w.r.t. time denoted by $\overline{\mathbf{U}}_i$, $\overline{\dot{\mathbf{U}}}_i$, and $\mathrm{var}(\dot{\mathbf{U}}_i)$, respectively. Here, $\mathrm{var}(\dot{\mathbf{U}}_i)$ is a matrix with size $N' \times N'$. For PDEs, we also need to calculate the partial derivatives w.r.t. the spatial variables denoted by $\overline{\mathbf{V}}_i$, such as first-order and second-order partial derivatives matrices $\overline{\mathbf{V}}_i$ (order = 1) and $\overline{\mathbf{V}}_i$ (order = 2). Typically, to ensure accuracy, the size of the prediction is set to be greater than the size of the training data, *i.e.*, $N' \geq N$.

Remark 1. Standard GPR has cubic computational complexity when learning the hyperparameters and predicting target values. To alleviate this computational burden, various algorithms have been developed [34,35]. Nonetheless, to keep a clear structure of this paper, we utilize conventional GPR with a constrained dataset size to maintain the computational tractability. Furthermore, at the stage of hyperparameters optimization, subsampling of the training data is an optional technique to considerably diminish the runtime.

Let $\overline{\mathbf{U}} = [\overline{\mathbf{U}}_1, \overline{\mathbf{U}}_2, \cdots, \overline{\mathbf{U}}_d]$, $\overline{\mathbf{V}} = [\overline{\mathbf{V}}_1, \overline{\mathbf{V}}_2, \cdots, \overline{\mathbf{V}}_d]$. In the function library of PDEs, the term $\mathbf{u}$, $\mathbf{u}_x$, and $\mathbf{u}_{xx}$ are approximated by the state matrix $\overline{\mathbf{U}}$, the first-order and second-order partial derivatives matrices $\overline{\mathbf{V}}$ (order = 1) and $\overline{\mathbf{V}}$ (order = 2), respectively. The nonlinear term $\mathbf{u}\mathbf{u}_x$ is approximated by element-wise product of two matrices $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$ (order = 1). Hence, in equation (3), the derivatives matrix $\dot{\mathbf{U}}_i$ is computed by the posterior mean $\overline{\dot{\mathbf{U}}}_i$, and the function library matrix $\mathbf{\Phi}$ is constructed using $\overline{\mathbf{U}}$ and $\overline{\mathbf{V}}$, *i.e.*, $\mathbf{\Phi} = \mathbf{\Phi}(\overline{\mathbf{U}}, \overline{\mathbf{V}})$. As a result, we obtain the following linear equation to be solved:

$$\overline{\dot{\mathbf{U}}}_i = \mathbf{\Phi}\mathbf{C}_i + \boldsymbol{\varepsilon}_i, \tag{8}$$

where $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_i)$ accounts for the noise and embodies the uncertainty associated with the derivative data $\overline{\dot{\mathbf{U}}}_i$. We can rewrite this equation as a WLS problem given by,

$$\min_{\mathbf{C}_i} \left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right)^T \mathbf{W}_i \left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right). \tag{9}$$

Before computing the sparse solution of $\mathbf{C}_i$, the following lemma illustrates the WLS estimator of equation (9).

Lemma 1 *(Gauss-Markov Theorem [36]). For the least squares problem*

$$\min_{\mathbf{C}_i} \left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right)^T \left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right),$$

arising from Equation (8) *with* $\mathbf{\Sigma}_i = \sigma^2\mathbf{I}$*, the best linear unbiased estimator (BLUE) is given by* $\hat{\mathbf{C}}_i = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\overline{\dot{\mathbf{U}}}_i$.

Theorem 1. *For the weighted least squares problem* (9) *associated with Equation* (8)*, the optimal solution is given by*

$$\hat{\mathbf{C}}_i = (\mathbf{\Phi}^T\mathbf{W}_i\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{W}_i\overline{\dot{\mathbf{U}}}_i.$$

Moreover, $\hat{\mathbf{C}}_i$ *becomes the best linear unbiased estimator (BLUE) for the special case where* $\mathbf{W}_i = \mathbf{\Sigma}_i^{-1}$.

Proof. The optimality condition of (9) is given by

$$\frac{\partial}{\partial \mathbf{C}_i}\left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right)^T \mathbf{W}_i \left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right) = 2\mathbf{\Phi}^T\mathbf{W}_i\left(\overline{\dot{\mathbf{U}}}_i - \mathbf{\Phi}\mathbf{C}_i\right) = 0,$$

and we obtain the optimal solution

$$\hat{\mathbf{C}}_i = (\mathbf{\Phi}^T\mathbf{W}_i\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{W}_i\overline{\dot{\mathbf{U}}}_i.$$

Since $\mathbb{E}\left(\overline{\dot{\mathbf{U}}}_i\right) = \mathbb{E}\left(\boldsymbol{\Phi}\mathbf{C}_i + \boldsymbol{\varepsilon}_i\right) = \boldsymbol{\Phi}\mathbf{C}_i$, we have

$$\begin{aligned}\mathbb{E}\left(\hat{\mathbf{C}}_i\right) &= (\boldsymbol{\Phi}^T\mathbf{W}_i\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}_i\mathbb{E}\left(\overline{\dot{\mathbf{U}}}_i\right)\\ &= (\boldsymbol{\Phi}^T\mathbf{W}_i\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{W}_i\boldsymbol{\Phi}\mathbf{C}_i\\ &= \mathbf{C}_i,\end{aligned}$$

which indicates that $\hat{\mathbf{C}}_i$ is unbiased.

For the positive definite matrix $\boldsymbol{\Sigma}_i$, denote its Cholesky decomposition by $\boldsymbol{\Sigma}_i = \mathbf{L}\mathbf{L}^T$. Then, Equation (8) can be rewritten by,

$$\mathbf{L}^{-1}\overline{\dot{\mathbf{U}}}_i = \mathbf{L}^{-1}\boldsymbol{\Phi}\mathbf{C}_i + \mathbf{L}^{-1}\boldsymbol{\varepsilon}_i,$$

where $\mathbf{L}^{-1}\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This least square problem

$$\min_{\mathbf{C}_i}\left(\mathbf{L}^{-1}\overline{\dot{\mathbf{U}}}_i - \mathbf{L}^{-1}\boldsymbol{\Phi}\mathbf{C}_i\right)^T\left(\mathbf{L}^{-1}\overline{\dot{\mathbf{U}}}_i - \mathbf{L}^{-1}\boldsymbol{\Phi}\mathbf{C}_i\right),$$

is identical to Equation (9) when $\mathbf{W}_i = \boldsymbol{\Sigma}_i^{-1}$. According to Lemma 1, the corresponding BLUE is given by

$$\begin{aligned}\hat{\mathbf{C}}_i &= \left(\left(\mathbf{L}^{-1}\boldsymbol{\Phi}\right)^T\mathbf{L}^{-1}\boldsymbol{\Phi}\right)^{-1}\left(\mathbf{L}^{-1}\boldsymbol{\Phi}\right)^T\mathbf{L}^{-1}\overline{\dot{\mathbf{U}}}_i\\ &= (\boldsymbol{\Phi}^T(\mathbf{L}\mathbf{L}^T)^{-1}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T(\mathbf{L}\mathbf{L}^T)^{-1}\overline{\dot{\mathbf{U}}}_i\\ &= (\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_i^{-1}\overline{\dot{\mathbf{U}}}_i. \quad \square\end{aligned}$$

The Sequential Threshold Ridge regression (STRidge) algorithm introduced in [9,13] offers a powerful approach to seek the sparse solution of equation (9). Unlike other methods that relax the ℓ_0 regularization to ℓ_1 regularization, such as the least absolute shrinkage and selection operator (LASSO), STRidge enforces sparsity by applying thresholds iteratively. Based on STRidge, we propose the Sequential Threshold for Weighted Least Squares (STWLS) described by Algorithm 1, which incorporates the WLS estimator into the sparsity-promoting process. In order to keep the clarity of Algorithm 1, we replace $\overline{\dot{\mathbf{U}}}_i$ and $\mathbf{W}_i$ by the notations $\mathbf{Z}$ and $\mathbf{W}$, respectively.

Algorithm 1 Sequential Threshold for Weighted Least Squares (STWLS).

Input: $\boldsymbol{\Phi} \in \mathbb{R}^{N'\times N_f}$, $\mathbf{Z} \in \mathbb{R}^{N'}$, $\mathbf{W} \in \mathbb{R}^{N_f\times N_f}$, $\Lambda(\lambda \geq 0, \forall\lambda \in \Lambda)$, $\eta > 0$, K, J.

1: **Initialization:** $\tau = 0$, $\mathcal{I}_0 = \{1, \cdots, N_f\}$, $\mathbf{c}_0 = \boldsymbol{\Phi}^\dagger\mathbf{Z}$, $\mathcal{L}^* = (\boldsymbol{\Phi}\mathbf{c}_0 - \mathbf{Z})^T\mathbf{W}(\boldsymbol{\Phi}\mathbf{c}_0 - \mathbf{Z}) + \eta\|\mathbf{c}_0\|_0$, $\mathbf{c} = \mathbf{0}_{N_f}$
2: **for** $\lambda \in \Lambda$ **do**
3: **for** $k = 1$ to K **do**
4: $\mathcal{I} = \mathcal{I}_0$, $\widetilde{\boldsymbol{\Phi}} = \boldsymbol{\Phi}$, $\widetilde{\mathbf{W}} = \mathbf{W}$
5: **for** $j = 1$ to J **do**
6: $\mathbf{c}[\mathcal{I}] = (\widetilde{\boldsymbol{\Phi}}^T\widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Phi}} + \lambda\mathbf{I})^{-1}(\widetilde{\boldsymbol{\Phi}}^T\widetilde{\mathbf{W}}\mathbf{Z})$
7: $\mathcal{I} = \{r : |\mathbf{c}[r]| \geq \tau\}$, $\mathbf{c}[\mathcal{I}_0 \setminus \mathcal{I}] = 0$
8: $\widetilde{\boldsymbol{\Phi}} = \widetilde{\boldsymbol{\Phi}}[:, \mathcal{I}]$, $\widetilde{\mathbf{W}} = \widetilde{\mathbf{W}}[\mathcal{I}, \mathcal{I}]$
9: **end for**
10: $\mathbf{c}[\mathcal{I}] = (\widetilde{\boldsymbol{\Phi}}^T\widetilde{\mathbf{W}}\widetilde{\boldsymbol{\Phi}})^{-1}(\widetilde{\boldsymbol{\Phi}}^T\widetilde{\mathbf{W}}\mathbf{Z})$, $\mathbf{c}[\mathcal{I}_0 \setminus \mathcal{I}] = 0$
11: $\mathcal{L} = (\boldsymbol{\Phi}\mathbf{c} - \mathbf{Z})^T\mathbf{W}(\boldsymbol{\Phi}\mathbf{c} - \mathbf{Z}) + \eta\|\mathbf{c}\|_0$
12: **if** $\mathcal{L} \leq \mathcal{L}^*$ **then**
13: $\mathcal{L}^* = \mathcal{L}$, $\mathbf{c}^* = \mathbf{c}$
14: **end if**
15: $\tau = 1.05\min_i\{|c_i^*| : c_i^* \neq 0\}$
16: **end for**
17: **end for**

Output: Sparse solution $\mathbf{c}^*$.

In STWLS, Λ is the set that contains all candidate values of λ. For a fixed $\lambda \in \Lambda$, there are two loops, the outer loop is to determine the suitable threshold τ, while the inner loop seeks the sparse solution under the specified threshold iteratively. It is noteworthy that in STWLS, λ and η are two key parameters for sparse solutions. A series of ridge regression techniques are employed to identify the support function terms in the inner loop, wherein λ serves as the regularization parameter. Non-zero λ is employed to highlight the correct feature terms. Different from the standard ridge regression, λ in STWLS merely has impact on the support function terms (line 6 in Algorithm 1), which makes an indirect contribution to the final coefficient matrix. Another key parameter η is utilized to decide whether to accept the sparse solution from the inner loop or not, which makes the trade-off between the residual of linear equations and the sparsity of the solution. When η is smaller, the sparsity loss term is less penalized, which leads to the defectively sparse solution. Conversely, if it is too large, the solution exhibits an exceptionally high degree of sparsity since the sparsity loss term becomes dominant.

Remark 2. In the numerical experiments of Section 4, the final outcomes are not sensitive to K, J, and η. In practice, λ is typically difficult to choose, especially when the training data is limited, small λ would result in an excess of the function terms while larger values would yield much fewer terms. Analogous phenomenon also present in [21], where corner point criterion in Pareto curve is exploited to balance the fitting and sparsity of the solution. Here, we choose λ in a naive but effective manner by setting a candidate set Λ, where the total least square residual $\mathcal{L}$ reaches its minimum when the optimal λ is selected. This straightforward approach demonstrates its effectiveness in our experiments. We believe that more sophisticated methods, such as Bayesian optimization, would also be effective in finding the optimal λ.

Remark 3. In Algorithm 1, we address the WLS problem in the context of sparse identification of nonlinear systems, where the variance of noise is approximated by the posterior variance of temporal partial derivatives in GPR. Similar techniques can be found in related literatures, for example, the generalized least squares problem [25,37], wherein the covariance matrix (corresponding to the inverse of the aforementioned weight matrix) is approximated through the residual analysis of methods in [25,37].

