Bayesian Spline Learning for Equation Discovery of Nonlinear Dynamics with Quantified Uncertainty

Anonymous Author(s)
Affiliation
Address
email

Abstract

Nonlinear dynamics are ubiquitous in science and engineering applications, but the physics of most complex systems is far from being fully understood. Discovering interpretable governing equations from measurement data can help us understand and predict the behavior of complex dynamic systems. Although extensive work has recently been done in this field, robustly distilling explicit model forms from very sparse data with considerable noise remains intractable. Moreover, quantifying and propagating the uncertainty of the identified system from noisy data is challenging, and relevant literature is still limited. To bridge this gap, we develop a novel Bayesian spline learning framework to identify parsimonious governing equations of nonlinear (spatio)temporal dynamics from sparse, noisy data with quantified uncertainty. The proposed method utilizes spline basis to handle the data scarcity and measurement noise, upon which a group of derivatives can be accurately computed to form a library of candidate model terms. The equation residuals are used to inform the spline learning in a Bayesian manner, where approximate Bayesian uncertainty calibration techniques are employed to approximate posterior distributions of the trainable parameters. To promote the sparsity, an iterative sequential-threshold Bayesian learning approach is developed, using the alternative direction optimization strategy to systematically approximate L0 sparsity constraints. The proposed algorithm is evaluated on multiple nonlinear dynamical systems governed by canonical ordinary and partial differential equations, and the merit/superiority of the proposed method is demonstrated by comparison with state-of-the-art methods.

1 Introduction

In the realm of science and engineering, dynamical systems are ubiquitous. However, in actual circumstances, the governing equations behind complicated dynamics may not be completely understood, preventing researchers from developing first-principled models. On the other hand, the ever-increasing data availability opens up new avenues for scientists to identify predictive models from enormous observation data, a process known as *system identification* (SI). Recent advances in deep learning have prompted the rapid development of powerful SI models for high-dimensional problems using deep neural networks (DNNs). Many DNN-based SI models have been proposed to learn differential operators for complex (spatio)temporal physics from data and shown good potential in terms of data reproduction and state prediction [1–3]. However, deep learning models usually lack interpretability and are difficult to comprehend. Furthermore, it is questionable in terms of generalizability when compared to first-principle models, as such black-box DNN models provide less insight into the underlying processes.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Instead of identifying a black-box model, we focus on extracting analytical equation forms from data, which has higher interpretability and has the potential to advance our knowledge of unknown physics. The sparse identification of non-linear dynamic (SINDy) algorithm [4] is an excellent development along this path. The central idea is to use sparse linear regression to uncover parsimonious governing equations from a dictionary of basis functions constructed by data, where the sparsity is promoted by pruning out redundant terms based on certain specified thresholds [5]. SINDy, although showing great promise, faces several challenges: (1) it heavily relies on high-quality data to extract derivative information, which is typically based on finite difference (FD) methods, making it impossible to handle incomplete, scarce, or noisy data; (2) it is formulated in a deterministic fashion and cannot account for the uncertainties from multiple sources, which is critical for real-world applications where data is frequently corrupted and can be very sparse. In the past few years, the SINDy framework has been further improved in various aspects to address these challenges, e.g., enhancing the library [6] or using deep learning for denoising and derivative computation by fitting the noisy data in a decoupled [7–9] or coupled manner [10–14]. For uncertainty quantification, the dictionary-based equation discovery algorithms have been recently extended to Bayesian settings [15–17], based on the idea of sparse Bayesian learning pioneered by Tipping and co-workers [18–22].

Despite recent progress and extensive work in this field, reliably distilling explicit equation forms from very sparse data with significant noise remains an unsolved challenge. There are still significant gaps in handling data scarcity and noise, quantifying and reducing multi-source uncertainties, and promoting sparsity, which most existing equation discovery techniques struggle to simultaneously address. To this end, we propose a novel Bayesian Spline Learning (BSL) approach to identify parsimonious ordinal/partial differential equations (ODEs/PDEs) from sparse and noisy measurements; meanwhile, the associated uncertainties are quantified. To deal with data scarcity and measurement noise, the proposed BSL uses a spline basis, on which a collection of derivatives can be reliably computed for the library construction. The posterior distributions of spline-based model parameters are approximated by a stochastic gradient descent (SGD) trajectory-based training scheme, where the first moment of SGD iterations is computed by stochastic weight averaging approach [23]. The proposed BSL approach is effective in two aspects. On the one hand, the spline-based representation can help to interpolate locally the spatiotemporal field and perform differentiation analytically. As a result, it considerably enhances learning efficiency in cases with sparse and noisy data. The posterior distributions of spline, library, and equation coefficients are estimated simultaneously, without adding too much overhead to the training process. Besides the measurement uncertainty, the model-form uncertainty is also obtained in the proposed method, which can be used for downstream Bayesian data assimilation, where online data is assimilated to improve the predictability of the identified system for chaotic scenarios

2 Related Work

The data in real-world circumstances is frequently sparse in spatial/temporal domains and may contain considerable noise, posing significant challenges to SINDy or its variants [4–6, 16, 24–27]. Deep learning (DL) has been leveraged as a superb interpolator for concurrently generating metadata and smoothing high-frequency noise [28], effectively improving the performance in identifying equations from imperfect measurements [7, 10–14]. For example, automatic differentiation (AD) is effective for computing derivatives analytically from sparse, noisy data using a point-wise multi-layer perceptron (MLP) [13, 29, 30]. However, it is difficult to impose locality constraints in the point-wise formulation. Wandel et al. [31] demonstrated the merit of employing a spline basis for analytically calculating derivatives while enforcing the locality of spatiotemporal fields. Owing to the superiority of spline-based differentiation compared to numerical discretization, physics-informed spline networks remarkably outperform point-wise physics-informed neural networks (PINN) in the context of solving PDEs [31] and discovering ODEs [14].

However, these models fail to simultaneously deal with ODE and PDE systems when the data is sparse and substantially corrupted. For the PDE datasets, for example, the PINN-based sparse regression (PINN-SR) [13] works admirably, but it fails to converge on ODE datasets. On the other hand, algorithms such as physics-informed spline learning (PiSL) [14] that performs very well for ODE discovery cannot handle PDE problems due to the limitation of the B-spline basis adopted. More importantly, all these SOTA sparse learning algorithms are formulated in deterministic settings and uncertainties introduced from data/library imperfection cannot be quantified.

For inverse problems like equation discovery, it is natural to use Bayesian framework to quantify and analyze the prediction uncertainty. To enable the posterior computation for high-dimensional trainable parameters that are intractable by traditional Bayesian inference, people have resorted to various approximation strategies, e.g., variational inference [32–34], Monte Carlo dropout [35], Bayes by backprop [36], Laplace approximation [37], and deep ensemble approaches [38]. Although these techniques have had great success, training may still be challenging, and costs will rise dramatically as the problems become more sophisticated. Alternatively, we chose to employ an SGD trajectory-based algorithm, Stochastic Weight Averaging-Gaussian (SWAG) [23], where the information in the SGD trajectory is exploited to approximate the posterior. As demonstrated empirically, SWAC is well scalable to high-dimensional problems and can accurately estimate uncertainty across many different Bayesian learning tasks [23, 39].

Data assimilation (DA) has been widely used in numerical weather prediction (NWP) by fusing online sensing data into a predictive model for nonlinear dynamics forecast. People have recently integrate deep learning into DA to improve online prediction performance [40–42]. However, the predictive model in DA is assumed to be known *a priori*, and uncertainties from the model and observations, which are required for DA, are usually hard to obtain. In our framework, the predictive model can be unknown *a priori* and will be identified to assimilate additional data for online forecasting. Moreover, instead of arbitrarily guessing the observation and model-form errors, this information can be learned in the proposed BSL framework, as the data and model-form uncertainties are quantified. The UQ capability of BSL naturally integrates the equation discovery with Bayesian DA techniques.

The main contributions of this work are three-fold: (1) we extended spline learning for sparse equation discovery of spatiotemporal physics governed by ODEs or PDEs; (2) we developed sparsity-promoting Bayesian learning for UQ tasks; and (3) the proposed BSL framework can seamlessly integrate with Bayesian data assimilation techniques to improve online dynamics forecasting.

3 Methodology

Let us consider a dynamical system, which is governed by a parametric ODE/PDE system in the following general form

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

where $\mathbf{u}$ denotes the state vector, for ODE systems, $\mathbf{u} = [u_1(t), u_2(t), \cdots, u_d(t)]^T \in \mathbb{R}^d$ depends only on time t, for PDE systems, $\mathbf{u}(\mathbf{x}, t) = [u_1(\mathbf{x}, t), u_2(\mathbf{x}, t), \cdots, u_d(\mathbf{x}, t)]^d \in \mathbb{R}^d$ depends on both time t and space $\mathbf{x}$, and $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^d$ represents unknown nonlinear functions. The states are observed at discrete times $\{t_i\}_{i=1}^n$ and at spatial locations $\{\mathbf{x}_j\}_{j=1}^s$ for PDE systems. Let $\tilde{\mathbf{u}}$ denote the noisy observation vector, the observation set is $\tilde{\mathbf{U}} = \{\tilde{\mathbf{u}}(t_1), \tilde{\mathbf{u}}(t_2), \cdots, \tilde{\mathbf{u}}(t_n)\}^T \in \mathbb{R}^{n\times d}$ and $\tilde{\mathbf{U}} = \{\tilde{\mathbf{u}}(\mathbf{x}_1, t_1), \tilde{\mathbf{u}}(\mathbf{x}_2, t_1), \cdots, \tilde{\mathbf{u}}(\mathbf{x}_s, t_n)\}^T \in \mathbb{R}^{(n\times s)\times d}$ for ODE/PDE systems, respectively. Our goal here is to explicitly discover the parsimonious form of $\mathcal{F}(\cdot)$ from a library of candidate basis functions and quantify the associated uncertainty given noisy observation data.

3.1 Overview

Given the dataset $\tilde{\mathbf{U}}$, the SI problem can be solved by sparse regression techniques with a predefined library $\mathbf{\Phi}(\mathbf{u})$ of m basis functions,

$$\mathbf{\Phi}(\mathbf{u}) = [\phi_1(\mathbf{u}), \phi_2(\mathbf{u}), \cdots, \phi_m(\mathbf{u})] \in \mathbb{R}^m, \tag{2}$$

where $\phi_i : \mathbb{R}^d \to \mathbb{R}, 1 \le i \le m$ denotes a basis function, which, for instance, can be the polynomial or trigonometric function. Hence, the matrix of library terms evaluated at observed states is defined as,

$$\begin{aligned} \text{ODE}: \quad \mathbf{\Phi}(\tilde{\mathbf{U}}) &= \left[\mathbf{\Phi}\big(\tilde{\mathbf{u}}(t_1)\big)^T, \mathbf{\Phi}\big(\tilde{\mathbf{u}}(t_2)\big)^T, \cdots, \mathbf{\Phi}\big(\tilde{\mathbf{u}}(t_n)\big)^T\right]^T \in \mathbb{R}^{n\times m}, \\ \text{PDE}: \quad \mathbf{\Phi}(\tilde{\mathbf{U}}) &= \left[\mathbf{\Phi}\big(\tilde{\mathbf{u}}(\mathbf{x}_1, t_1)\big)^T, \mathbf{\Phi}\big(\tilde{\mathbf{u}}(\mathbf{x}_2, t_1)\big)^T, \cdots, \mathbf{\Phi}\big(\tilde{\mathbf{u}}(\mathbf{x}_s, t_n)\big)^T\right]^T \in \mathbb{R}^{(n\times s)\times m}. \end{aligned} \tag{3}$$

Given the library, the overview of the BSL framework is shown in Fig. 1. Firstly, a spline-based model is constructed to represent state variables and their derivatives by denoising the observation

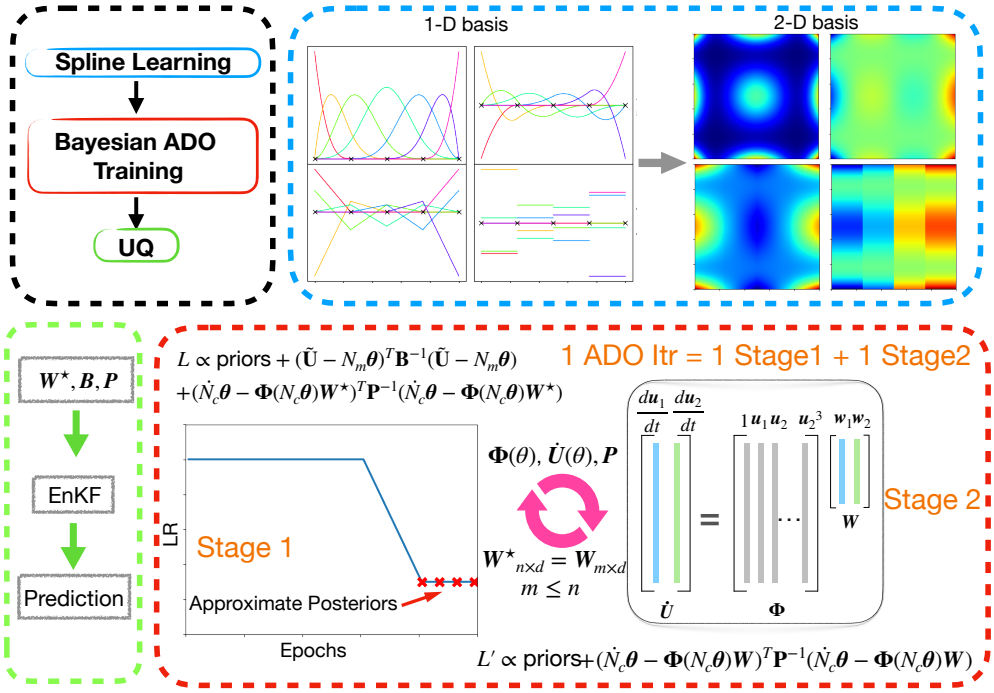

Figure 1: Overview of method. Black box: overall work flow. Blue box: a demo for spline basis with derivatives in 1D and 2D cases. Red box: a sketch for the Bayesian ADO training process. Green box: the data assimilation enhanced uncertainty quantification.

data, which is illustrated in the blue box and discussed in Sec.3.2. Then the spline-based model and sparse regression are trained simultaneously by alternating direction optimization (ADO), as shown in the red box. Specifically, a single ADO iteration contains two sub-processes, where sub-process I trains the spline-based model using log posterior loss and passes the trainable parameters and the noise estimations to sub-process II, which uses a Bayesian SINDy-like method to prune out redundant terms in the library as defined in Eq. 3, and then updates the number of relevant terms in the training loss for sub-process I. After several ADO iterations, the parsimonious form of the governing equations and the posterior distribution for the coefficients will be estimated. More details about the ADO iterations in the Bayesian framework are shown in Sec. 3.3 and supplemental materials. Finally, with the estimated posterior, the predictive uncertainty can be quantified by evaluating the identified system with an ensemble of parameters. To further improve the prediction capability, especially for chaotic systems, we propose to leverage data assimilation techniques, which is shown in the green box and discussed in Sec.3.4 and supplemental materials.

3.2 Spline-based learning

Several previous works have already shown the potential of spline-based learning and demonstrated the advantages compared with the classical DL structures, e.g.,(MLP, CNN) [14, 31, 43]. Therefore, we use this structure to smooth the solution fields based on the noisy measurement and then identify the underlying PDE/ODE and calculate the corresponding derivatives from the spline-reconstructed fields. The B-spline curves, based on the De Boor's algorithm, are defined in a recursive way as:

$$
\begin{aligned}
N_{s,0}(t) &= \begin{cases} 1, & \text{if } \tau_s \leq t \leq \tau_{s+1}. \\ 0, & \text{otherwise.} \end{cases} \\
N_{s,k}(t) &= \frac{t - \tau_s}{\tau_{s+k} - \tau_s} N_{s,k-1}(t) + \frac{\tau_{s+k+1} - t}{\tau_{s+k+1} - \tau_{s+1}} N_{s+1,k-1}(t),
\end{aligned}
\tag{4}
$$

where τ_s is the location of knots, k is the degree of polynomial. When $k = 3$, it is the well-used Cubic-B Spline curve. With the defined basis, the spline interpolation can be write as $y(t) = \Sigma_{s=0}^{r+k-1} N_{s,k}(t)\theta_s$. And the number of control points (trainable weights) $\boldsymbol{\theta} \in \mathbb{R}^{r+k}$ is chosen

empirically. It can be proved that the derivative of p order B-spline basis is a function of $p-1$ order B-spline, written as:

$$\frac{d}{dt}N_{s,k}(t) = \frac{k}{\tau_{s+k}-\tau_s}N_{s,k-1}(t) - \frac{k}{\tau_{s+k+1}-\tau_{s+1}}N_{s+1,k-1}(t). \tag{5}$$

The proof of Eq. 5 is attached in supplemental materials. The higher-order derivative can be calculated by recursively using Eq. 5. The analytical derivatives of spline basis are very beneficial for PDE discovery tasks since it always involves constructing library terms containing high order derivatives. With a proper order p, the first p derivatives are accurate, and there is no error introduced during the derivation, as opposed to using the numerical methods to approximate derivatives. The spline function and its derivatives are defined in a one-dimensional scenario. It is straightforward to extend to n-dimension by direct using tensor-product. For example, for a two-dimensional problem with spatial-temporal fields, the basis can be defined as

$$N_{s_1,s_2}^{k_1,k_2}(t_i,x_j) = N_{s_1,k_1}(t_i)N_{s_2,k_2}(x_j). \tag{6}$$

Here the two-dimensional basis is denoted by a different style to write it compactly. Similarly, the partial derivative for two-dimensional basis is defined as:

$$\frac{\partial^{(q_1+q_2)}N_{s_1,s_2}^{k_1,k_2}}{\partial t^{(q_1)}\partial x^{(q_2)}}(t_i,x_j) = \frac{d^{(q_1)}N_{s_1,k_1}}{dt^{(q_1)}}(t_i)\frac{d^{(q_2)}N_{s_2,k_2}}{dx^{(q_2)}}(x_j). \tag{7}$$