Finally, for the WLS problem (9), the STWLS algorithm (Algorithm 1) is employed to derive the sparse solution $\mathbf{C}_i$. We leverage the posterior variance of the partial derivative w.r.t. time to approximate the covariance matrix of $\boldsymbol{\epsilon}_i$, *i.e.*, the weight matrix $\mathbf{W}_i = \left(\mathrm{diag}\left(\mathrm{var}(\dot{\mathbf{U}}_i)\right)\right)^{-1}$. Consequently, our Gaussian Process based SINDy (GP-SINDy) algorithm is summarized in Algorithm 2.

Algorithm 2 Gaussian Process based SINDy (GP-SINDy).

Input: Data $\mathcal{D} = \left\{\mathbf{X} \in \mathbb{R}^{D\times N}, \mathbf{U} \in \mathbb{R}^{N\times d}\right\}$, test input $\mathbf{X}_p \in \mathbb{R}^{D\times N'}$, parameters of STWLS Λ, η, K, J.

1: **for** $i \in \{1, \cdots, d\}$ **do**
2: Construct the GPR model with $\{\mathbf{X}, \mathbf{U}_i\}$, and learn the optimal hyperparameters, θ_i and σ_i^2.
3: Infer the posterior $\overline{\mathbf{U}}_i$, $\overline{\dot{\mathbf{U}}}_i$, $\mathrm{var}\left(\dot{\mathbf{U}}_i\right)$, and $\overline{\mathbf{V}}_i$ at test input $\mathbf{X}_p$.
4: Compute the weight matrix $\mathbf{W}_i = \left(\mathrm{diag}\left(\mathrm{var}(\dot{\mathbf{U}}_i)\right)\right)^{-1}$.
5: **end for**
6: Assemble the data matrices, $\overline{\mathbf{U}} = [\overline{\mathbf{U}}_1, \overline{\mathbf{U}}_2, \cdots, \overline{\mathbf{U}}_d]$, $\overline{\mathbf{V}} = [\overline{\mathbf{V}}_1, \overline{\mathbf{V}}_2, \cdots, \overline{\mathbf{V}}_d]$.
7: Compute the function library matrix $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\overline{\mathbf{U}}, \overline{\mathbf{V}})$.
8: **for** $i \in \{1, \cdots, d\}$ **do**
9: $\mathbf{C}[:, i]$ = STWLS($\boldsymbol{\Phi}$, $\overline{\dot{\mathbf{U}}}_i$, $\mathbf{W}_i$, Λ, η, K, J).
10: **end for**

Output: Sparse coefficient matrix $\mathbf{C}$.

Remark 4. In Algorithm 2, GP is utilized to mitigate the impact of noise in the observed data. It is noteworthy that the selection of the basis function library Φ can also introduce model errors, especially for some complicated differential equations. Given Φ, the discovered equation could be considered as an approximation of the "true equation".

3.3. MFGP-SINDy for multi-fidelity data

In this part, we will briefly review the nonlinear autoregressive multi-fidelity GP regression (MFGP) [29] to assimilate the nonlinear information from the MF data into the inference of the HF model. Let $\mathcal{D}^l = \left\{(\mathbf{x}_i^l, y_i^l)\middle|, i = 1, \cdots, N^l\right\}$ with $l = 1, \cdots, L$ denote the observed MF data, which satisfies,

$$y_i^l = f^l(\mathbf{x}_i^l) + \varepsilon_i^l,$$

where $\mathbf{x}_i^l \in \Omega^l \subset \mathbb{R}^D$ represents the input, $y_i^l \in \mathbb{R}$ is the noisy output of level l, and the noise term ε_i^l is i.i.d. at each level with Gaussian distribution $\mathcal{N}\left(0, \left(\sigma_{\mathrm{mf}}^l\right)^2\right)$. We rewrite the training data as $\mathcal{D}^l = \left\{(\mathbf{X}^l, \mathbf{y}^l)\right\}$ with $l = 1, \cdots, L$ where $\mathbf{X}^l = [\mathbf{x}_1^l, \mathbf{x}_2^l, \cdots, \mathbf{x}_{N^l}^l] \in \mathbb{R}^{D\times N^l}$ and $\mathbf{y}^l = [y_1^l, y_2^l, \cdots, y_{N^l}^l]^T \in \mathbb{R}^{N^l}$. Here, $\mathcal{D}^L$ corresponds to the HF data, and $\mathcal{D}^l$ for $(l = 1, \cdots, L-1)$ are the LF data sorted by increasing the level of accuracy. For the function f^1, it can be described by a Gaussian process, *i.e.*,

$$f^1(\mathbf{x}) \sim \mathcal{GP}\left(\overline{f^1}(\mathbf{x}), k^1\left(\mathbf{x}, \mathbf{x}'; \theta^1\right)\right),$$

where $k^1\left(\mathbf{x}, \mathbf{x}'; \theta^1\right)$ is an SE kernel. We use the notation $\mathrm{Cov}(\cdot, \cdot)$ to denote the covariance between two random variables. Assume that the $f^1, f^2, \cdots, f^L$ follow the Markov property,

$$\mathrm{Cov}\left(f^l(\mathbf{x}), f^{l-1}(\mathbf{x}')\middle| f^{l-1}(\mathbf{x})\right) = 0, \forall \mathbf{x} \neq \mathbf{x}', l = 2, \cdots, L,$$

which means that given the value of $f^{l-1}(\mathbf{x})$, we can learn nothing more about $f^l(\mathbf{x})$ from any other information $f^{l-1}(\mathbf{x}')$, for $\mathbf{x} \neq \mathbf{x}'$ [27,29]. Hence, the correlation between two levels of models, f^{l-1} and f^l, can be described by,

$$f^l(\mathbf{x}) = z^l(f^{l-1}(\mathbf{x})) + \delta^l(\mathbf{x}), \tag{10}$$

where $z^l : \mathbb{R} \to \mathbb{R}$ is an unknown nonlinear function and $\delta^l(\mathbf{x})$ is a Gaussian process independent of z^l. Specifically, when z^l is linear, Equation (10) degenerates to a linear autoregressive structure described in previous works [27,28].

However, Equation (10) involves a complicated problem that results in huge computational complexity. To address this issue, we can approximate z^l with a Gaussian process, which embeds the LF model $f^{l-1}(\mathbf{x})$ into the higher fidelity model $f^l(x)$. This is achieved by substituting $f^{l-1}(\mathbf{x})$ with its posterior estimation $\hat{f}^{l-1}(\mathbf{x})$. As a result, the correlation (10) can be transformed as:

$$f^l(\mathbf{x}) = g^l(\mathbf{x}, \hat{f}^{l-1}(\mathbf{x})),$$

where g^l is a nonlinear function, which maps from a $(D+1)$-dimensional subspace to a scalar in $\mathbb{R}$. It can be represented by a Gaussian process, and prior distribution of the l-th level function $f^l(\mathbf{x})$ is

$$f^l(\mathbf{x}) \sim \mathcal{GP}\left(\overline{f^l}(\mathbf{x}), k^l\left(\left(\mathbf{x}, \hat{f}^{l-1}(\mathbf{x})\right), \left(\mathbf{x}', \hat{f}^{l-1}(\mathbf{x}')\right); \theta^l\right)\right),$$

where $k^l\left(\left(\mathbf{x}, \hat{f}^{l-1}(\mathbf{x})\right), \left(\mathbf{x}', \hat{f}^{l-1}(\mathbf{x}')\right); \theta^l\right)$ is the corresponding kernel function and θ^l is its hyperparameter. Here, $\delta^l(\mathbf{x})$ can be considered implicitly due to the first input argument $\mathbf{x}$.

However, $\mathbf{x}$ and $\hat{f}^{l-1}(\mathbf{x})$ belong to different spaces, which pose a challenge when computing the partial derivative of the kernel w.r.t. $\mathbf{x}_j$ since $\hat{f}^{l-1}(\mathbf{x})$ is related to $\mathbf{x}$. For the sake of derivation of partial derivative of the kernel, we adopt the decomposition structure defined by Perdikaris et al. [29],

$$k^l\left(\left(\mathbf{x}, \hat{f}^{l-1}(\mathbf{x})\right), \left(\mathbf{x}', \hat{f}^{l-1}(\mathbf{x}')\right); \theta^l\right) = k_\rho(\mathbf{x}, \mathbf{x}'; \theta_\rho) k_f\left(\hat{f}^{l-1}(\mathbf{x}), \hat{f}^{l-1}(\mathbf{x}'); \theta_f\right) + k_\delta(\mathbf{x}, \mathbf{x}'; \theta_\delta), \tag{11}$$

where k_ρ, k_f, and k_δ are three SE kernels. In this paper, this kernel is referred to as the MFGP kernel. It bridges the gap between the input $\mathbf{x}$ and the estimated LF function values $\hat{f}^{l-1}(\mathbf{x})$ in a flexible form.

3.3.1. MFGP construction

In this paper, we focus on coping with the bi-fidelity data, *i.e.*, $L = 2$, wherein $f^1(\mathbf{x}) \sim \mathcal{GP}\left(\overline{f^1}(\mathbf{x}), k^1\left(\mathbf{x}, \mathbf{x}'; \theta^1\right)\right)$ and $f^2(\mathbf{x}) \sim \mathcal{GP}\left(\overline{f^2}(\mathbf{x}), k^2\left(\left(\mathbf{x}, \hat{f}^1(\mathbf{x})\right), \left(\mathbf{x}', \hat{f}^1(\mathbf{x}')\right); \theta^2\right)\right)$ represent the LF and HF GP models. We use the abbreviated notations $\mathcal{GP}^1$ and $\mathcal{GP}^2$ to represent them, respectively.

Before describing the computation of the MFGP, we analyze the structure of the MFGP model and make a reasonable assumption. Let $\mathbf{x}^* \in \mathbb{R}^D$ be a predictive point. Given that the LF model $f^1(\mathbf{x}^*)$ subject to a GP prior, its posterior distribution $\hat{f}^1(\mathbf{x}^*)$ remains to be GP, which enables the analytical computations of both the posterior mean and variance. Nevertheless, the HF model $f^2(\mathbf{x}^*)$ follows a GP prior, its posterior distribution do not follow a GP anymore due to the nonlinear mapping $f^2(\mathbf{x}^*)$, wherein the uncertainty no longer subject to a Gaussian distribution. The posterior distribution of the HF model $p\left(\hat{f}^2(\mathbf{x}^*)\right)$ can be approximated merely by a numerical approach [29], such as the Monte-Carlo integration,

$$\begin{aligned} p\left(\hat{f}^2(\mathbf{x}^*)\right) &= p\left(\hat{f}^2\left(\mathbf{x}^*, \hat{f}^1(\mathbf{x}^*)\right) \middle| \hat{f}^1(\mathbf{x}^*), \mathbf{x}^*, \mathbf{y}^2, \mathbf{x}^2\right) \\ &= \int p\left(\hat{f}^2\left(\mathbf{x}^*, \hat{f}^1(\mathbf{x}^*)\right) \middle|, \mathbf{x}^*, \mathbf{y}^2, \mathbf{x}^2\right) p\left(\hat{f}^1(\mathbf{x}^*)\right) \mathrm{d}\mathbf{x}^*, \end{aligned} \tag{12}$$

where the posterior distribution of the LF model $p\left(\hat{f}^1(\mathbf{x}^*)\right)$ is a Gaussian distribution, but the conditional probability $p\left(\hat{f}^2\left(\mathbf{x}^*, \hat{f}^1(\mathbf{x}^*)\right) \middle|, \mathbf{x}^*, \mathbf{y}^2, \mathbf{x}^2\right)$ is non-Gaussian.

However, such numerical computations result in unaffordable computational cost. To reduce the computational complexity, we assume that the posterior variance of the LF model $\hat{f}^1(\mathbf{x}^*)$ is sufficiently small to be ignored, thereby, it could be treated as a deterministic variable. In other words, $\hat{f}^1(\mathbf{x}^*)$ in HF model $\mathcal{GP}^2$ is replaced by its mean value $\overline{f^1}(\mathbf{x}^*)$. Subsequently, the posterior distribution of the HF model turns into as a Gaussian distribution and the posterior mean and variance could be computed analytically. Since we are focusing on the inference of the HF model, this assumption is reasonable and efficient in practical computations. Numerical experiments in Section 4.2.3 illustrate the fairness of this assumption.

Analogous to GPR, MFGP consists of two stages: training and inferring, and this bi-fidelity GP scheme can be readily extended to deal with even higher fidelity levels data. In the training stage, we firstly construct the $\mathcal{GP}^1$ model using the LF data $\mathcal{D}^1 = \{(\mathbf{x}^1, \mathbf{y}^1)\}$ and a SE kernel $k^1(\mathbf{x}, \mathbf{x}'; \theta^1)$. The hyperparameter θ^1 and noise variance $\left(\sigma_{\mathrm{mf}}^1\right)^2$ are obtained by minimizing the negative log marginal likelihood in equation (7). Then we compute the posterior mean denoted by $\overline{f^1}(\mathbf{x}^2)$ of $\mathcal{GP}^1$ at $\mathbf{x}^2$ using equation (5). Moreover, $\mathcal{GP}^2$ is devised based on the data $\left\{\left(\mathbf{x}^2, \overline{f^1}(\mathbf{x}^2)\right), \mathbf{y}^2\right\}$ and the kernel $k^2((\mathbf{x}, \overline{f^1}(\mathbf{x})), (\mathbf{x}', \overline{f^1}(\mathbf{x}')); \theta^2)$ defined in (11). The hyperparameter θ^2 and $\left(\sigma_{\mathrm{mf}}^2\right)^2$ is optimized by minimizing equation (7).