With the definition of sparse system identification and spline reconstruction, the whole spline learning can be stated as follows: given noisy measurement data $\tilde{\mathbf{U}}$, find the best sets of weights $\boldsymbol{\theta}$ and $\mathbf{W}$ so that data fitting loss and the weakly physics-informed loss can be minimized under sparsity constraints , as shown in Eq. 8:

$$\{\boldsymbol{\theta},\mathbf{W}\} = \arg\min_{\boldsymbol{\theta}',\mathbf{W}'} \frac{1}{\mathcal{N}_m}||\mathbf{N}_m\boldsymbol{\theta}' - \tilde{\mathbf{U}}||_{L_2} + \frac{1}{\mathcal{N}_c}||\boldsymbol{\Phi}(\mathbf{N}_c\boldsymbol{\theta}')\mathbf{W}' - \dot{\mathbf{N}}_c\boldsymbol{\theta}'||_{L_2} + \lambda||\mathbf{W}'||_{L_\alpha}. \tag{8}$$

Here, $\mathbf{N}_m$ and $\mathbf{N}_c$ denote the spline basis matrices evaluated at measurement and collocation locations. Moreover, $\mathcal{N}_m$ and $\mathcal{N}_c$ are numbers of measurement data and collocation points. Furthermore, the alternating direction optimization (ADO) shown in previous works [13, 14] can be adopted to minimize the loss function efficiently, and the details are attached in supplemental materials.

3.3 Sparse system identification in Bayesian formulation

Bayesian methods provide a natural probabilistic representation of uncertainty, which is crucial for model predictions. System identification from noisy and sparse measurements generally contains two types of errors (similar to the hidden Markov model): (1) Observation error, where the data is noisy, and the smoothed data is reconstructed through spline-based learning, as shown in Eq. 9

$$\tilde{\mathbf{U}} = \mathbf{N}_m\boldsymbol{\theta}' + \boldsymbol{\epsilon}_1, \tag{9}$$

where $\boldsymbol{\epsilon}_1$ represents the observation error. (2) Evolution error or model form error, since the discovered system cannot be exact due to library imperfection and needs to be reformulated as,

$$\dot{\mathbf{u}} = \boldsymbol{\Phi}(\mathbf{u})\boldsymbol{W'} + \boldsymbol{\epsilon}_2, \tag{10}$$

where $\boldsymbol{\epsilon}_2$ represents model form error. And Eq. (10) is evaluated on collocation points.

In this work, these error terms are modeled as zero-mean multivariate Gaussian random variables: $\boldsymbol{\epsilon_1} \sim \mathcal{N}(\mathbf{0},\mathbf{B})$ and $\boldsymbol{\epsilon_2} \sim \mathcal{N}(\mathbf{0},\mathbf{P})$. And we further assume that these error covariance matrices $\mathbf{B},\mathbf{P}$ are diagonal matrix with diagonal terms $\{b_k\}$ and $\{p_k\}$ with $1 \le k \le d$ and they are learn-able parameters during training.

According to Bayes' rule, the posterior can be written as:

$$p(\boldsymbol{\theta},\mathbf{W},\mathbf{B},\mathbf{P}|\tilde{\mathbf{U}},\dot{\mathbf{U}}) \propto p(\boldsymbol{\theta},\mathbf{W},\mathbf{B},\mathbf{P})p(\tilde{\mathbf{U}},\dot{\mathbf{U}}|\mathbf{W},\boldsymbol{\theta},\mathbf{B},\mathbf{P}). \tag{11}$$

Here $\dot{\mathbf{U}} = \dot{\mathbf{N}}_c\boldsymbol{\theta}$ denotes the derivative estimation on all the collocation points. The prior is further decomposed by

$$p(\mathbf{W},\boldsymbol{\theta},\mathbf{B},\mathbf{P}) \propto p(\mathbf{W}|\alpha)p(\alpha)p(\boldsymbol{\theta}|\beta)p(\beta)p(\mathbf{B})p(\mathbf{P}). \tag{12}$$

Currently, we specify the prior for the linear coefficient matrix as a zero mean Gaussian distribution $p(\mathbf{W}|\alpha) = \mathcal{N}(\mathbf{W}|0, \alpha^{-1}\mathbf{I})$ with the hyper prior as a Gamma distribution $p(\alpha) = \text{Gamma}(\alpha|a_0, b_0)$. Similarly, we also define the prior for the spline trainable parameters as zero mean Gaussian distribution with the hyper prior as another Gamma distribution $p(\boldsymbol{\theta}|\beta) = \mathcal{N}(\boldsymbol{\theta}|0, \beta^{-1}\mathbf{I})$ and $p(\beta) = \text{Gamma}(\beta|a_1, b_1)$. To account for the data uncertainty and process uncertainty, the diagonal covariance matrices $\mathbf{B}$ and $\mathbf{P}$ are also set as learn-able during the training. And hence improper uniform priors are used for $\mathbf{B}$ and $\mathbf{P}$.
The likelihood consists of two parts, as:

$$p(\tilde{\mathbf{U}}, \dot{\mathbf{U}}|\mathbf{W}, \boldsymbol{\theta}, \mathbf{B}, \mathbf{P}) = p(\tilde{\mathbf{U}}|\boldsymbol{\theta}, \mathbf{B})p(\dot{\mathbf{U}}|\mathbf{W}, \mathbf{P}, \boldsymbol{\theta}), \tag{13}$$

where

$$p(\tilde{\mathbf{U}}|\boldsymbol{\theta}, \mathbf{B}) \propto \exp\{-\frac{1}{2}(\tilde{\mathbf{U}} - \mathbf{N}_m\boldsymbol{\theta})^T\mathbf{B}^{-1}(\tilde{\mathbf{U}} - \mathbf{N}_m\boldsymbol{\theta})\}, \tag{14}$$

$$p(\dot{\mathbf{U}}|\mathbf{W}, \mathbf{P}, \boldsymbol{\theta}) \propto \exp\{-\frac{1}{2}(\dot{\mathbf{N}}_c\boldsymbol{\theta} - \boldsymbol{\Phi}(\mathbf{N}_c\boldsymbol{\theta})\boldsymbol{W})^T\mathbf{P}^{-1}(\dot{\mathbf{N}}_c\boldsymbol{\theta} - \boldsymbol{\Phi}(\mathbf{N}_c\boldsymbol{\theta})\boldsymbol{W})\}. \tag{15}$$

Traditional Bayesian sampling approaches (i.e., Markov chain Monte Carlo methods) can be usually intractable and expensive, especially when the parameter dimensionality is high. Therefore, researchers tend to use alternative approximation approaches. Current work uses a stochastic gradient descent (SGD) trajectory-based approach, Stochastic Weight Averaging Gaussian (SWAG) [23] algorithm to approximately sample from the posterior distribution. This method approximates the posterior by collecting the parameters near the loss plateau after a sufficient number of the training steps. To further reduce the inference cost, we construct a subspace by finding the PCA components of the SWAG trajectories [44] and then draw samples in the subspace instead. In terms of the loss function in the probabilistic model, we chose to maximize the log form of the posterior density Eq. 11, also by leveraging the ADO algorithm. The sub-process I minimizes the log density function defined by Eq. 11-13 and the sub-process II adopts the Bayesian variants of SINDy algorithm modified from [17, 19]. The likelihood function in sub-process II has the same form as in Eq. 15 but with a different sparsity promoting prior $p(\mathbf{W}|\mathbf{A}) = \prod_{j=1}^{m} \mathcal{N}(\mathbf{W}_j|0, {\alpha'}_j^{-1})$, where $\boldsymbol{A} = [\alpha'_1, \alpha'_2, ..., \alpha'_m]^T$. In a single ADO iteration, the sub-process I provides the updated θ and $\mathbf{P}$ to sub-process II for constructing the library terms. While the sub-process II shrinks the library terms and passes updated library/weight structure $\mathbf{W}$ back to sub-process I. The whole training requires multiple ADO iterations before it reaches the final balance, where no more terms will be pruned out in sub-process II. The detailed ADO algorithm is listed in supplemental materials. After obtaining the approximated posterior distribution, predicting uncertainty can be estimated by marginalizing out the model parameters.

3.4 Data assimilation for enhanced predictability

For a chaotic system, one notorious problem is that a slight perturb in any model parameters can significantly influence the prediction. For example, in numerical weather prediction (NWP) tasks where the researcher always needs to predict the behavior of chaotic weather systems, various data assimilation (DA) techniques have been developed to assimilate the available data and the known equations. Kalman filter and its variants are successful mathematical tools for data assimilation. Our Bayesian formulation, providing the observation error covariance matrix $\mathbf{B}$ and evolution error covariance matrix $\mathbf{P}$, can be naturally incorporated with the Kalman filter frameworks to improve the predictability of chaotic systems. Current works choose the ensemble Kalman filter for the task, and more background can be found in supplemental materials.

4 Experiment and Result

In this section, we first show the equation discovery and uncertainty quantification results for nonlinear ODE systems. We also show that incorporating DA techniques can improve the predicting ability for chaotic systems. Finally, we present the PDE discovery results for several canonical PDE systems.

The first pedagogical ODE example is the Van der Pol oscillator, which is defined as $\frac{dx}{dt} = y$ and $\frac{dy}{dt} = \mu(1 - x^2)y - x$ with $\mu = 0.5$. The data is corrupted with 5% noise, and the library

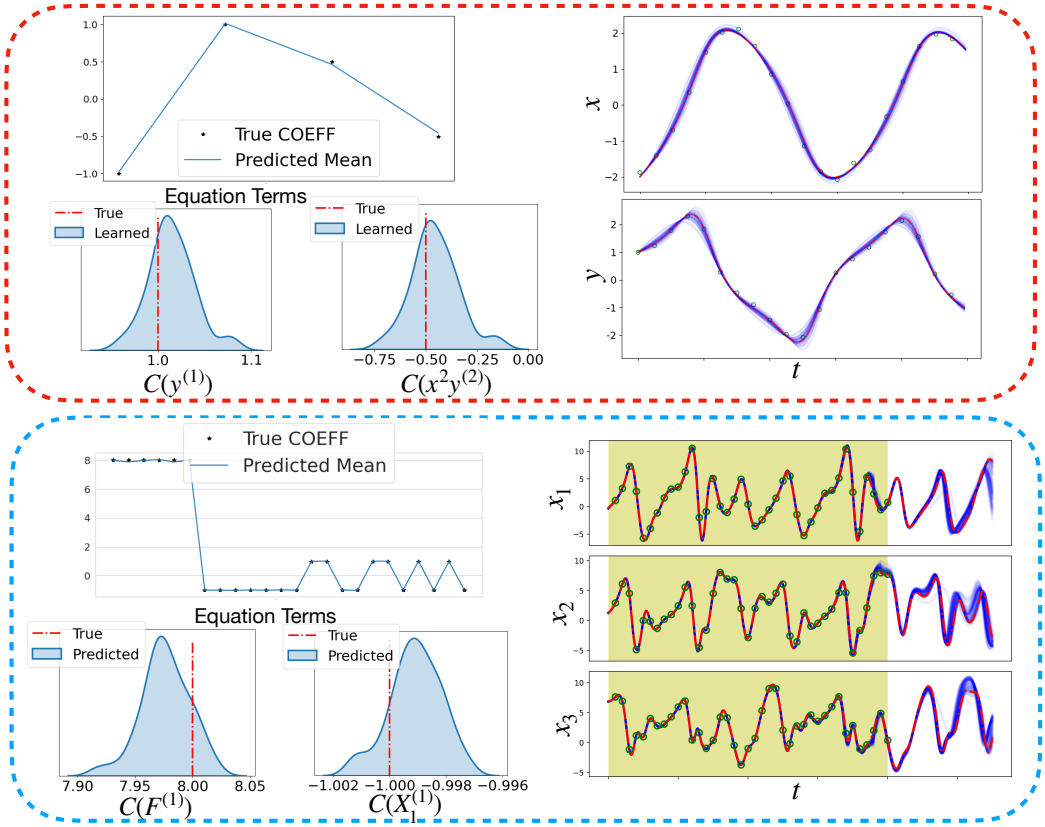

Figure 2: The discovery results for ODE systems; the red box shows results for the Van der Pol system, and the blue box shows the result for Lorenz 96 system. The layout inside each box follows the rules below. Upper left sub-fig: discovered mean for the relevant library terms; Lower left sub-figs: selected posterior distribution for the identified distributions; Right sub-figs: ensemble prediction plots for UQ. More explanation for the upper left sub-fig, for VdP, the governing equation in parametric form is $\frac{dx}{dt} = y$ and $\frac{dy}{dt} = -x - 0.5x^2y + 0.5y$. We represent the actual values in a order of $\{x^{(2)}, y^{(1)}, y^{(2)}, (x^2y)^{(2)}\} \rightarrow \{-1, 1, 0.5, -0.5\}$, where the subscripts denotes the corresponding equation. Similarly for the Lorenz96 system, the compact form of the governing equation is $\frac{dX_i}{dt} = (X_{i+1} - X_{i-2})X_{i-1} - X_i + F$, with periodic boundary conditions $X_{-1} = X_{n-1}$, $X_0 = X_n$ and $X_{n+1} = X_1$. We plot the predicted ODE coefficient on top of the true coefficients

consists of polynomials of state variables up to $3rd$ order. The parsimonious model structure can be correctly identified, and the result is shown in the upper red box of Fig. 2. There are five sub-figures, which depict the library discovery, coefficient distribution, and the forward propagated UQ results. Specifically, the single upper left figure shows the mean of discovered coefficients (blue lines —) and the truth equation coefficients (black stars $*$), where the horizontal axis represents term indices. For example, the Van der Pol system has four different terms, and the x-axis ranges from 1 to 4. The two sub-figures in the lower left part show probability density distributions (PDF) of two identified coefficients, where truth equation coefficients (red lines —) fall within the confidence interval with high probability. The right sub-figures show the propagated ensemble results (blue lines —) based on the discovered equations, measurement data (green circles ○), and the true state trajectories of x and y (red dashed lines — —). The result clearly shows that the ensemble predictions can cover the data, and the uncertainty range fluctuates around the truth state values.

The second ODE example is a chaotic system, Lorenz 96. It is a simplified mathematical model for atmospheric convection, defined as:

$$\frac{dX_i}{dt} = (X_{i+1} - X_{i-2})X_{i-1} - X_i + F, \qquad i = 1, 2, ...n \tag{16}$$

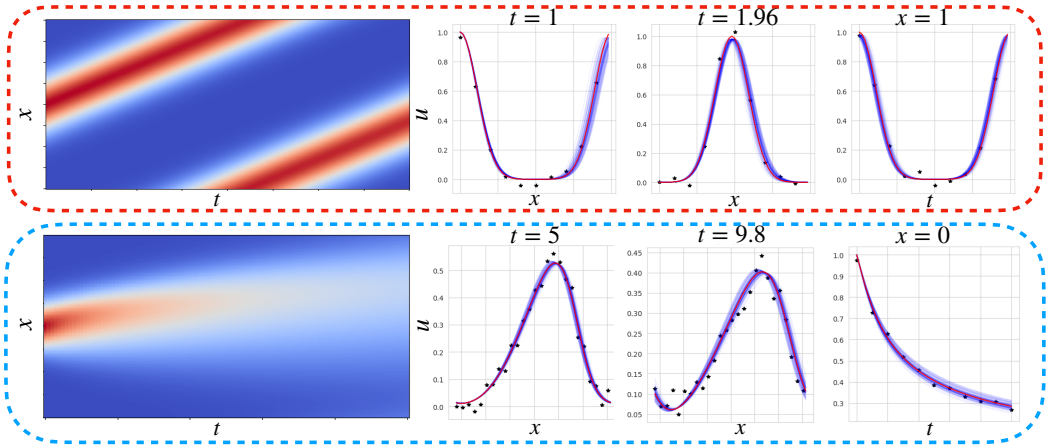

Figure 3: The discovery results for PDE systems; the red box shows results for the advection system, and the blue box shows the result for the Burgers' system. The layout inside each box follows the rules below. Leftmost sub-fig: true contour plot; Middle two sub-figs: the spatial results at different time t; Rightmost sub-figs: the temporal result at a fixed point x.

with periodic boundary conditions $X_{-1} = X_{n-1}, X_0 = X_n$ and $X_{n+1} = X_1$. We chose $n = 6$ in current case and $F = 8$ for the forcing terms. The measurement states variables are corrupted with 5% Gaussian noise, and the library consists of polynomials of state variables up to $3rd$ order. The discovery result for these systems is shown in the lower blue box in Fig. 2 following the same layout as the Van der Pol systems. The upper left figure indicates that the sparsity structure can be identified, and the horizontal axis marks 24 parsimonious terms (4 terms for each state, 6 states in total) out of 84 library terms. The detailed PDF plots show that posterior distributions still cover truth values with high probability. Although the discovery result is quite accurate, the forward simulations of identified Lorenz 96 system would induce a large phase difference compared with truth trajectories due to the chaotic nature of the underlying system (shown in Supplemental Materials). To alleviate the chaotic behavior and improve the predictability, we seamlessly coupled the ensemble Kalman filter (EnKF), a classical DA technique, to assimilate noisy measurements with the identified system. The observation error matrix $\boldsymbol{B}$ and evolution error matrix $\boldsymbol{P}$ required for the EnKF scheme can be directly passed from the Bayesian spline learning framework. The right subplots show the DA results for the 3 out of 6 state variables to save space. The horizontal axis marks the evolving time, and the vertical axis represents the state variables. In each subplot, the green shaded region indicates the time interval with available noisy measurement data (marked by green circles ($\circ$)). And the red solid-line ($-$) is the true L96 states. Finally, the blue curves ($-$) are the ensemble predictions. It can be seen that the ensemble predictions inside the region with measurement data almost overlap with the true trajectory. Furthermore, the ensemble uncertainty grows more significantly in the extrapolation region, but the ensemble ranges still fluctuate around the true trajectory. Note that the prediction for the chaotic system with the DA process is much better than directly forward simulating the identified system, where the predictable interval is only about 1 second. More discovery results are attached in Supplemental Materials.