During the inference stage, we initially compute the posterior mean of $\mathcal{GP}^1$ at predictive input $\mathbf{x}^*$ by equation (5) denoted by $\overline{f^1}(\mathbf{x}^*)$. We obtain the posterior mean and variance of $\mathcal{GP}^2$ at $\left(\mathbf{x}^*, \overline{f^1}(\mathbf{x}^*)\right)$ represented by $\overline{f^2}(\mathbf{x}^*)$, $\mathrm{var}\left(f^2(\mathbf{x}^*)\right)$, respectively. Furthermore, we can also employ MFGP to infer the posterior mean and variance of partial derivatives of $\mathcal{GP}^2$ at $\left(\mathbf{x}^*, \overline{f^1}(\mathbf{x}^*)\right)$ by equation (6) denoted by $(\overline{\partial f^2}(\mathbf{x}^*))_j$, $\mathrm{var}\left((\overline{\partial f^2}(\mathbf{x}^*))_j\right)$. Here, the subscript j means the partial derivatives to the j-th element of $\mathbf{x}^*$,

so we tend to use the $\overline{\partial f^2}(\mathbf{x}^*)$ and var $\left(\overline{\partial f^2}(\mathbf{x}^*)\right)$ to abbreviate them. Details on partial derivatives of the MFGP kernel (11) are demonstrated in Appendix A.2.

The MFGP algorithm for the case of bi-fidelity data ($L = 2$) is summarized in Algorithm 3, where step 1 to 3 focus on learning the optimal hyperparameters in the kernel and the noise variance, while steps 4 to 6 are concerned with predictions for the state and its partial derivatives of the HF model.

Remark 5. As for the inference of partial derivatives, we can utilize the automatic differentiation techniques to track the gradient information during the process of state prediction, such as PyTorch [38]. This approach offers the advantage that we only need to conduct state prediction once, wherein the derivatives are obtained through automatic differentiation. However, it brings two drawbacks. One is that the posterior mean results of first-order partial derivatives is accurate, whereas accuracy diminishes for the second and higher-order derivatives due to the existence of noise. The other is that this method cannot offer an approximation to the posterior variance.

Algorithm 3 Multi-fidelity GP regression (MFGP).

Input: MF training data $\mathcal{D}^l = \{(\mathbf{X}^l, \mathbf{y}^l)\}, l = 1, 2$, predictive input $\mathbf{x}^*$, kernel functions $k^1(\mathbf{x}, \mathbf{x}'; \theta^1)$, and $k^2((\mathbf{x}, \overline{f^1}(\mathbf{x})), (\mathbf{x}', \overline{f^1}(\mathbf{x}')); \theta^2)$.

1: Construct the $\mathcal{GP}^1$ based on $\mathcal{D}^1$ and $k^1(\mathbf{x}, \mathbf{x}'; \theta^1)$, and learn θ^1 and $\left(\sigma^1_{\text{mf}}\right)^2$ by minimizing equation (7).
2: Compute the posterior mean $\overline{f^1}(\mathbf{x}^2)$ of $\mathcal{GP}^1$ at $\mathbf{x}^2$ using equation (5).
3: Construct $\mathcal{GP}^2$ based on $\left\{\left(\mathbf{x}^2, \overline{f^1}(\mathbf{x}^2)\right), \mathbf{y}^2\right\}$ and $k^2((\mathbf{x}, \overline{f^1}(\mathbf{x})), (\mathbf{x}', \overline{f^1}(\mathbf{x}')); \theta^2)$ in equation (11), and learn θ^2 and $\left(\sigma^2_{\text{mf}}\right)^2$ by minimizing equation (7).
4: Compute the posterior mean $\overline{f^1}(\mathbf{x}^*)$ of $\mathcal{GP}^1$ at $\mathbf{x}^*$ using equation (5).
5: Calculate the posterior mean $\overline{f^2}(\mathbf{x}^*)$ and variance var $\left(\overline{f^2}(\mathbf{x}^*)\right)$ of $\mathcal{GP}^2$ at $\left(\mathbf{x}^*, \overline{f^1}(\mathbf{x}^*)\right)$ using equation (5).
6: Calculate the posterior mean $\overline{\partial f^2}(\mathbf{x}^*)$ and variance of partial derivative var $\left(\overline{\partial f^2}(\mathbf{x}^*)\right)$ of $\mathcal{GP}^2$ at $\left(\mathbf{x}^*, \overline{f^1}(\mathbf{x}^*)\right)$ using equation (6).

Output: $\overline{f^2}(\mathbf{x}^*)$, var $\left(f^2(\mathbf{x}^*)\right)$, $\overline{\partial f^2}(\mathbf{x}^*)$, var $\left(\overline{\partial f^2}(\mathbf{x}^*)\right)$.

3.3.2. MFGP-SINDy

Similar with GP-SINDy, MFGP-SINDy also consists of three steps: constructing the MFGP model based on MF data, inferring the state and its partial derivatives, and computing a sparse solution of a WLS problem.

Firstly, MFGP is exploited as a surrogate model, which enables one to sample more points in the grid. Next, we use MFGP to infer the variables of interest. The MFGP kernel provides a sophisticated and powerful approach wherein the partial derivatives are calculated though the differentiation of the kernel function. This analytical manner is highly appropriate suitable for sparse and noisy observed data. Several local surrogate approaches can smooth the noisy data, but their parameters are difficult to select. Since the outcomes heavily depend on the choice of parameters, particularly when the noisy data is scarce. GP and MFGP have the merit that they do not need to select the parameters manually.

Finally, we obtain the discovered system via solving the WLS problem, which is accomplished by STWLS (Algorithm 1). Combined with GP-SINDy (Algorithm 2) and MFGP (Algorithm 3), the Multi-Fidelity Gaussian Process based SINDy (MFGP-SINDy) algorithm for sparse and noisy MF observed data is outlined in Algorithm 4. The number of HF observed data is less than that of LF due to fact that the cost of numerical simulation of HF model is more expensive. The size of HF data is a significant factor for the trade-off between computational cost and accuracy, which will be demonstrated in Section 4.

Algorithm 4 Multi-Fidelity Gaussian Process based SINDy (MFGP-SINDy).

Input: MF data $\mathcal{D}^l = \left\{\mathbf{X}^l \in \mathbb{R}^{D\times N_l}, \mathbf{U}^l \in \mathbb{R}^{N_l\times d}\right\}$ with $l = 1, 2$, test input $\mathbf{X}_p \in \mathbb{R}^{D\times N'}$, parameters of STWLS Λ, η, K, J.

1: **for** $i \in \{1, \cdots, d\}$ **do**
2: Construct the MFGP model using training data $\{\mathbf{X}^l, \mathbf{U}^l_i\}$ with $l = 1, 2$ and the kernel functions k^1, k^2, and learn the optimal hyperparameters θ^1_i, θ^2_i, $\left(\sigma^1_{\text{mf}}\right)^2_i$ and $\left(\sigma^2_{\text{mf}}\right)^2_i$ using steps 1 to 3 in Algorithm 3.
3: Infer the posterior $\overline{\mathbf{U}}_i$, $\overline{\dot{\mathbf{U}}}_i$, var $\left(\dot{\mathbf{U}}_i\right)$, and $\overline{\mathbf{V}}_i$ at test input $\mathbf{X}_p$ using steps 4 to 6 in Algorithm 3.
4: Compute the weight matrix $\mathbf{W}_i = \left(\text{diag}\left(\text{var}(\dot{\mathbf{U}}_i)\right)\right)^{-1}$.
5: **end for**
6: Assemble the data matrices, $\overline{\mathbf{U}} = [\overline{\mathbf{U}}_1, \overline{\mathbf{U}}_2, \cdots, \overline{\mathbf{U}}_d]$, $\overline{\mathbf{V}} = [\overline{\mathbf{V}}_1, \overline{\mathbf{V}}_2, \cdots, \overline{\mathbf{V}}_d]$.
7: Compute the function library matrix $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\overline{\mathbf{U}}, \overline{\mathbf{V}})$.
8: **for** $i \in \{1, \cdots, d\}$ **do**
9: $\mathbf{C}[:, i]$ = STWLS($\boldsymbol{\Phi}$, $\overline{\mathbf{U}}_i$, $\mathbf{W}_i$, Λ, η, K, J).
10: **end for**

Output: Sparse coefficient matrix $\mathbf{C}$.

The selection of the basis function library is a very typical question for SINDy. One intuitive approach is to integrate prior knowledge into the function library. Another commonly used function library takes a set of partial derivatives with respect to the spatial variables, such as u_x, u_{xx}, and their combination functions such as $u_x u_{xx}$. In this paper, we also follow this way. However, we

must point out that this method does not guarantee to obtain the true equations for all PDEs, especially for PDEs exhibiting complex dynamic behavior. Meanwhile, the number of basis functions depends on the highest order of the partial derivatives and the highest order of polynomials while both are manually specified. To the best of our knowledge, the selection of basis functions still remains challenging, as it is difficult to determine solely based on observation data. One possible strategy is to consider a basis function library that is as comprehensive as possible.

4. Numerical experiments

In this section, we conduct three numerical experiments to demonstrate the performance of GP-SINDy and MFGP-SINDy, including the Lorenz system, the Burgers' equation, the KdV equation, and a two-dimensional example. In the discovery of Lorenz system, we focus on the performance of GP-SINDy in handling SF observed data. The subsequent part involves the application of MFGP-SINDy to MF data scenarios, which contains the discovery of the Burgers' equation, the KdV equation, and a two-dimensional example. Our approach is compared with other alternative methods in the context of SF data. Meanwhile, the effect of MF training data sizes on the outcomes is explored.

In GP-SINDy or MFGP-SINDy, all hyperparameters in the kernel and the noise variance are initialized randomly, and they are learned by minimizing equation (7). This is implemented through the `Rprop` optimizer in PyTorch [38]. For the parameters in STWLS, we set $K = 20$, $J = 10$ for all experiments. We set $\Lambda = \{10^i, i = 0, -1, -2, \cdots, -5\}$ with $\eta = 5$ for the Lorenz system, and $\Lambda = \{i/2, i = 0, 1, \cdots, 10\}$ with $\eta = 150$ for the remaining experiments.

Prior to conducting the experiments, we outline the setups of experiments. The noise-free data of ODEs is generated utilizing the `ode45` function within MATLAB that employs absolute and relative tolerance set of 10^{-8}. For all PDEs experiments except Section 4.4, the noise-free data of HF model is generated using the `spin` class from the `Chebfun` library [23,39]. The observed data is obtained from the clean data through adding i.i.d. Gaussian white noise with zero mean and variance σ^2, where σ is determined by the noise ratio σ_{NR} and the clean data matrix $\mathbf{U}^*$,

$$\sigma = \sigma_{\mathrm{NR}} \frac{\|\mathbf{U}^*\|_F}{\sqrt{Nd}}.$$

Here the positive parameter σ_{NR} is manually specified, and $\|\mathbf{U}^*\|_F$ represents the Frobenius norm of the matrix $\mathbf{U}^* \in \mathbb{R}^{N\times d}$. This formulation indicates that the noise-to-signal ratio is approximately equivalent to σ [25], *i.e.*, $\sigma \approx \|\mathbf{U} - \mathbf{U}^*\|_F / \|\mathbf{U}^*\|_F$.

Let $\mathbf{C}^* \in \mathbb{R}^{N_f \times d}$ denote the true sparse coefficient matrix, we use three error metrics to evaluate the accuracy of $\mathbf{C}$ obtained by GP-SINDy and MFGP-SINDy algorithms. The first is the maximum error of the true non-zero coefficients,

$$E_\infty(\mathbf{C}) = \max_{\mathbf{C}^*_{i,j}\neq 0} \frac{|\mathbf{C}_{i,j} - \mathbf{C}^*_{i,j}|}{|\mathbf{C}^*_{i,j}|}.$$

The second is the relative ℓ_2 error between the discovered coefficients $\mathbf{C}$ and the true coefficients $\mathbf{C}^*$,

$$E_2(\mathbf{C}) = \frac{\|\mathbf{C} - \mathbf{C}^*\|_F}{\|\mathbf{C}^*\|_F},$$

And the last metric is the true positivity ratio (TPR) [22],

$$TPR(\mathbf{C}) = \frac{TP}{TP + FN + FP},$$

where TP represents the number of correctly discovered non-zero coefficients, FN represents the number of coefficients falsely discovered as zero, and FP represents the number of coefficients falsely discovered as non-zero. TPR measures the percentage of correctly identified function terms, and a TPR value of 1 indicates a perfect discovery of non-zero function terms.

4.1. Lorenz system

The Lorenz system is described as,

$$\begin{cases} \dot{x} = \beta_1(y - x), \\ \dot{y} = x(\beta_2 - z) - y, \\ \dot{z} = xy - \beta_3 z. \end{cases}$$

Here we choose $\beta_1 = 10, \beta_2 = 28, \beta_3 = -\frac{8}{3}$, and the time domain $t \in [0, 10]$, the initial value $[x_0, y_0, z_0] = [-8, 7, 27]$. The training data is generated with time-step $\Delta_t = 0.001$. During the process of sparse learning, polynomials up to the third-order are utilized as the function library, *i.e.*, $\mathbf{\Phi}(u) = \mathbf{\Phi}(x, y, z) = [1, x, y, z, x^2, xy, xz, y^2, yz, z^2, x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3]$. For GP-SINDy, the predictive input is the same with the input in the training data. SE kernel is adopted in this experiment.