Then, we use the proposed BSL framework to identify classical PDE systems and show two of them in the main text. They include advection equation $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$ and Burgers' equation $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - 0.5\frac{\partial^2 u}{\partial x^2} = 0$. The results are shown in Fig. 3, inside the red and blue boxes, respectively. Inside each box, the leftmost contour plot shows the true state value in the spatiotemporal field, and the right three sub-figures show the UQ results for different cross-sections from the left contour plot. Specifically, the first 2 out of 3 subplots are sampled from different time instances, and the last subplot is sampled from a fixed coordinate point, which gives a clearer view of the UQ trend along the spatiotemporal field. The red lines ($-$) represent the truth value, blue lines ($-$) are obtained by ensemble predictions, and black stars ($*$) represent sparse and noisy measurements. The prediction from the identified PDE system is accurate in the spatiotemporal field, and the ensemble fluctuates around the truth state variables. Finally we benchmark our proposed method with several state-of-the-art discovery algorithms (PINN-SR [13], SINDy [5] and RVM [45]). The error metric is defined as:

$$\textbf{rmse} = \frac{||\mathbf{C}_{\text{Discovery}} - \mathbf{C}_{\text{True}}||_2}{||\mathbf{C}_{\text{True}}||_2} \tag{17}$$

$$\mathbf{M_P} = \frac{||\mathbf{C}_{\text{Discovery}} \odot \mathbf{C}_{\text{True}}||_0}{||\mathbf{C}_{\text{Discovery}}||_0} \tag{18}$$

$$\mathbf{M_R} = \frac{||\mathbf{C}_{\text{Discovery}} \odot \mathbf{C}_{\text{True}}||_0}{||\mathbf{C}_{\text{True}}||_0} \tag{19}$$

where $\mathbf{C}_{\text{Discovery}}$ are the non-zero mean prediction from the posterior distribution and $\mathbf{C}_{\text{True}}$ are the true coefficients of the governing equations. If the method fails to converge or cannot identify the correct parsimonious form, we will report the final result as **Fail**. The errors are scaled by $\times 10^{-3}$ to have a clear comparison. Two additional metrics are defined as precision M_P and recall M_R and they are defined in Eq. 19,where the $\odot$ symbols element-wise product of vectors and the l_0 norm is the non-zero terms in a vector. It can be seen from Table 1 that our BSL method always performs best, when the noise is significant ($> 5\%$), demonstrating its robustness to noise. The PINN-SR can discover the PDE equation with corrupted data set, but it fails to predict accurate time trajectories for ODE systems. This is also reported in a relevant paper [46], that the plain MLP structure is not satisfactory for predicting time series. The SINDy method can behave much better when large high-quality data exist. However, the SINDy method can easily fail when noise is significant. Our BSL requires fewer parameters and is easier to converge due to the enforcement of locality constraints by the spline basis model. Furthermore, the SINDy method requires much more data (100%) to train but is still vulnerable to data noise. These benchmark cases show the potential of our BSL for equation discovery tasks. The full table with extended cases can be found in Tab. 3 in appendix.

Table 1: ODE and PDE discovery comparison

Name	**rmse**(0%)	**rmse**(1%)	**rmse** (large[1])	$\mathbf{M_P}$	$\mathbf{M_R}$[2]	Training Cost[3]
			Van der Pol Oscillator			
BSL(Ours)	**0.2**	2.82	**18.04**	**1**	**1**	$\sim 133(+3)s$
PINN-SR	Fail[4]	Fail	Fail	0.214	0.75	$\sim 1213s$
SINDy	1.0	**1.93**	Fail	0.267	1.0	$\sim 10s$
RVM	1.0	2.54	27.46	**1**	**1**	$\sim 10s$
			Lorenz 96			
BSL(Ours)	**0.269**	1.47	**13.0**	**1**	**1**	$\sim 1654(+438)s$
PINN-SR	Fail	Fail	Fail	0.5	0.22	$\sim 10788s$
SINDy	0.4	0.64	Fail	0.75	**1**	$\sim 10s$
RVM	0.4	**0.6**	49.7	**1**	**1**	$\sim 25s$
			Advection Equation			
BSL(Ours)	**0.26**	**1**	**1.9**	**1**	**1**	$\sim 946(+233)s$
PINN-SR	5.9	4.5	30.4	**1**	**1**	$\sim 650s$
SINDy	2.3	8.2	38.9	**1**	**1**	$\sim 10s$
RVM	0.77	6.76	Fail	0.2	**1**	$\sim 4s$
			Burgers' Equation			
BSL(Ours)	3.62	4.13	**6.38**	**1**	**1**	$\sim 117(+74)s$
PINN-SR	10.2	**3.3**	10.3	**1**	**1**	$\sim 512s$
SINDy	0.826	Fail	Fail	**1**	0.5	$\sim 10s$
RVM	**0.754**	Fail	Fail	0.1429	0.5	$\sim 4s$

[1] Large noise for different cases: Van der Pol: 5%, Lorenz 96: 10%, Advection: 20%, Burgers: 10%,Burgers' with source: 20%, Heat: 15%, Poisson: 5%

[2] $\mathbf{M_P}, \mathbf{M_R}$ are only reported for the largest noise cases

[3] All cases are running on a Nvidia 2070 Ti GPU card

[4] Fail means failure in discovery of the parsimonious ODE/PDE forms.

5 Discussion and Limitation

In the current work, we developed a novel Bayesian spline learning (BSL) framework for equation discoveries from sparse and noisy datasets with quantified uncertainty. The proposed framework improves the result in the following aspects. Firstly,the spline basis can accurately interpolate solution surfaces and analytically compute derivatives to form the candidate library, outperforming the other benchmark methods based on the finite difference (FD) or auto differentiation (AD). The improvement in performance can be accounted for two reasons; Compared to numerical methods FD, it can represent the derivatives with closed-form expression. Compared to AD, it leverages the local support of spline basis, which gives a more physical inductive bias to the learning process. In the case of the Bayesian learning part, the weakly physics-informed likelihood function is constrained to inform the model learning in a Bayesian manner,without introducing significant computational overhead. During the training, we adopt a Bayesian sparsity-promoting ADO iteration strategy to recover the parsimonious governing equation as well as the posterior distribution of its coefficients. Last but not least, the Bayesian DA is also integrated into the BSL framework to improve the online predictability of the chaotic systems, which can potentially benefit real-world tasks like numerical weather forecasting (NWP). The proposed framework is evaluated on discovering multiple canonical ODE and PDE systems, and great superiority has been demonstrated in comparison with state-of-the-art methods. Similarly to all machine learning models, the current work should be used with caution in order not to generate potential negative societal impacts, e.g., modeling natural systems. Admittedly, this work still relies on a pre-defined library of candidate terms, and thus the identified system is largely limited to the functional space determined by the user-specified library. In general, for library-based methods, how to design an inclusive but not unnecessarily large library *a priori* is important yet very challenging, which may require prior knowledge of the system to be identified and thus limit their applications for systems involving complex governing physics. Moreover, in terms of the spline representation, the current work only uses a uniform displacement of knot and uses the tensor product for high dimensional problems. It is *not* an optimal setting in general and limits the method's ability when the spatial dimension is higher. In future work, we aim to apply more advanced techniques, e.g., to find the best knot size and to reduce the computational cost as listed in [47, 48]. We will also adopt a better spline definition as shown in [43], namely, defining spline kernel and only expanding the basis function in the subdomain. It would allow a significant reduction of trainable parameters and B-spline basis functions. A more detailed implementation of the current work can be found in Appendix A.9.

References

[1] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Learning Representations, 2020.

[2] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nature Machine Intelligence 3 (3) (2021) 218–229.

[3] X. Han, H. Gao, T. Pfaff, J.-X. Wang, L. Liu, Predicting physics in mesh-reduced space with temporal attention, in: International Conference on Learning Representations, 2022.

[4] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (15) (2016) 3932–3937.

[5] S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations, Science advances 3 (4) (2017) e1602614.

[6] H. K. Chu, M. Hayashibe, Discovering interpretable dynamics by sparsity promotion on energy and the lagrangian, IEEE Robotics and Automation Letters 5 (2) (2020) 2154–2160.

[7] S. H. Rudy, J. N. Kutz, S. L. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, Journal of Computational Physics 396 (2019) 483–506.

[8] J. Wang, X. Xie, J. Shi, W. He, Q. Chen, L. Chen, W. Gu, T. Zhou, Denoising autoencoder, a deep learning algorithm, aids the identification of a novel molecular signature of lung adenocarcinoma, Genomics, proteomics & bioinformatics 18 (4) (2020) 468–480.

[9] H. Wu, P. Du, R. Kokate, J.-X. Wang, A semi-analytical solution and ai-based reconstruction algorithms for magnetic particle tracking, Plos one 16 (7) (2021) e0254051.

[10] Z. Long, Y. Lu, B. Dong, Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network, Journal of Computational Physics 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 Transactions on Neural Networks and Learning Systems 32 (9) (2020) 4166–4177.

[12] M. Corbetta, Application of sparse identification of nonlinear dynamics for physics-informed learning, in: 2020 IEEE Aerospace Conference, IEEE, 2020, pp. 1–8.

[13] Z. Chen, Y. Liu, H. Sun, Physics-informed learning of governing equations from scarce data, Nature Communications 12 (2021) 6136.

[14] F. Sun, Y. Liu, H. Sun, Physics-informed spline learning for nonlinear dynamics discovery, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), 2021, pp. 2054–2061.

[15] S. Zhang, G. Lin, Robust data-driven discovery of governing physical laws with error bars, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474 (2217) (2018) 20180305.

[16] S. Zhang, G. Lin, Robust data-driven discovery of governing physical laws using a new subsampling-based sparse bayesian method to tackle four challenges (large noise, outliers, data integration, and extrapolation), arXiv preprint arXiv:1907.07788 (2019).

[17] S. M. Hirsh, D. A. Barajas-Solano, J. N. Kutz, Sparsifying priors for bayesian uncertainty quantification in model discovery, arXiv preprint arXiv:2107.02107 (2021).

[18] M. E. Tipping, Sparse bayesian learning and the relevance vector machine, Journal of machine learning research 1 (Jun) (2001) 211–244.

[19] M. E. Tipping, A. C. Faul, et al., Fast marginal likelihood maximisation for sparse bayesian models., in: AISTATS, 2003.

[20] C. M. Bishop, M. Tipping, Variational relevance vector machines, arXiv preprint arXiv:1301.3838 (2013).

[21] A. C. Faul, M. E. Tipping, A variational approach to robust regression, in: International Conference on Artificial Neural Networks, Springer, 2001, pp. 95–102.

[22] A. C. Faul, M. E. Tipping, Analysis of sparse bayesian learning, in: Advances in neural information processing systems, 2002, pp. 383–389.

[23] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, A. G. Wilson, A simple baseline for bayesian uncertainty in deep learning, Advances in Neural Information Processing Systems 32 (2019).

[24] M. Quade, M. Abel, J. Nathan Kutz, S. L. Brunton, Sparse identification of nonlinear dynamics for rapid model recovery, Chaos: An Interdisciplinary Journal of Nonlinear Science 28 (6) (2018) 063116.

[25] K. Champion, B. Lusch, J. N. Kutz, S. L. Brunton, Data-driven discovery of coordinates and governing equations, Proceedings of the National Academy of Sciences 116 (45) (2019) 22445–22451.

[26] K. P. Champion, S. L. Brunton, J. N. Kutz, Discovery of nonlinear multiscale systems: Sampling strategies and embeddings, SIAM Journal on Applied Dynamical Systems 18 (1) (2019) 312–333.

[27] L. Zhang, H. Schaeffer, On the convergence of the sindy algorithm, Multiscale Modeling & Simulation 17 (3) (2019) 948–972.

[28] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, R. Ng, Fourier features let networks learn high frequency functions in low dimensional domains, Advances in Neural Information Processing Systems 33 (2020) 7537–7547.

[29] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[30] G.-J. Both, S. Choudhury, P. Sens, R. Kusters, Deepmod: Deep learning for model discovery in noisy data, Journal of Computational Physics 428 (2021) 109985.

[31] N. Wandel, M. Weinmann, M. Neidlin, R. Klein, Spline-pinn: Approaching pdes without data using fast, physics-informed hermite-spline cnns, arXiv preprint arXiv:2109.07143 (2021).

[32] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural network, in: International conference on machine learning, PMLR, 2015, pp. 1613–1622.

[33] Q. Liu, D. Wang, Stein variational gradient descent: A general purpose bayesian inference algorithm, Advances in neural information processing systems 29 (2016).

[34] D. M. Blei, A. Kucukelbir, J. D. McAuliffe, Variational inference: A review for statisticians, Journal of the American statistical Association 112 (518) (2017) 859–877.

[35] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, PMLR, 2016, pp. 1050–1059.

[36] S. Ebrahimi, M. Elhoseiny, T. Darrell, M. Rohrbach, Uncertainty-guided continual learning with bayesian neural networks, in: International Conference on Learning Representations, 2020.

[37] H. Ritter, A. Botev, D. Barber, A scalable laplace approximation for neural networks, in: 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings, Vol. 6, International Conference on Representation Learning, 2018.

[38] A. G. Wilson, P. Izmailov, Bayesian deep learning and a probabilistic perspective of generalization, Advances in neural information processing systems 33 (2020) 4697–4708.

[39] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al., A review of uncertainty quantification in deep learning: Techniques, applications and challenges, Information Fusion 76 (2021) 243–297.

[40] T. Frerix, D. Kochkov, J. Smith, D. Cremers, M. Brenner, S. Hoyer, Variational data assimilation with a learned inverse observation operator, in: International Conference on Machine Learning, PMLR, 2021, pp. 3449–3458.

[41] P. Grönquist, C. Yao, T. Ben-Nun, N. Dryden, P. Dueben, S. Li, T. Hoefler, Deep learning for post-processing ensemble weather forecasts, Philosophical Transactions of the Royal Society A 379 (2194) (2021) 20200092.

[42] J. McCabe, J. Brown, Learning to assimilate in chaotic dynamical systems, Advances in Neural Information Processing Systems 34 (2021) 12237–12250.

[43] M. Fey, J. E. Lenssen, F. Weichert, H. Müller, Splinecnn: Fast geometric deep learning with continuous b-spline kernels, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 869–877.

[44] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, A. G. Wilson, Subspace inference for bayesian deep learning, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 1169–1179.

[45] M. Tipping, The relevance vector machine, Advances in neural information processing systems 12 (1999).

[46] S. Wang, P. Perdikaris, Long-time integration of parametric evolution equations with physics-informed deeponets, arXiv preprint arXiv:2106.05384 (2021).

[47] P. Laube, M. O. Franz, G. Umlauf, Deep learning parametrization for b-spline curve approximation, in: 2018 International Conference on 3D Vision (3DV), IEEE, 2018, pp. 691–699.

[48] C. Durkan, A. Bekasov, I. Murray, G. Papamakarios, Neural spline flows, Advances in neural information processing systems 32 (2019).

[49] C. de Boor, On the convergence of odd-degree spline interpolation, Journal of approximation theory 1 (4) (1968) 452–463.

[50] B. Swartz, $o(h^{2n+2-l})$ bounds on some spline interpolation errors, Bulletin of the American Mathematical Society 74 (6) (1968) 1072–1078.

[51] J. Z. Huang, C. O. Wu, L. Zhou, Polynomial spline estimation and inference for varying coefficient models with longitudinal data, Statistica Sinica (2004) 763–788.

[52] C. De Boor, C. De Boor, A practical guide to splines, Vol. 27, springer-verlag New York, 1978.

Checklist

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3
   (b) Did you include complete proofs of all theoretical results? [Yes] See supplemental materials
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] It is included in the supplemental materials
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplemental materials
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 4 and supplemental materials
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental materials
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] It is general open source code available in public for research purpose
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Proof: analytical form of spline derivatives

Theorem 1. . *The derivative of B-spline curve can be analytically evaluated as:*

$$\dot{y}(t) = \Sigma_{s=0}^{r+k-1} \dot{N}_{s,k}(t)\theta_s, \tag{20}$$

where

$$\dot{N}_{s,k}(t) = \frac{k}{\tau_{s+k} - \tau_s} N_{s,k-1}(t) - \frac{k}{\tau_{s+k+1} - \tau_{s+1}} N_{s+1,k-1}(t). \tag{21}$$

Proof. We will prove by the induction. For the base case $k = 1$, we have

$$\dot{N}_{s,1}(t) = \begin{cases} \frac{1}{\tau_{s+1}-\tau_s} & t \in [\tau_s, \tau_{s+1}] \\ -\frac{1}{\tau_{s+2}-\tau_{s+1}} & t \in [\tau_{s+1}, \tau_{s+2}] \end{cases} = \frac{1}{\tau_{s+1} - \tau_s} N_{s,0}(t) - \frac{1}{\tau_{s+2} - \tau_{s+1}} N_{s+1,0}(t). \tag{22}$$