We plot the ground truth and noisy observed data with a noise ratio $\sigma_{\mathrm{NR}} = 0.1$ (the exact standard deviation $\sigma = 1.59742$) in Fig. 1. Our GP-SINDy algorithm is applied to these observed data, where observed data is subsampled evenly with size of 626 during the training of hyperparameters in order to mitigate the computation afford of GPR. The results of hyperparameters and noise variance

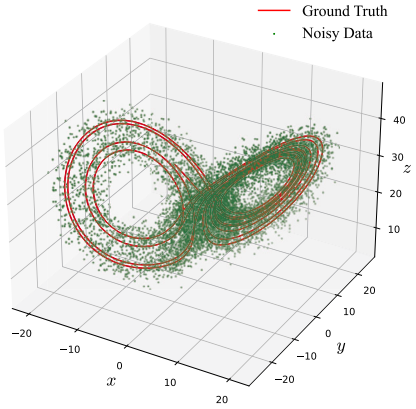

Fig. 1. The ground truth and noisy data with $\sigma_{\mathrm{NR}} = 0.1$.

Table 1
Optimal hyperparameters results of GPR in GP-SINDy. Here, x, y, and z are state variables, which correspond to three GP models.

	$(\theta_i)_0$	$(\theta_i)_1$	σ_i^2
$x(i=1)$	0.826	0.023	0.047
$y(i=2)$	0.865	0.014	0.034
$z(i=3)$	1.049	0.015	0.034

in GPR are shown in Table 1. The corresponding states prediction and derivatives prediction results with uncertainty approximation obtained from GPR are exhibited in Fig. 2 and 3, respectively. We observe that the posterior variance of the derivatives data are greater near the boundary points compared to the interior points, which attributes to the availability of information on only one side rather than both sides.

Next, we compare GP-SINDy with SINDy [9], Savitzky-Golay SINDy (SG-SINDy) [40], and Modified SINDy (M-SINDy) [41], wherein the same function library is exploited. In SG-SINDy, the Savitzky-Golay filter is exploited to fit a series of local polynomials for the training data, and the derivatives are calculated by the finite difference method. Here, we set the order of local polynomials to 3 and the length of window to 10. Besides, M-SINDy is another robust algorithm that combines the automatic differentiation and time-stepping constraints. The discovered results and coefficients error of these algorithms under $\sigma_{\mathrm{NR}} = 0.05, 0.1$ are summarized in Table 2 and the best results are listed in bold. The comparison of discovered dynamic systems are shown in Fig. 4. As for a local fitting model, the performance of SG-SINDy closely depends on the selection of parameters (the length of window). M-SINDy requires to solve an ill-posed optimization problem, since the degree of freedom of the optimized variables is more than the known data, and we observe that it is prone to overfit and fall into a local minimal. As pointed out by [41], it may produce a system that has a similar behavior to the actual system in given time interval, but the symbolic repression is extremely different from the exact solution.

4.2. Burgers' equation

Consider the 1D Burgers' equation,

$$u_t = \nu u_{xx} - u u_x, \tag{13}$$

with the initial condition $u(x,0) = -\sin(\pi x/8)$ and the periodic boundary condition $u(-8,t) = u(8,t)$, where $t \in [0,10]$, $x \in [-8,8]$. Here, the diffusion coefficient $\nu = 0.5$.

4.2.1. GP-SINDy for SF data

The SF data is generated by the fine spatiotemporal grid using the `spin` class from the `Chebfun` library. The time step size is $\Delta t = 0.002$ and the spatial grid size $\Delta x = 0.00625$. The training data is obtained by adding noise and downsampling of the clean data. Here, we downsample the data evenly with size n_s, which is referred to the size of SF data. For the sake of notation simplicity, we describe the training data using its size, instead of the downsampling step. For instance, n_s $(41 \times 65 = 2665)$ represents a uniform spatiotemporal grid with time step 41 and space size 65. The function library contains spatial partial derivatives up to the second-order and polynomials up to the second-order, *i.e.*, $\mathbf{\Phi}(u) = [1, u, u_x, u_{xx}, u^2, uu_x, uu_{xx}, u_x^2, u_x u_{xx}, u_{xx}^2]$. The predictive points are set as the spatiotemporal grid in $[0,10] \cup [-8,8]$ with $\Delta t' = 0.1$, $\Delta x' = 0.0625$ throughout this experiment.

First, we demonstrate the performance of GP-SINDy with SF data, here we use three SF data sizes, n_{s1} (41×65), n_{s2} (41×33), and n_{s3} (41×17). We show the identification results and error under various noise levels in Table 3, where σ represents the exact

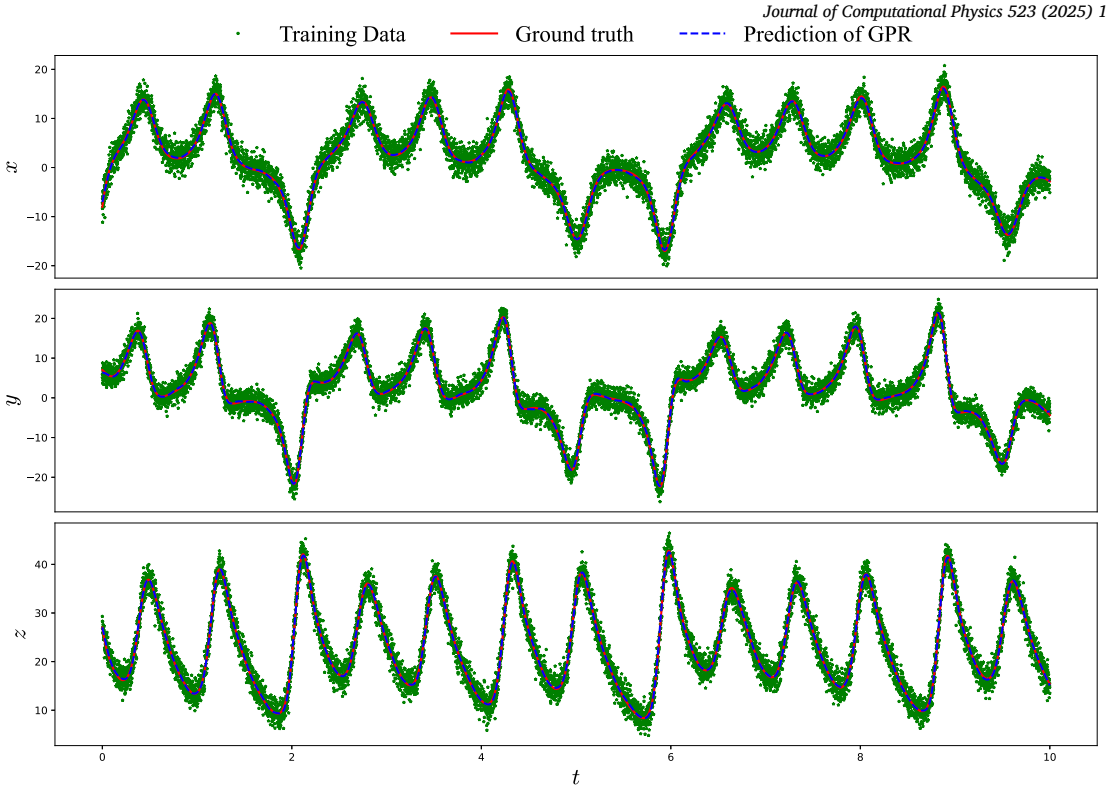

Fig. 2. Comparison between the ground truth, training data, and states prediction obtained from GPR.

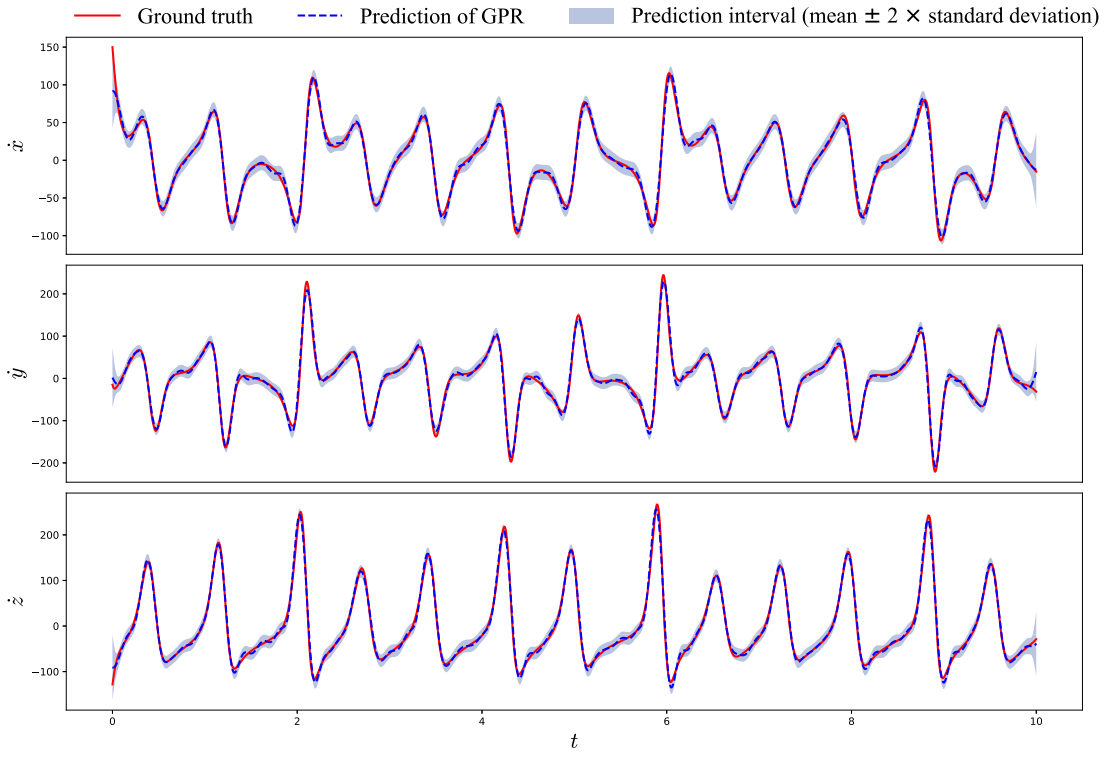

Fig. 3. The prediction and uncertainty approximation of derivatives.

Table 2
Discovered results and error under two noise levels ($\sigma_{\mathrm{NR}} \in \{0.05, 0.1\}$), where excessive results in SINDy are replaced by "…".

Method	Discovered equation ($\sigma_{\mathrm{NR}} = 0.05$)			$E_\infty(\%)$	$E_2(\%)$	TPR
SINDy	$\dot{x} = -0.814 + 4.078y - 0.265xz + 0.149yz$	$\dot{y} = -1.249 + 28.651x - 1.276y + 0.131z - 0.113x^2 + \cdots$	$\dot{z} = -1.761 - 0.358x + 0.3y - 2.58z + 0.998xy$	100.00	37.70	0.33
SG-SINDy	$\dot{x} = 0.284 - 9.83x + 9.746y$	$\dot{y} = 27.853x - 1.273y - 0.988xz$	$\dot{z} = 3.312 - 2.778z + 0.986xy$	27.34	10.64	0.78
M-SINDy	$\dot{x} = -9.908x + 9.914y$	$\dot{y} = 28.083x - 1.033y - 1.002xz$	$\dot{z} = -0.396 - 2.654z + 1.002xy$	3.30	1.35	0.88
GP-SINDy	$\dot{x} = -9.944x + 9.933y$	$\dot{y} = 27.916x - 0.991y - 0.999xz$	$\dot{z} = -2.663z + 0.999xy$	**0.93**	**0.39**	**1**
Method	Discovered equation ($\sigma_{\mathrm{NR}} = 0.1$)			$E_\infty(\%)$	$E_2(\%)$	TPR
SINDy	$\dot{x} = 56.124 + 9.489x - 0.815y - 5.446z - 0.502xz + \cdots$	$\dot{y} = 24.768x + 1.948y + 0.12z - 0.151x^2 + 0.345xy + \cdots$	$\dot{z} = -4.71 - 1.04x + 0.79y - 2.437z + 1.001xy$	294.76	193.44	0.35
SG-SINDy	$\dot{x} = 0.82 - 9.429x + 9.156y$	$\dot{y} = 2.955 + 26.468x - 1.609y - 0.114z - 0.935xz$	$\dot{z} = 0.176 + 0.28y - 2.568z + 0.687xy + 0.187y^2$	60.94	11.62	0.54
M-SINDy	$\dot{x} = -9.747x + 9.769y$	$\dot{y} = 28.165x - 1.062y - 1.004xz$	$\dot{z} = -0.965 - 2.636z + 1.007xy$	6.15	3.30	0.88
GP-SINDy	$\dot{x} = -9.881x + 9.861y$	$\dot{y} = 27.721x - 0.942y - 0.995xz$	$\dot{z} = -2.659z + 0.998xy$	**5.81**	**1.08**	**1**

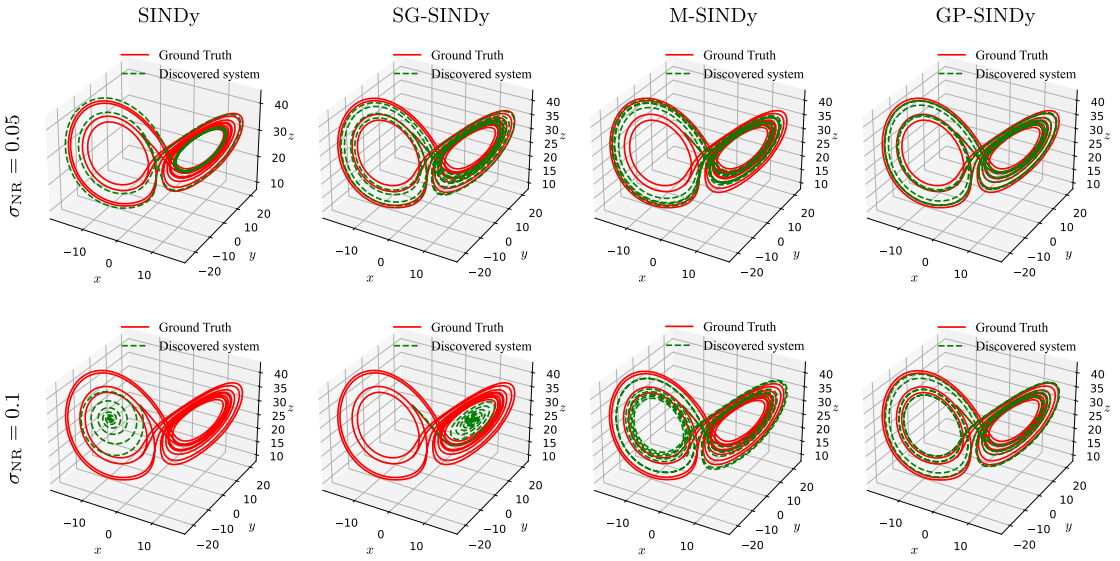

Fig. 4. The visualization of discovered dynamic systems, and these figures correspond to Table 2, where all initial conditions are set to [-8, 7, 27].

standard deviation of noise. We observe that for each SF data size n_s, the results deteriorate as the noise level get larger. Results in Table 3 show that GP-SINDy is capable of discovering the correct differential equation when the observation data is sparse and noisy. In the case of noise level $\sigma_{\mathrm{NR}} = 0.3$, although the corresponding coefficient errors E_∞ and E_2 are too large to accept, GP-SINDy could discover the correct function terms, *i.e.*, TPR is 1. Additionally, as the SF training data gets coarser from n_{s1} to n_{s2} to n_{s3}, the results get worse consistently at each specific noise level. The optimal hyperparameters of GPR are obtained by minimizing the negative log marginal likelihood, and we list the optimal values under different noise levels in Table 4.

Next, we compare GP-SINDy with other methods, including Tik in PDE-FIND [13], Robust IDENT (rIDENT) [14,42], and SG-Tik [13]. For Tik, the Tikhonov differentiation [13] is used for the derivatives computations. In rIDENT, we select the Savitzky-Golay filter as the smoother during the process of successively denoised differentiation (SDD) and subspace pursuit cross-validation (SC) for the sparse discovery of the differential equations. SG-Tik is the combination of Tik and rIDENT, and we use the Savitzky-Golay filter to smooth the noisy data and apply Tik to the smoothed data for the derivatives computations.

For Tik, rIDENT, and SG-Tik, the data used for sparse identification only focuses on the training data grid. However, GP-SINDy provides an alternative, which allows to infer the data on a finer grid and apply to discover the differential equations. To ensure the fairness and illustrate the smoothing effect instead of the interpolation, we add a control method called GP-SINDy* based on GP-SINDy, where these two methods differ only on the prediction points. Specifically, GP-SINDy uses the above default prediction points ($\Delta t' = 0.1$, $\Delta x' = 0.0625$), and GP-SINDy* infers the data only in the training data points. For sparse training data, the size of predictive points in GP-SINDy is larger than that of GP-SINDy*.

The error comparison (E_2) with three SF data sizes n_{s3} (41×81), n_{s4} (41×129), and n_{s5} (41×257) are listed in Table 5. We can see that GP-SINDy* outperforms Tik, rIDENT, and SG-Tik under these noise levels, and GP-SINDy gives slightly better results than GP-SINDy*. Results demonstrate that GP-SINDy has better denoising effect than other three methods, and it is capable to discover the

Table 3
Identification results and error of GP-SINDy with three SF data sizes.

SF data size	σ_{NR}	σ	Discovered equation	E_∞ (%)	E_2(%)	TPR
n_{s1}	0.02	0.009	$u_t = 0.495u_{xx} - 0.983uu_x$	1.69	1.57	1
	0.1	0.046	$u_t = 0.479u_{xx} - 0.933uu_x$	6.73	6.30	1
	0.2	0.093	$u_t = 0.469u_{xx} - 0.87uu_x$	12.96	11.92	1
	0.3	0.139	$u_t = 0.462u_{xx} - 0.822uu_x$	17.83	16.30	1
n_{s2}	0.02	0.009	$u_t = 0.492u_{xx} - 0.969uu_x$	3.08	2.85	1
	0.1	0.046	$u_t = 0.467u_{xx} - 0.89uu_x$	11.02	10.29	1
	0.2	0.093	$u_t = 0.445u_{xx} - 0.826uu_x$	17.40	16.33	1
	0.3	0.139	$u_t = 0.426u_{xx} - 0.775uu_x$	22.48	21.17	1
n_{s3}	0.02	0.009	$u_t = 0.488u_{xx} - 0.945uu_x$	5.51	5.03	1
	0.1	0.046	$u_t = 0.457u_{xx} - 0.843uu_x$	15.72	14.58	1
	0.2	0.093	$u_t = 0.446u_{xx} - 0.774uu_x$	22.61	20.79	1
	0.3	0.139	$u_t = 0.433u_{xx} - 0.694uu_x$	30.55	27.98	1

Table 4
Optimal hyperparameter results of GPR for data size n_{s1} under various noise levels, where $i = 1$.

σ_{NR}	σ	$(\theta_i)_0$	$(\theta_i)_1$	$(\theta_i)_2$	σ_i^2
0.02	0.009	0.493	10.112	2.542	0.0004
0.1	0.046	0.657	19.651	2.909	0.010
0.2	0.093	0.617	21.896	3.337	0.039
0.3	0.139	0.597	23.335	3.599	0.084

Table 5
Identification error $E_2(\%)$ of Tik, rIDENT, SG-Tik, GP-SINDy* and GP-SINDy with three SF data sizes, and we use "—" to represent the failed discovery and give the discovered function terms following.

σ_{NR}	Size	Tik (%)	rIDENT (%)	SG-Tik (%)	GP-SINDy* (%)	GP-SINDy (%)
0.02	n_{s3}	5.36	5.82	4.62	1.27	**1.24**
	n_{s4}	5.09	5.75	3.87	0.90	**0.89**
	n_{s5}	4.73	5.64	2.84	0.78	**0.78**
0.1	n_{s3}	— (uu_x, u_xu_{xx})	13.37	9.87	6.97	**6.89**
	n_{s4}	— (uu_x)	12.61	9.12	4.89	**4.83**
	n_{s5}	— (uu_x)	11.66	7.32	3.87	**3.84**
0.2	n_{s3}	— (uu_x)	16.50	10.91	10.68	**10.52**
	n_{s4}	— (uu_x)	21.46	18.38	11.35	**11.26**
	n_{s5}	— (u, uu_x, u_xu_{xx})	21.27	15.86	10.16	**10.10**

correct differential equations with sparse and noisy data. It is capable to infer the states or derivatives analytically at any point in the input domain with their uncertainty. Tik can discover the correct function terms when the noise level is low, but fails when the noise level is large since the data is sparse and highly corrupted. Notably, when the noise level is relatively high ($\sigma_{\text{NR}} = 0.2$), increasing the size of the dataset does not necessarily lead to more accurate results. In other words, when the noise level is large, using less training data may yield better outcomes. In rIDENT and SG-Tik, the smoothing parameters of the Savitzky-Golay filter (windows parameters) are difficult to choose, and they are closely related to the data sizes, which is a common issue for the local parameterized surrogate model. For different sizes of data, we need to adjust them manually. However, GP-SINDy offers an elegant manner that constructs a non-parametric model, which significantly increases the applicability.

4.2.2. MFGP-SINDy for MF data

In this part, we test the performance of MFGP-SINDy with MF data. The HF data is generated by the fine spatiotemporal grid using the `spin` class from the `Chebfun` library. The time step size is $\Delta t = 0.002$ and the spatial grid size $\Delta x = 0.00625$. The LF data is generated by the finite different method with coarse spatiotemporal grid $\Delta t_l = 0.25$, $\Delta x_l = 0.5$. We interpolate the LF data linearly on the fine grid to generate more LF data. The HF ground truth of equation (13) and LF data before interpolation are depicted in Fig. 5. In MFGP-SINDy, the LF and HF training data are corrupted by the noise with the same noise level σ_{NR}. We use n_h and n_l to denote the size of HF and LF data, respectively. Three SE kernels k_ρ, k_f, k_δ are utilized in the MFGP kernel (11).

We consider three evenly sampled HF training data sizes $n_{h1} = n_{s1}(41 \times 65)$, $n_{h2} = n_{s2}(41 \times 33)$, $n_{h3} = n_{s3}(41 \times 17)$ and LF training data size n_{l1} (51×81). Meanwhile, we investigate the result of GP-SINDy with data size n_{l1} as a comparison experiment, and the results are listed in Table 6. We can see that all results under these noise levels are not satisfactory. The discovered outcomes of MFGP-SINDy with three MF data sizes (n_{h1}, n_{l1}), (n_{h2}, n_{l1}), and (n_{h3}, n_{l1}) are listed in Table 7. Compared with the results in Table 3, the LF

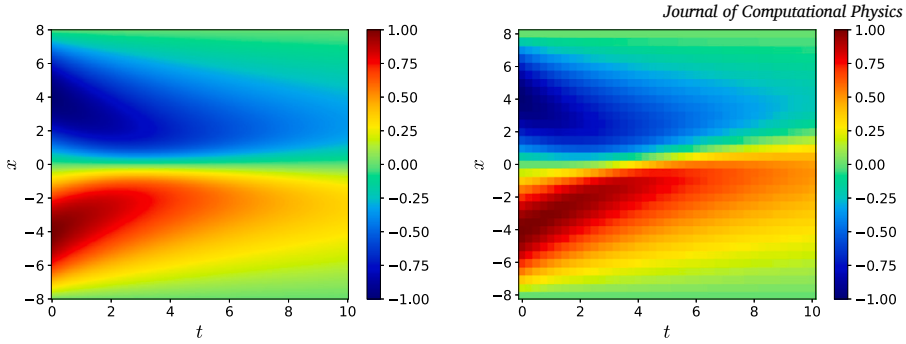

Fig. 5. The HF data and LF data (before interpolation) in Burgers' equation (13). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 6
Identification results and error of GP-SINDy with data size n_{l1}.

σ_{NR}	Discovered equation	E_∞ (%)	E_2(%)	TPR
0.02	$u_t = 0.478u_{xx} - 1.007uu_x - 0.246uu_{xx}$	4.45	22.13	0.67
0.1	$u_t = 0.44u_{xx} - 0.978uu_x + 0.231u_{xx}^2$	12.06	21.44	0.67
0.2	$u_t = 0.307u_{xx} - 1.204uu_x - 0.692u_xu_{xx}$	38.53	66.76	0.67
0.3	$u_t = -1.224uu_x - 1.572u_xu_{xx}$	100.00	148.87	0.33

Table 7
Identification results and error of MFGP-SINDy with three MF data sizes.

MF data size	σ_{NR}	Discovered equation	E_∞ (%)	E_2(%)	TPR
(n_{h1}, n_{l1})	0.02	$u_t = 0.495u_{xx} - 0.995uu_x$	1.04	0.67	1
	0.1	$u_t = 0.475u_{xx} - 0.966uu_x$	4.95	3.75	1
	0.2	$u_t = 0.465u_{xx} - 0.933uu_x$	7.05	6.75	1
	0.3	$u_t = 0.461u_{xx} - 0.912uu_x$	8.81	8.63	1
(n_{h2}, n_{l1})	0.02	$u_t = 0.493u_{xx} - 0.99uu_x$	1.46	1.14	1
	0.1	$u_t = 0.458u_{xx} - 0.903uu_x$	9.68	9.45	1
	0.2	$u_t = 0.456u_{xx} - 0.89uu_x$	11.04	10.62	1
	0.3	$u_t = 0.465u_{xx} - 0.863uu_x$	13.70	12.65	1
(n_{h3}, n_{l1})	0.02	$u_t = 0.478u_{xx} - 0.98uu_x$	4.33	2.61	1
	0.1	$u_t = 0.46u_{xx} - 0.9uu_x$	10.04	9.66	1
	0.2	$u_t = 0.449u_{xx} - 0.852uu_x$	14.77	13.99	1
	0.3	$u_t = 0.43u_{xx} - 0.77uu_x$	23.01	21.51	1

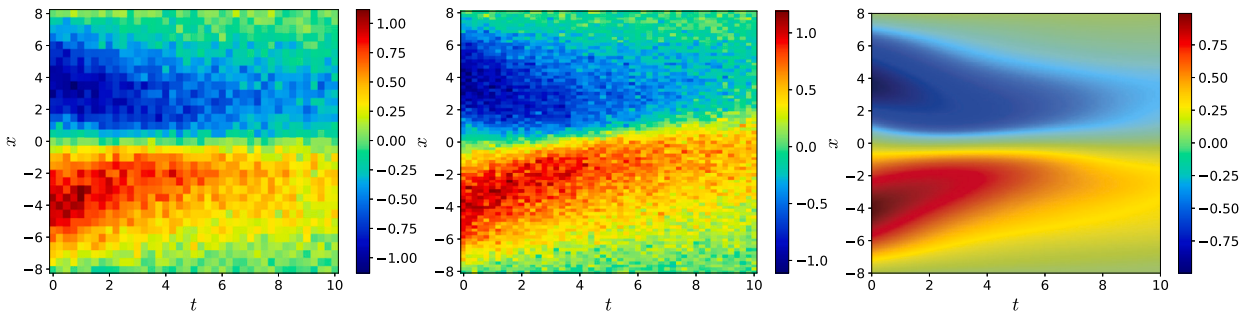

Fig. 6. The HF and LF noisy observation data of sizes (n_{h1}, n_{l1}), and prediction result of MFGP under $\sigma_{\text{NR}} = 0.2$.

data improves the accuracy of outcomes dramatically, especially when the noise levels are 0.2, 0.3. This indicates the robustness of MFGP-SINDy to large noise levels.