Let's suppose that the formula holds for k up to n. We will prove that this formula also holds for $k = n + 1$. By the definition in Eq. 4 and the chain rule, we can get that:

$$\begin{aligned}
\dot{N}_{s,n+1}(t) &= \frac{t - \tau_s}{\tau_{s+n+1} - \tau_s} \dot{N}_{s,n}(t) + \frac{N_{s,n}(t)}{\tau_{s+n+1} - \tau_s} + \frac{\tau_{s+n+2} - t}{\tau_{s+n+2} - \tau_{s+1}} \dot{N}_{s+1,n}(t) - \frac{N_{s+1,n}(t)}{\tau_{s+n+2} - \tau_{s+1}} \\
&= \frac{t - \tau_s}{\tau_{s+n+1} - \tau_s} \left(\frac{n}{\tau_{s+n} - \tau_s} N_{s,n-1}(t) - \frac{n}{\tau_{s+n+1} - \tau_{s+1}} N_{s+1,n-1}(t) \right) \\
&+ \frac{\tau_{s+n+2} - t}{\tau_{s+n+2} - \tau_{s+1}} \left(\frac{n}{\tau_{s+n+1} - \tau_{s+1}} N_{s+1,n-1}(t) - \frac{n}{\tau_{s+n+2} - \tau_{s+2}} N_{s+2,n-1}(t) \right) \\
&+ \frac{1}{\tau_{s+n+1} - \tau_s} N_{s,n}(t) - \frac{1}{\tau_{s+n+2} - \tau_{s+1}} N_{s+1,n}(t) \\
&= \frac{n+1}{\tau_{s+n+1} - \tau_s} N_{s,n}(t) - \frac{n+1}{\tau_{s+n+2} - \tau_{s+1}} N_{s+1,n}(t).
\end{aligned} \tag{23}$$

□

A.2 Spline representation

In this section, we give error bounds for spline representation. For simplicity, we consider 1D scenario and assume the target function $u : [0, 1] \to R$ is periodic and defined on the unit interval $\Omega = [0, 1]$. Consider a set of uniform knots $\Gamma : 0 = \tau_0 \leq \tau_1 \leq \cdots \leq \tau_{r+k} = 1$ with . The space of kth degree $\{N_{s,k}\}_{s=0}^{r+k-1}$ splines is

$$S^k(\Omega, \Gamma) = \{p | p(t) \text{ is a polynomial of degree } k \text{ in each } (\tau_i, \tau_{i+1})\} \cap \mathcal{C}^{k-1}(\Omega).$$

Spline interpolation seeks $\hat{u} \in S^k(\Omega, \Gamma)$ that satisfies $u(\tau_i) = \hat{u}(\tau_i) \quad \forall \quad 0 \leq r + k$.

Theorem 2 (Spline interpolation error bounds [49, 50])**.** *Assume that u is periodic, for odd degree spline interpolation k, we have*

$$\begin{aligned}
&\|u - \hat{u}\|_\infty = \Delta\tau^{k+1} \Big(C_{k+1} \|u^{(k+1)}\|_\infty + \mathcal{O}(\omega(u^{(k+1)}, \Delta\tau)) \Big), \\
&\|u^{(l)} - \hat{u}^{(l)}\|_\infty = \Delta\tau^{k+1-l} \Big(D_{k+1-l} \|u^{(k+1)}\|_\infty + \mathcal{O}(\omega(u^{(k+1)}, \Delta\tau)) \Big) \quad \forall 1 \leq l \leq k - 1.
\end{aligned} \tag{24}$$

Here C_{k+1} and D_{k+1-l} are constant parameters, which are independent of $\Delta\tau$ and u, and $\omega(u^{(k+1)}, \Delta\tau) = \sup_{|x-y| \leq \Delta\tau} |u^{(k+1)}(x) - u^{(k+1)}(y)|$.

In the present work, we focus on using spline for smoothing noisy data. It is essentially a nonparameteric estimation of function u from noisy data $\{t_i, \tilde{u}_i\}_{i=1}^n$,

$$\tilde{u}_i = u(t_i) + \eta_i \quad \text{with} \quad \eta_i \sim \mathcal{N}(0, \delta_\eta^2).$$

We further assume that $\{t_i\}$ are uniformly sampled from Ω. The unregularized smoothing process estimates spline basis coefficients

$$\hat{\boldsymbol{\theta}} = (\mathbf{N}_m^T \mathbf{N}_m)^{-1} \mathbf{N}_m^T \cdot \tilde{\mathbf{U}}$$

through minimizing $\|\tilde{\mathbf{U}} - \mathbf{N}_m \cdot \boldsymbol{\theta}\|$. Here $\tilde{\mathbf{U}} = \left[\tilde{u}_1, \tilde{u}_2, \cdots, \tilde{u}_n\right]^T$ and $\mathbf{N}_m$ is the spline basis matrix evaluated at these measurement locations. The optimal $\boldsymbol{\theta}^{\text{opt}}$ satisfies $\hat{u}(t) = \mathbf{N}(t) \cdot \boldsymbol{\theta}^{\text{opt}}$, where $\hat{u}$ is the spline interpolation of u, and $\mathbf{N}(t) = [N_{0,k}(t), N_{1,k}(t), \cdots, N_{r+k-1,k}(t)]^T$ denotes spline basis function vector. Following [51], we have spline fitting error bounds, as following.

Theorem 3 (Spline fitting error bounds)**.** *Assume that u is periodic and the number of data n is sufficient large, for odd degree spline interpolation k, we have*

$$\|\mathbb{E}\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^{\text{opt}}\|_2 \leq C_1 \Delta\tau^{k+1} \qquad \|\text{Cov}\hat{\boldsymbol{\theta}}\|_2 \leq C_2 \frac{\delta_\eta^2}{n},$$

where C_1 and C_2 are constant and independent of n.

Proof. Let us denote

$$\mathbf{U} = [u(t_1), u(t_2), \cdots, u(t_n)]^T \quad \hat{\mathbf{U}} = [\hat{u}(t_1), \hat{u}(t_2), \cdots, \hat{u}(t_n)]^T$$
$$\boldsymbol{\eta} = [\eta_1, \eta_2, \cdots, \eta_n]^T \quad \mathbf{e} = \mathbf{U} - \hat{\mathbf{U}}.$$

We have

$$\begin{aligned} \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^{opt} &= (\mathbf{N}_m^T \mathbf{N}_m)^{-1} \mathbf{N}_m^T \cdot \tilde{\mathbf{U}} - (\mathbf{N}_m^T \mathbf{N}_m)^{-1} \mathbf{N}_m^T \mathbf{N}_m \boldsymbol{\theta}^{\text{opt}} \\ &= (\mathbf{N}_m^T \mathbf{N}_m)^{-1} \mathbf{N}_m^T \cdot (\mathbf{U} + \boldsymbol{\eta} - \hat{\mathbf{U}}) \\ &= (\mathbf{N}_m^T \mathbf{N}_m)^{-1} \mathbf{N}_m^T \cdot (\mathbf{e} + \boldsymbol{\eta}). \end{aligned} \tag{25}$$

We will first prove that

$$\|\mathbf{N}_m^T \mathbf{N}_m\|_2 = \mathcal{O}(n) \qquad \|(\mathbf{N}_m^T \mathbf{N}_m)^{-1}\|_2 = \mathcal{O}(\frac{1}{n}). \tag{26}$$

For any $\boldsymbol{\theta}$, $\frac{1}{n}\boldsymbol{\theta}^T \mathbf{N}_m^T \mathbf{N}_m \boldsymbol{\theta}$ is the Monte Carlo approximation of $\int \left(\mathbf{N}(t)^T \cdot \boldsymbol{\theta}\right)^2 dt$, and hence

$$\frac{1}{n}\boldsymbol{\theta}^T \mathbf{N}_m^T \mathbf{N}_m \boldsymbol{\theta} = \int (\mathbf{N}(t)^T \boldsymbol{\theta})^2 dt + \mathcal{O}(\frac{1}{\sqrt{n}}). \tag{27}$$

Bringing the following property of B-splines [52]

$$M_1 \boldsymbol{\theta}^T \boldsymbol{\theta} \leq \int (\mathbf{N}(t)^T \boldsymbol{\theta})^2 dt \leq M_2 \boldsymbol{\theta}^T \boldsymbol{\theta} \qquad \exists M_1, M_2 > 0$$

into Eq. (27) leads to Eq. (26). Then we prove that

$$\|\mathbf{N}_m^T \mathbf{e}\|_2 = \mathcal{O}(n\Delta\tau^{k+1}). \tag{28}$$

Since $\sum_s N_{s,k}(t) = 1$ and $\|\mathbf{e}\|_\infty = \mathcal{O}(\Delta\tau^{k+1})$, we have

$$\|\mathbf{N}_m^T \mathbf{e}\|_2 \leq \|\mathbf{N}_m^T \mathbf{e}\|_1 = \mathcal{O}(n\Delta\tau^{k+1}).$$

Finally, combining Eq. (25), Eq. (26) and Eq. (28) leads to

$$\|\mathbb{E}\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^{\text{opt}}\|_2 = \|(\mathbf{N}_m^T \mathbf{N}_m)^{-1} \mathbf{N}_m^T \cdot \mathbf{e}\|_2 \leq \|(\mathbf{N}_m^T \mathbf{N}_m)^{-1}\|_2 \|\mathbf{N}_m^T \cdot \mathbf{e}\|_2 \leq C_1 \Delta\tau^{k+1}$$
$$\|\text{Cov}\hat{\boldsymbol{\theta}}\|_2 = \sigma_\eta^2 \|(\mathbf{N}_m^T \mathbf{N}_m)^{-1}\|_2 \leq C_2 \frac{\delta_\eta^2}{n}$$

□

In our sparse Bayesian regression, in stead of solving the aforementioned minimization problem, we have additional regularization terms.