Fig. 6 illustrates the HF and LF noisy observation data with sizes (n_{h1}, n_{l1}), and prediction results of MFGP under noise level $\sigma_{\text{NR}} = 0.2$. The corresponding optimal hyperparameters in MFGP are listed in Table 8.

Table 8
Optimal hyperparameters in MFGP, where θ^l and $(\sigma^l)^2$ are hyperparameters in the GP model level-l ($l = 1, 2$), with the MFGP kernel $k^2(\theta^2) = k_\rho(\theta_\rho)k_f(\theta_f) + k_\delta(\theta_\delta)$. Here, large $(\theta_\delta)_1$ implies that the corresponding input (spatial coordinate) has little contribution.

$(\theta^1)_0$	$(\theta^1)_1$	$(\theta^1)_2$	$(\theta_\rho)_0$	$(\theta_\rho)_1$	$(\theta_\rho)_2$	
0.554	14.745	2.580	0.735	65.972	5.607	
$(\theta_f)_0$	$(\theta_f)_1$	$(\theta_\delta)_0$	$(\theta_\delta)_1$	$(\theta_\delta)_2$	$(\sigma^1_{\text{mf}})^2$	$(\sigma^2_{\text{mf}})^2$
0.746	0.664	0.488	4.581×10^{25}	13.561	0.039	0.037

Table 9
Identification error $E_2(\%)$ and $E_\infty(\%)$ of MFGP-SINDy*, MFGP-SINDy(MC) and MFGP-SINDy with three MF data sizes.

MF data size	MFGP-SINDy*		MFGP-SINDy(MC)		MFGP-SINDy	
	$E_\infty(\%)$	$E_2(\%)$	$E_\infty(\%)$	$E_2(\%)$	$E_\infty(\%)$	$E_2(\%)$
(n_{h1}, n_{l1})	5.22	4.33	2.94	1.99	**2.88**	**1.94**
(n_{h2}, n_{l1})	6.69	6.64	3.23	2.79	**3.19**	**2.75**
(n_{h3}, n_{l1})	9.52	8.96	5.51	5.25	**5.49**	**5.23**

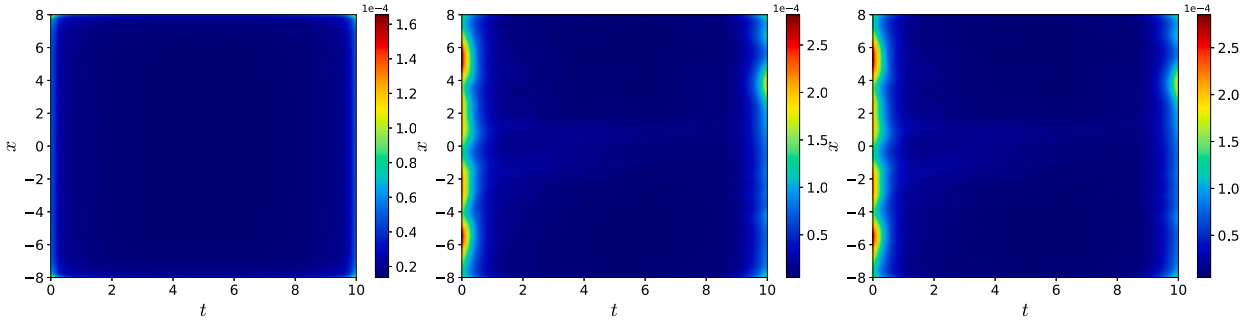

Fig. 7. The visualization of LF posterior variance of u, HF posterior variance of u_t in MFGP-SINDy and MFGP-SINDy(MC).

4.2.3. *Test for posterior variance of MF prediction*

To compute the posterior distribution of the HF model prediction, one can apply the Monte-Carlo methods for direct computations (cf. Equation (12)). However, such operations suffer high computational complexity. To simplify computations, we assume that the posterior variance of the LF model is sufficiently small to be ignored in Section 3.3. In this experiment, we evaluate the fairness of our assumption.

We denote the method which applies the Monte-Carlo methods for direct computations by MFGP-SINDy (MC). Besides, we introduce a control method named MFGP-SINDy*, where the posterior variance in MFGP-SINDy* is ignored, *i.e.*, the weight matrix in MFGP-SINDy* is an identity matrix. Under our assumption, MFGP-SINDy allows one to compute the posterior distribution of the HF model analytically. In contrast, the posterior mean and variance are obtained via a group of samples for MFGP-SINDy (MC). Denote the number of sample points in MFGP-SINDy (MC) by n_{sam}. Since each sample point corresponds one standard GP implementation, the runtime of the prediction step for the HF model in MFGP-SINDy (MC) is approximately n_{sam} times that of MFGP-SINDy.

Firstly, the error comparison of identified outcomes of MFGP-SINDy*, MFGP-SINDy (MC), and MFGP-SINDy under noise level $\sigma_{\text{NR}} = 0.05$ are listed in Table 9, where we set $n_{sam} = 50$ for MFGP-SINDy (MC). We observe that the uncertainty information (posterior variance) of the HF model can enhance the accuracy of discovered results compared with the performance of MFGP-SINDy*. Moreover, the outcomes of MFGP-SINDy (MC) and MFGP-SINDy are very close, and MFGP-SINDy is slightly better than MFGP-SINDy (MC).

Fig. 7 depicts the LF posterior variance of u and HF posterior variance of u_t in MFGP-SINDy and MFGP-SINDy (MC) where the MF data size is set as (n_{h1}, n_{l1}). We can conclude that the variance at most points over the spatiotemporal grids is concentrated between 0 and 0.4×10^{-4}, indicating a small overall variation.

4.2.4. *Randomly sampled data*

In the previous part, we only consider the uniform grid data. Given that the MF data size plays an essential role, we focus on testing the performance of MFGP-SINDy using randomly sampled observation data in this part. To explore the impact of HF training data size on the outcome, we fix the size of LF data to n_{l1} (51×81) and randomly select n_h points from the fine spatiotemporal grid as the HF training data. Meanwhile, the results of GP-SINDy are used for comparison.

Fig. 8 illustrates E_∞, E_2, and TPR under varying HF training data sizes from 100 to 1000 when the noise level $\sigma_{\text{NR}} \in \{0.1, 0.2, 0.3\}$, where the results are computed on the average over 50 runs. It can be observed that for each noise level, both GP-SINDy and MFGP-

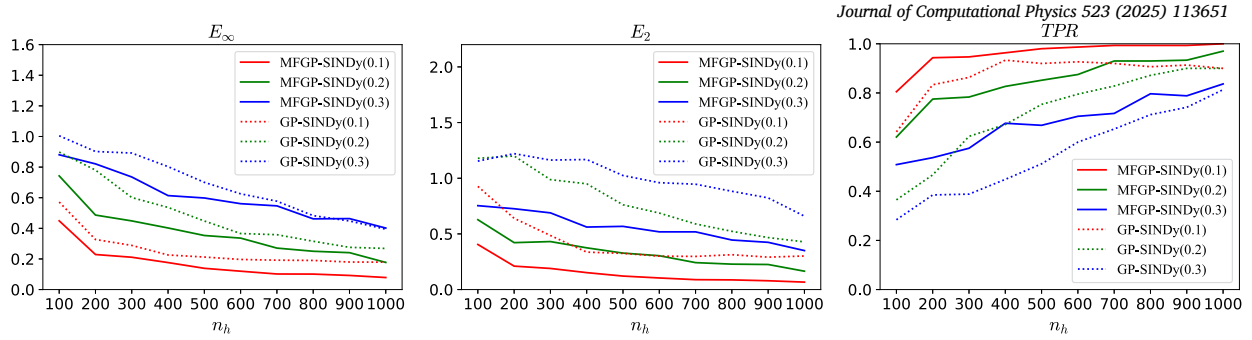

Fig. 8. The comparison of GP-SINDy and MFGP-SINDy with varying randomly sampled HF data sizes.

Table 10
Discovered results and error of MFGP-SINDy.

MF data size	σ_{NR}	Discovered equation	E_∞ (%)	E_2(%)	TPR
(n_{h2}, n_{l2})	0.02	$u_t = 0.496u_{xx} - 0.985uu_x$	1.55	1.43	1
	0.1	$u_t = 0.479u_{xx} - 0.92uu_x$	8.03	7.42	1
	0.2	$u_t = 0.455u_{xx} - 0.861uu_x$	13.94	13.11	1
	0.3	$u_t = 0.431u_{xx} - 0.801uu_x$	19.86	18.80	1

SINDy have better outcome with the increasing HF data size n_h. Significantly, for noise levels 0.2 and 0.3, the E_∞ (the maximum error in the true non-zero coefficients) of GP-SINDy is close to MFGP-SINDy, but E_2 and TPR are worse. When the data is abundant, GP-SINDy yields accurate results. The incorporation of MF data in MFGP-SINDy enables effective information fusion, thereby yields satisfactory results with lower computational cost compared with large high resolution SF data.

4.2.5. *Another multi-fidelity structure*

In this part, we focus on another MF structure, where the only difference with Section 4.2.2 is the LF model which is obtained by neglecting the nonlinear term in the HF model, such LF model also appears in [31]. The HF and LF models are described by

$$\begin{cases} (u_h)_t = \nu(u_h)_{xx} - u_h(u_h)_x, \\ (u_l)_t = \nu(u_l)_{xx}. \end{cases} \tag{14}$$

Here, the "clean" LF data is generated by solving the LF model via the `spin` class from the `Chebfun` library [39] with the same spatiotemporal grid $\Delta t_l = 0.25$, $\Delta x_l = 0.5$. When we set the training data sizes n_{h2} (41×33), n_{l2} (41×33), the identification results under different σ_{NR} are listed in Table 10. The LF model in (14) has different evolution with the HF model, which makes LF data inaccurate, although its effects is not as strong as the coarse grid data in Table 7. MFGP-SINDy has the ability to identify the correct function terms using MF data.

4.3. *KdV equation*

Our third example is the Korteweg-de Vries (KdV) equation [23], which is given by,

$$u_t = -u_{xxx} - uu_x, \tag{15}$$

with the initial condition $u(x,0) = \exp(-\pi(x/30)^2)\cos(\pi x/10)$ and the periodic boundary condition $u(-20,t) = u(20,t)$, where $t \in [0,40]$, $x \in [-20,20]$. Similar with the Burgers' equation, we generate the HF and LF data by the `spin` class from the `Chebfun` library with fine grid $\Delta t = 0.002$, $\Delta x = 0.015625$ and coarse grid $\Delta t_l = 0.1$, $\Delta x_l = 1.25$, respectively. The LF data is interpolated linearly on the grid $\widetilde{\Delta t_l} = \Delta t_l$, $\widetilde{\Delta x_l} = \Delta x_l/8$ in order to generate more data. Fig. 9 shows the ground truth (HF data) and LF data before interpolation. The function library contains spatial partial derivatives up to the third-order and polynomials up to the second-order, *i.e.*, $\boldsymbol{\Phi}(u) = [1, u, u_x, u_{xx}, u_{xxx}, u^2, uu_x, uu_{xx}, uu_{xxx}, u_x^2, u_xu_{xx}, u_xu_{xxx}, u_{xx}^2, u_{xx}u_{xxx}, u_{xxx}^2]$. The predictive points are set as the spatiotemporal grid in $[0,40] \cup [-20,20]$ with $\Delta_t' = 0.4$, $\Delta_x' = 0.15625$.

The SF training data size is set to n_s (41×81) and MF data size is set to n_h (41×81), n_l (41×129). Here, we test the performance of Tik, rIDENT, SG-Tik, GP-SINDy with SF data size n_s, and MFGP-SINDy with MF data size (n_h, n_l) under three noise levels $\sigma_{\text{NR}} \in \{0.03, 0.05, 0.1\}$. Table 11 shows the symbolic discovered results of GP-SINDy and MFGP-SINDy. The identification error (E_2) of Tik, rIDENT, SG-Tik, GP-SINDy, and MFGP-SINDy are listed in Table 12. rIDENT has better outcomes than SG-Tik since rIDENT smooths the data repeatedly rather than once in SG-Tik. Overall, MFGP-SINDy performs best to capture the accurate system. Meanwhile, the LF training data contributes to the final symbolic results, especially when the noise level is relatively high ($\sigma_{\text{NR}} = 0.1$).

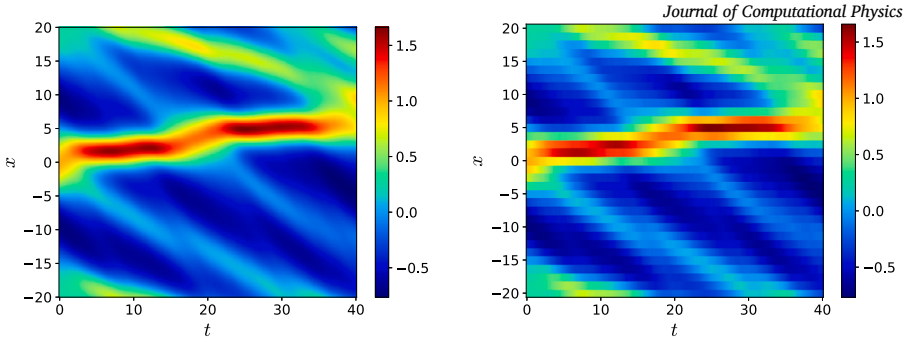

Fig. 9. The HF data and LF data (before interpolation) in KdV equation (15).

Table 11
Identification results of GP-SINDy with SF data size n_s and MFGP-SINDy with MF data size (n_h, n_l).

σ_{NR}	Discovered equation (GP-SINDy)	Discovered equation (MFGP-SINDy)
0.03	$u_t = -0.882u_{xxx} - 0.875uu_x$	$u_t = -0.901u_{xxx} - 0.894uu_x$
0.05	$u_t = -0.824u_{xxx} - 0.813uu_x$	$u_t = -0.86u_{xxx} - 0.853uu_x$
0.1	$u_t = -0.699u_{xxx} - 0.685uu_x$	$u_t = -0.804u_{xxx} - 0.804uu_x$

Table 12
Identification error $E_2(\%)$ of Tik, rIDENT, SG-Tik, and GP-SINDy with SF data size n_s and MFGP-SINDy with MF data size (n_h, n_l), and we use "—" to represent the failed discovery and give the discovered function terms.

σ_{NR}	Tik (%)	rIDENT (%)	SG-Tik (%)	GP-SINDy (%)	MFGP-SINDy (%)
0.03	— (uu_x)	17.12	43.76	12.12	**10.27**
0.05	— $(u_x, uu_x, u_x u_{xx})$	28.01	48.21	18.17	**14.36**
0.1	— (u_x, uu_x)	43.45	54.13	30.80	**19.63**

4.4. Two-dimensional PDE

Our final example is a two-dimensional PDE problem [14], which is given by,

$$u_t = \nu u_{xx} - uu_y, \tag{16}$$

with the initial condition $u(x, y, 0) = \sin^2(\pi x/8)\cos^2(\pi y/16)$ and the periodic Dirichlet boundary condition. Here, $t \in [0, 1.0]$, $(x, y) \in [-8, 8]^2$, and $\nu = 0.5$. Both HF and LF data are generated by finite different method with fine grid $\Delta t = 0.001$, $\Delta x = \Delta y = 0.0625$ and coarse grid $\Delta t_l = 0.01$, $\Delta x_l = \Delta y_l = 1.0$, respectively. Similar to the settings in Section 4.2.2, the LF data is interpolated linearly over the grid $\widetilde{\Delta t_l} = \Delta t_l/10$, $\widetilde{\Delta x_l} = \Delta x_l/16$, $\widetilde{\Delta y_l} = \Delta y_l/16$. Fig. 10 illustrates the ground truth (HF data) and LF data before interpolation at $t = 0$. The function library contains spatial partial derivatives up to the second-order and polynomials up to the second-order, *i.e.*, $\boldsymbol{\Phi}(u)$ consists of $1, u, u_x, u_{xx}, u_y, u_{yy}, u_{xy}$ and their combination functions, which results in 28 basis functions in $\boldsymbol{\Phi}(u)$. The predictive points are defined over the spatiotemporal grid in $[0, 1.0] \cup [-8, 8]^2$ with $\Delta_t' = 0.02$, $\Delta_x' = \Delta_y' = 0.25$.

The SF training data size is set to n_s $(26 \times 9 \times 9)$ and the MF data size is set to n_h $(26 \times 9 \times 9)$, n_l $(26 \times 17 \times 17)$. Here the data size $(26 \times 9 \times 9)$ represents a uniform spatiotemporal grid with 26 time steps and a spatial grid of size 9 in both the x and y dimensions. Here, we evaluate the performance of Tik, rIDENT, SG-Tik, GP-SINDy with data size n_s, and MFGP-SINDy with MF data size (n_h, n_l) under three noise levels $\sigma_{\text{NR}} \in \{0.01, 0.05, 0.1\}$. Table 13 gives the symbolic results of identified PDEs by GP-SINDy and MFGP-SINDy. The identification errors (E_2 or TPR) for Tik, rIDENT, SG-Tik, GP-SINDy, and MFGP-SINDy are listed in Table 14. We observe that Tik, rIDENT, and SG-Tik fail to discover the correct function terms due to the limited amount of training data (only $26 \times 9 \times 9 = 2106$ data points) and inaccurate approximation of partial derivatives. However, GP-SINDy and MFGP-SINDy remain effective for sparse noisy data. Furthermore, the integration of LF data dramatically reduces the identification error.

5. Conclusion

In this paper, we propose two robust algorithms GP-SINDy and MFGP-SINDy for effective sparse discovery of differential equations. GP-SINDy and MFGP-SINDy are designed for coping with single-fidelity and multi-fidelity observed data, respectively. Both of them are based on Gaussian process regression which eliminate the effect of noise and provide the uncertainty quantification for the inference variables. We compute the variance of time derivatives by their posterior variance in GPR, which is embodied in the

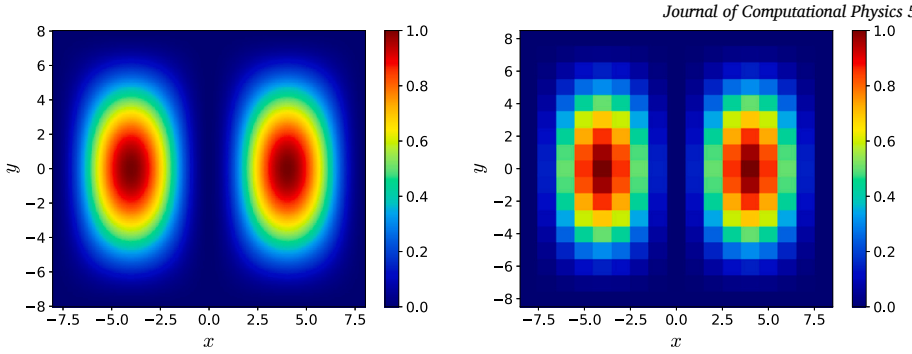

Fig. 10. The HF data and LF data (before interpolation) at $t = 0$ in two-dimensional PDE (16).

Table 13
Identification results of GP-SINDy with SF data size n_s and MFGP-SINDy with MF data size (n_h, n_l).

σ_{NR}	Discovered equation (GP-SINDy)	Discovered equation (MFGP-SINDy)
0.01	$u_t = 0.516u_{xx} - 0.994uu_y$	$u_t = 0.506u_{xx} - 1.0uu_y$
0.05	$u_t = 0.496u_{xx} - 0.924uu_y$	$u_t = 0.493u_{xx} - 1.008uu_y$
0.1	$u_t = 0.465u_{xx} - 0.825uu_y$	$u_t = 0.486u_{xx} - 1.013uu_y$

Table 14
Identification error $E_2(\%)$ of Tik, rIDENT, SG-Tik, and GP-SINDy with SF data size n_s and MFGP-SINDy with MF data size (n_h, n_l), and we use "—" to represent the failed discovery and give the TPR value.

σ_{NR}	Tik, rIDENT, SG-Tik (%)	GP-SINDy (%)	MFGP-SINDy (%)
0.01	— (0.50)	1.50	**0.52**
0.05	— (0.50)	6.77	**0.96**
0.1	— (0.50)	15.94	**1.72**

weighted least-squares to improve the discovery outcomes. MFGP-SINDy enables to use less amount of high-fidelity data to obtain satisfactory results, which reduces the computational cost.

CRediT authorship contribution statement

Yuhuang Meng: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation. **Yue Qiu:** Writing – original draft, Writing – review & editing, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

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.

Acknowledgements

We would like to thank Yuchen He and Sung Ha Kang for sharing their code. We also would like to thank the anonymous referees that help to improve the quality of this paper.

Appendix A. Computations of the partial derivatives of the kernel functions

A.1. Partial derivatives of the SE kernel

Before the derivation of the partial derivatives of the MFGP kernel, we first give some useful formulas in this part. For an SE kernel function $k_{\text{SE}}\left(\mathbf{x}, \mathbf{x}'; \theta\right) = \theta_0^2 \exp\left(-\sum_{s=1}^{D} \frac{\left(\mathbf{x}_s - \mathbf{x}'_s\right)^2}{2\theta_s^2}\right)$, its first-order partial derivative w.r.t. $\mathbf{x}'_j$ (the j-th elements of $\mathbf{x}'$) is

$$\frac{\partial k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial \mathbf{x}'_j}=\theta_0^2\exp\left(-\sum_{s=1}^{D}\frac{\left(\mathbf{x}_s-\mathbf{x}'_s\right)^2}{2\theta_s^2}\right)\frac{\mathbf{x}_j-\mathbf{x}'_j}{\theta_j^2}, \tag{A.1}$$

and its first-order partial derivative w.r.t. $\mathbf{x}_i$ is given by,

$$\frac{\partial k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial \mathbf{x}_i}=-\theta_0^2\exp\left(-\sum_{s=1}^{D}\frac{\left(\mathbf{x}_s-\mathbf{x}'_s\right)^2}{2\theta_s^2}\right)\frac{\mathbf{x}_i-\mathbf{x}'_i}{\theta_i^2}=-\frac{\partial k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial \mathbf{x}'_i}.$$

Its second-order partial derivative w.r.t. $\mathbf{x}'_j$ and $\mathbf{x}_i$ are

$$\frac{\partial^2 k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial(\mathbf{x}'_j)^2}=\theta_0^2\exp\left(-\sum_{s=1}^{D}\frac{\left(\mathbf{x}_s-\mathbf{x}'_s\right)^2}{2\theta_s^2}\right)\frac{1}{\theta_j^2}\left(\frac{\left(\mathbf{x}_j-\mathbf{x}'_j\right)^2}{\theta_j^2}-1\right), \tag{A.2}$$

and

$$\frac{\partial^2 k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial(\mathbf{x}_i)^2}=\frac{\partial^2 k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial(\mathbf{x}'_i)^2}.$$

Meanwhile, its second-order cross partial derivative is

$$\frac{\partial^2 k_{\mathrm{SE}}\left(\mathbf{x},\mathbf{x}';\theta\right)}{\partial \mathbf{x}_i\partial \mathbf{x}'_j}=\begin{cases}-\theta_0^2\exp\left(-\sum_{s=1}^{D}\frac{\left(\mathbf{x}_s-\mathbf{x}'_s\right)^2}{2\theta_s^2}\right)\frac{\mathbf{x}_i-\mathbf{x}'_i}{\theta_i^2}\frac{\mathbf{x}_j-\mathbf{x}'_j}{\theta_j^2}, & i\neq j,\\ \theta_0^2\exp\left(-\sum_{s=1}^{D}\frac{\left(\mathbf{x}_s-\mathbf{x}'_s\right)^2}{2\theta_s^2}\right)\frac{1}{\theta_j^2}\left(1-\frac{\left(\mathbf{x}_j-\mathbf{x}'_j\right)^2}{\theta_j^2}\right)=-\frac{\partial^2 k_{\mathrm{SE}}(\mathbf{x},\mathbf{x}';\theta)}{\partial(\mathbf{x}'_j)^2}, & i=j.\end{cases}$$

A.2. Partial derivatives of the MFGP kernel

The MFGP kernel is given by,

$$k^l\left(\left(\mathbf{x},\hat{f}^{l-1}(\mathbf{x})\right),\left(\mathbf{x}',\hat{f}^{l-1}(\mathbf{x}')\right);\theta^l\right)=k_\rho(\mathbf{x},\mathbf{x}';\theta_\rho)k_f\left(\hat{f}^{l-1}(\mathbf{x}),\hat{f}^{l-1}(\mathbf{x}');\theta_f\right)+k_\delta(\mathbf{x},\mathbf{x}';\theta_\delta).$$

For simplicity, we rewrite it as

$$k((\mathbf{x},f(\mathbf{x})),(\mathbf{x}',f(\mathbf{x}')))=k_\rho(\mathbf{x},\mathbf{x}')k_f(f(\mathbf{x}),f(\mathbf{x}'))+k_\delta(\mathbf{x},\mathbf{x}'),$$

where $k_\rho(\mathbf{x},\mathbf{x}')$, $k_f(f(\mathbf{x}),f(\mathbf{x}'))$ and $k_\delta(\mathbf{x},\mathbf{x}')$ are three SE kernel functions, and its first-order partial derivative w.r.t. $\mathbf{x}'_j$ is

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{x}'_j}k((\mathbf{x},f(\mathbf{x})),(\mathbf{x}',f(\mathbf{x}')))\\ =&\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}k_f(f(\mathbf{x}),f(\mathbf{x}'))+k_\rho(\mathbf{x},\mathbf{x}')\frac{k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial \mathbf{x}'_j}+\frac{\partial k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}\\ =&\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}k_f(f(\mathbf{x}),f(\mathbf{x}'))+k_\rho(\mathbf{x},\mathbf{x}')\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}'_j}+\frac{\partial k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j},\end{aligned}$$

where $\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}$, $\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}$, and $\frac{\partial k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}$ are computed by (A.1). Here, $\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}'_j}$ is the prediction of the first-order partial derivatives in the LF model.

The second-order partial derivative of MFGP kernel is given by,

$$\begin{aligned}&\frac{\partial^2}{\partial \mathbf{x}_i\partial \mathbf{x}'_j}k((\mathbf{x},f(\mathbf{x})),(\mathbf{x}',f(\mathbf{x}')))\\ =&\frac{\partial}{\partial \mathbf{x}_i}\left[\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}k_f(f(\mathbf{x}),f(\mathbf{x}'))+k_\rho(\mathbf{x},\mathbf{x}')\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}'_j}+\frac{\partial k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}\right]\\ =&\frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i\partial \mathbf{x}'_j}k_f(f(\mathbf{x}),f(\mathbf{x}'))+\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}'_j}\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x})}\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i}\\ &+\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i}\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}'_j}+k_\rho(\mathbf{x},\mathbf{x}')\frac{\partial}{\partial \mathbf{x}_i}\left(\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\right)\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}'_j}+\frac{\partial^2 k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i\partial \mathbf{x}'_j}\end{aligned}$$

$$
\begin{aligned}
&= \frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i \partial \mathbf{x}_j'} k_f(f(\mathbf{x}), f(\mathbf{x}')) - \frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \\
&\quad - \frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i'} \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} + k_\rho(\mathbf{x},\mathbf{x}') \frac{\partial^2 k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}) \partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} + \frac{\partial^2 k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i \partial \mathbf{x}_j'} \\
&= \frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i \partial \mathbf{x}_j'} k_f(f(\mathbf{x}), f(\mathbf{x}')) - \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \left(\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} + \frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i'} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right) \\
&\quad + k_\rho(\mathbf{x},\mathbf{x}') \frac{\partial^2 k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}) \partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} + \frac{\partial^2 k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i \partial \mathbf{x}_j'},
\end{aligned}
$$

where $\frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i \partial \mathbf{x}_j'}$, $\frac{\partial^2 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial (f(\mathbf{x}))^2}$, $\frac{\partial^2 k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_i \partial \mathbf{x}_j'}$ are computed through (A.2), $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i}$ and $\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}$ are the first-order partial derivatives of the LF model.

Another second-order partial derivative of the MFGP kernel is given by,

$$
\begin{aligned}
&\frac{\partial^2}{\partial (\mathbf{x}_j')^2} k((\mathbf{x}, f(\mathbf{x})), (\mathbf{x}', f(\mathbf{x}'))) \\
&= \frac{\partial}{\partial \mathbf{x}_j'} \left[\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} k_f(f(\mathbf{x}), f(\mathbf{x}')) + k_\rho(\mathbf{x},\mathbf{x}') \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} + \frac{\partial k_\delta(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} \right] \\
&= \frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^2} k_f(f(\mathbf{x}), f(\mathbf{x}')) + 2 \frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \\
&\quad + k_\rho(\mathbf{x},\mathbf{x}') \frac{\partial}{\partial \mathbf{x}_j'} \left[\frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right] + \frac{\partial^2 k_\delta(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^2}.
\end{aligned}
$$

Here,

$$
\begin{aligned}
&\frac{\partial}{\partial \mathbf{x}_j'} \left[\frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right] \\
&= \frac{\partial^2 k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}') \partial \mathbf{x}_j'} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} + \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial^2 f(\mathbf{x}')}{\partial (\mathbf{x}_j')^2} \\
&= \frac{\partial^2 k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial (f(\mathbf{x}'))^2} \left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right)^2 + \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial^2 f(\mathbf{x}')}{\partial (\mathbf{x}_j')^2},
\end{aligned}
$$

where $\frac{\partial^2 f_{*1}(\mathbf{x}')}{\partial (\mathbf{x}_j')^2}$ is the second-order partial derivative in the LF model.

Analogously, the third-order partial derivative of the MFGP kernel is,

$$
\begin{aligned}
&\frac{\partial^3}{\partial (\mathbf{x}_j')^3} k((\mathbf{x}, f(\mathbf{x})), (\mathbf{x}', f(\mathbf{x}'))) = \frac{\partial}{\partial \mathbf{x}_j'} \left(\frac{\partial^2}{\partial (\mathbf{x}_j')^2} k((\mathbf{x}, f(\mathbf{x})), (\mathbf{x}', f(\mathbf{x}'))) \right) \\
&= \frac{\partial^3 k_\rho(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^3} k_f(f(\mathbf{x}), f(\mathbf{x}')) + 3 \frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^2} \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \\
&\quad + 2 \frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} \frac{\partial}{\partial \mathbf{x}_j'} \left(\frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right) \\
&\quad + \frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'} \left(\frac{\partial^2 k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial (f(\mathbf{x}'))^2} \left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right)^2 + \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial^2 f(\mathbf{x}')}{\partial (\mathbf{x}_j')^2} \right) \\
&\quad + k_\rho(\mathbf{x},\mathbf{x}') \frac{\partial}{\partial \mathbf{x}_j'} \left(\frac{\partial^2 k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial (f(\mathbf{x}'))^2} \left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'} \right)^2 + \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial^2 f(\mathbf{x}')}{\partial (\mathbf{x}_j')^2} \right) + \frac{\partial^3 k_\delta(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^3} \\
&= \frac{\partial^3 k_\rho(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^3} k_f(f(\mathbf{x}), f(\mathbf{x}')) + 3 \frac{\partial^2 k_\rho(\mathbf{x},\mathbf{x}')}{\partial (\mathbf{x}_j')^2} \frac{\partial k_f(f(\mathbf{x}), f(\mathbf{x}'))}{\partial f(\mathbf{x}')} \frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}
\end{aligned}
$$

$$+3\frac{\partial k_\rho(\mathbf{x},\mathbf{x}')}{\partial \mathbf{x}_j'}\left(\frac{\partial^2 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial(f(\mathbf{x}'))^2}\left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}\right)^2+\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial^2 f(\mathbf{x}')}{\partial(\mathbf{x}_j')^2}\right)$$

$$+k_\rho(\mathbf{x},\mathbf{x}')\frac{\partial}{\partial \mathbf{x}_j'}\left(\frac{\partial^2 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial(f(\mathbf{x}'))^2}\left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}\right)^2+\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial^2 f(\mathbf{x}')}{\partial(\mathbf{x}_j')^2}\right)+\frac{\partial^3 k_\delta(\mathbf{x},\mathbf{x}')}{\partial(\mathbf{x}_j')^3}.$$

Here

$$\frac{\partial}{\partial \mathbf{x}_j'}\left(\frac{\partial^2 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial(f(\mathbf{x}'))^2}\left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}\right)^2+\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial^2 f(\mathbf{x}')}{\partial(\mathbf{x}_j')^2}\right)$$

$$=\frac{\partial^3 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial(f(\mathbf{x}'))^3}\left(\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}\right)^3+2\frac{\partial^2 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial(f(\mathbf{x}'))^2}\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}\frac{\partial^2 f(\mathbf{x}')}{\partial(\mathbf{x}_j')^2}$$

$$+\frac{\partial^2 k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial(f(\mathbf{x}'))^2}\frac{\partial f(\mathbf{x}')}{\partial \mathbf{x}_j'}\frac{\partial^2 f(\mathbf{x}')}{\partial(\mathbf{x}_j')^2}+\frac{\partial k_f(f(\mathbf{x}),f(\mathbf{x}'))}{\partial f(\mathbf{x}')}\frac{\partial^3 f(\mathbf{x}')}{\partial(\mathbf{x}_j')^3}.$$

Data availability

Data will be made available on request.

References

[1] B.O. Koopman, Hamiltonian systems and transformation in Hilbert space, Proc. Natl. Acad. Sci. 17 (1931) 315–318.
[2] J.N. Kutz, S.L. Brunton, B.W. Brunton, J.L. Proctor, Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.
[3] B. Lusch, J.N. Kutz, S.L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nat. Commun. 9 (2018) 4950.
[4] Y. Meng, J. Huang, Y. Qiu, Koopman operator learning using invertible neural networks, J. Comput. Phys. 501 (2024) 112795.
[5] M. Raissi, P. Perdikaris, G.E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, arXiv:1801.01236, 2018.
[6] M. Raissi, Deep hidden physics models: deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (2018) 1–24.
[7] S.H. Rudy, J. Nathan Kutz, S.L. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, J. Comput. Phys. 396 (2019) 483–506.
[8] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[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] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[11] S. Kim, P.Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Čeperić, M. Soljačić, Integration of neural network-based symbolic regression in deep learning for scientific discovery, IEEE Trans. Neural Netw. Learn. Syst. 32 (2021) 4166–4177.
[12] S.H. Kang, W. Liao, Y. Liu, IDENT: identifying differential equations with numerical time evolution, J. Sci. Comput. 87 (2021) 1.
[13] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (2017) e1602614.
[14] Y. He, S.-H. Kang, W. Liao, H. Liu, Y. Liu, Robust identification of differential equations by numerical techniques from a single set of noisy observation, SIAM J. Sci. Comput. 44 (2022) A1145–A1175.
[15] F. Sun, Y. Liu, Q. Wang, H. Sun, PiSL: physics-informed spline learning for data-driven identification of nonlinear dynamical systems, Mech. Syst. Signal Process. 191 (2023) 110165.
[16] A. Cortiella, K.C. Park, A. Doostan, A priori denoising strategies for sparse identification of nonlinear dynamical systems: a comparative study, J. Comput. Inf. Sci. Eng. (2022) 1–34.
[17] A. Sandoz, V. Ducret, G.A. Gottwald, G. Vilmart, K. Perron, SINDy for delay-differential equations: application to model bacterial zinc response, Proc. R. Soc. A, Math. Phys. Eng. Sci. 479 (2023) 20220556.
[18] F. Van Breugel, J.N. Kutz, B.W. Brunton, Numerical differentiation of noisy data: a unifying multi-objective optimization framework, IEEE Access 8 (2020) 196865–196877.
[19] 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.
[20] R. Fuentes, R. Nayek, P. Gardner, N. Dervilis, T. Rogers, K. Worden, E. Cross, Equation discovery for nonlinear dynamical systems: a Bayesian viewpoint, Mech. Syst. Signal Process. 154 (2021) 107528.
[21] A. Cortiella, Kwang-Chun Park, A. Doostan, Sparse identification of nonlinear dynamical systems via reweighted ℓ_1-regularized least squares, Comput. Methods Appl. Mech. Eng. 376 (2021) 113620.
[22] J.H. Lagergren, J.T. Nardini, G. Michael Lavigne, E.M. Rutter, K.B. Flores, Learning partial differential equations for biological transport models from noisy spatio-temporal data, Proc. R. Soc. A, Math. Phys. Eng. Sci. 476 (2020) 20190800.
[23] Robert Stephany, C. Earls, PDE-LEARN: using deep learning to discover partial differential equations from noisy, limited data, arXiv:2212.04971, 2023.
[24] P. Goyal, P. Benner, Discovery of nonlinear dynamical systems using a Runge–Kutta inspired dictionary-based sparse regression approach, Proc. R. Soc. A, Math. Phys. Eng. Sci. 478 (2022) 20210883.
[25] D.A. Messenger, D.M. Bortz, Weak SINDy: Galerkin-based data-driven model selection, Multiscale Model. Simul. 19 (2021) 1474–1497.
[26] D.A. Messenger, D.M. Bortz, Weak SINDy for partial differential equations, J. Comput. Phys. 443 (2021) 110525.
[27] M.C. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (2000) 1–13.
[28] L. Le Gratiet, J. Garnier, Recursive co-kriging model for design of computer experiments with multiple levels of fidelity, Int. J. Uncertain. Quantificat. 4 (2014) 365–386.
[29] P. Perdikaris, M. Raissi, A. Damianou, N.D. Lawrence, G.E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. R. Soc. A, Math. Phys. Eng. Sci. 473 (2017) 20160751.
[30] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020.

[31] S. Chakraborty, Transfer learning based multi-fidelity physics informed deep neural network, J. Comput. Phys. 426 (2021) 109942.
[32] P. Conti, M. Guo, A. Manzoni, J.S. Hesthaven, Multi-fidelity surrogate modeling using long short-term memory networks, Comput. Methods Appl. Mech. Eng. 404 (2023) 115811.
[33] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, USA, 2006.
[34] E. Snelson, Z. Ghahramani, Sparse Gaussian Processes Using Pseudo-Inputs, Advances in Neural Information Processing Systems, vol. 18, MIT Press, 2005.
[35] H. Liu, Y.-S. Ong, X. Shen, J. Cai, When gaussian process meets big data: a review of scalable gps, IEEE Trans. Neural Netw. Learn. Syst. 31 (2020) 4405–4423.
[36] B.E. Hansen, A modern Gauss–Markov theorem, Econometrica 90 (2022) 1283–1294.
[37] H. Chen, Data-driven sparse identification of nonlinear dynamical systems using linear multistep methods, Calcolo 60 (2023) 11.
[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: an imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8024–8035.
[39] T.A. Driscoll, N. Hale, L.N. Trefethen, Chebfun Guide, Pafnuty Publications, Oxford, 2014.
[40] A.A. Kaptanoglu, B.M. de Silva, U. Fasel, K. Kaheman, A.J. Goldschmidt, J. Callaham, C.B. Delahunt, Z.G. Nicolaou, K. Champion, J.-C. Loiseau, J.N. Kutz, S.L. Brunton, PySINDy: a comprehensive Python package for robust sparse system identification, J. Open Sour. Softw. 7 (2022) 3994.
[41] K. Kaheman, S.L. Brunton, J. Nathan Kutz, Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data, Mach. Learn.: Sci. Technol. 3 (2022) 015031.
[42] Y. He, S.H. Kang, W. Liao, H. Liu, Y. Liu, Group projected subspace pursuit for identification of variable coefficient differential equations (GP-IDENT), J. Comput. Phys. 494 (2023) 112526.