A.3 Algorithms

In this section, we present detailed algorithms used in the present work, which include Bayesian Alternative Direction Optimization (ADO) Learning 1, Sequential Threshold Sparse Bayesian Learning 2, and Ensemble Kalman Filter 3.

Algorithm 1: Bayesian Alternative Direction Optimization (ADO) Learning

Input : Library $\mathbf{\Phi}$, spline basis $\mathbf{N}$, time derivative of spline basis matrix $\dot{\mathbf{N}}_c$, negative log form of equation Eq. 11 $\mathcal{L}$

Output : Mean estimation: $\boldsymbol{\theta}_{\text{SWA}}$, $\mathbf{W}_{\text{SWA}}$, $\mathbf{B}_{\text{SWA}}$, $\mathbf{P}_{\text{SWA}}$
Samples from posterior distributions: $\mathring{\boldsymbol{\theta}}$, $\mathring{\mathbf{W}}$, $\mathring{\mathbf{B}}$, $\mathring{\mathbf{P}}$

Pretrain:

for $i = 1 : T_{\text{Pretrain}}$ **do**

  SDG optimization with the fixed library

  $\{\boldsymbol{\theta}_{i+1}, \mathbf{W}'_{i+1}, \mathbf{B}_{i+1}, \mathbf{P}'_{i+1}\} = \arg\min \mathcal{L}$

  Update library

  $\mathbf{W}_{i+1}, \mathbf{P}_{i+1} = \text{STSparseBayesian}(\mathbf{\Phi}, \dot{\mathbf{U}}_{i+1} = \dot{\mathbf{N}}_c \boldsymbol{\theta}_{i+1}, \mathbf{W}'_{i+1}, \mathbf{P}'_{i+1})$

  if $\mathcal{L}(\boldsymbol{\theta}_{i+1}, \mathbf{W}_{i+1}, \mathbf{B}_{i+1}, \mathbf{P}_{i+1}) < \mathcal{L}^\star$ **then**

    $\mathcal{L}^\star = \mathcal{L}(\boldsymbol{\theta}_{i+1}, \mathbf{W}_{i+1}, \mathbf{B}_{i+1}, \mathbf{P}_{i+1})$

    $\boldsymbol{\theta}^\star, \mathbf{W}^\star, \mathbf{B}^\star, \mathbf{P}^\star = \boldsymbol{\theta}_{i+1}, \mathbf{W}_{i+1}, \mathbf{B}_{i+1}, \mathbf{P}_{i+1}$

  else

    break

end

Stochastic Weight Averaging-Gaussian (SWAG) for posterior approximation:

$\boldsymbol{\theta}_{\text{SWA}}, \mathbf{W}_{\text{SWA}}, \mathbf{B}_{\text{SWA}}, \mathbf{P}_{\text{SWA}} = \boldsymbol{\theta}^\star, \mathbf{W}^\star, \mathbf{B}^\star, \mathbf{P}^\star$

With a constant learning rate **for** $i = 1 : T_{\text{SWAG}}$ **do**

  SGD update $\boldsymbol{\theta}_i, \mathbf{W}_i, \mathbf{B}_i, \mathbf{P}_i$

  $\boldsymbol{\theta}_{\text{SWA}}, \mathbf{W}_{\text{SWA}}, \mathbf{B}_{\text{SWA}}, \mathbf{P}_{\text{SWA}} = \frac{i\boldsymbol{\theta}_{\text{SWA}} + \boldsymbol{\theta}_i}{i+1}, \frac{i\boldsymbol{W}_{\text{SWA}} + \boldsymbol{W}_i}{i+1}, \frac{i\boldsymbol{B}_{\text{SWA}} + \boldsymbol{B}_i}{i+1}, \frac{i\boldsymbol{P}_{\text{SWA}} + \boldsymbol{P}_i}{i+1}$

end

Compute low-rank square root of empirical covariance matrices $\mathbf{\Lambda}_\theta, \mathbf{\Lambda}_W, \mathbf{\Lambda}_B, \mathbf{\Lambda}_P$ from $\{\boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{SWA}}\}, \{\boldsymbol{W}_i - \boldsymbol{W}_{\text{SWA}}\}, \{\boldsymbol{B}_i - \boldsymbol{B}_{\text{SWA}}\}, \{\boldsymbol{P}_i - \boldsymbol{P}_{\text{SWA}}\}$

Sampling:

$\mathring{\boldsymbol{\theta}} = \boldsymbol{\theta}_{\text{SWA}} + \mathbf{\Lambda}_\theta \mathring{\boldsymbol{z}}_\theta \qquad \mathring{\boldsymbol{W}} = \boldsymbol{W}_{\text{SWA}} + \mathbf{\Lambda}_W \mathring{\boldsymbol{z}}_W \qquad \mathring{\boldsymbol{B}} = \boldsymbol{B}_{\text{SWA}} + \mathbf{\Lambda}_B \mathring{\boldsymbol{z}}_B \qquad \mathring{\boldsymbol{P}} = \boldsymbol{P}_{\text{SWA}} + \mathbf{\Lambda}_P \mathring{\boldsymbol{z}}_P$

where $\mathring{\boldsymbol{z}}_*$ are the random samples from $\mathcal{N}(0, I)$.

Algorithm 2: Sequential Threshold Sparse Bayesian Learning

Input : Spline trainable parameter $\boldsymbol{\theta}$, library $\mathbf{\Phi}(\boldsymbol{\theta})$, approximated derivative $\dot{\mathbf{U}}$, library weight $\mathbf{W}$, and process error matrix $\mathbf{P}$

Output : Best solution library $\mathbf{\Phi}^\star$,$\mathbf{W}^\star$, $\mathbf{P}^\star$

Initialize : Threshold ϵ, number of library terms p_{old}, and Flag = $\mathbf{True}$

while *Flag is True* **do**

  while *not converged* **do**

    1. Compute the relevance variable $\eta_i = {q_i}^2 - s_i$ as defined in [19]

    2. Update library $\mathbf{\Phi}^\star$, weight $\mathbf{W}^\star$, and process error matrix $\mathbf{P}^\star$, as shown in [19]

  end

  for $j = 1 : p_{old}$ **do**

    $\mathbf{W}^\star(j) = 0$ **If** $|\mathbf{W}^\star(j)| \leq \epsilon$

  end

  Find the nonzero entries in $\mathbf{W}^\star$, record the index as $\mathbf{I}$, update $\mathbf{\Phi} = \mathbf{\Phi}(:, \mathbf{I})$ and p_{new} = length of $\mathbf{I}$;

  if $\mathbf{p_{new}} = \mathbf{p_{old}}$ **then**

    Flag = $\mathbf{False}$

end

Algorithm 3: Ensemble Kalman Filter

Input : ensemble number J, sampled weights $\{\mathbf{W}^j\}_{j=1}^J$, discovered dynamical model $\mathbf{M}(\bullet\,;\ \mathbf{W})$, process noise covariance $\mathbf{P}$, observation model $\mathbf{h}$, observation $\tilde{\mathbf{U}}$, observation noise covariance $\mathbf{B}$

Output : Analysis ensemble trajectories $\mathbf{U}_a^j(t)$

Forecast : $\mathbf{U}_f^j(t_{i+1}) = \mathbf{M}(\mathbf{U}_a^j(t_i); \mathbf{W}^j) + \boldsymbol{\epsilon}_2^j, \quad \boldsymbol{\epsilon}_2^j \sim (\mathbf{0}, \mathbf{P})$

$\overline{\mathbf{U}}_f(t_{i+1}) = \frac{1}{J}\sum_{j=1}^J \mathbf{U}_f^j(t_{i+1})$

Analysis : $\mathbf{U}_h^j(t_{i+1}) = \mathbf{h}(\mathbf{U}_f^j(t_{i+1})) \qquad \overline{\mathbf{U}}_h(t_{i+1}) = \frac{1}{J}\sum_{j=1}^J \mathbf{U}_h^j(t_{i+1})$

$\mathbf{C}^{fh}(t_{i+1}) = \frac{1}{J-1}\Sigma_{j=1}^J(\mathbf{U}_f^i(t_{i+1}) - \overline{\mathbf{U}}_f(t_{i+1}))(\mathbf{U}_h^i(t_{i+1}) - \overline{\mathbf{U}}_h(t_{i+1}))^T$

$\mathbf{C}^{hh}(t_{i+1}) = \frac{1}{J-1}\Sigma_{j=1}^J(\mathbf{U}_h^i(t_{i+1}) - \overline{\mathbf{U}}_h(t_{i+1}))(\mathbf{U}_h^i(t_{i+1}) - \overline{\mathbf{U}}_h(t_{i+1}))^T + \mathbf{B}$

$\mathbf{K}(t_{i+1}) = \mathbf{C}^{fh}\left(\mathbf{C}^{hh}\right)^{-1}$

$\mathbf{U}_a^j(t_{i+1}) = \mathbf{U}_f^j(t_{i+1}) + \mathbf{K}(t_{i+1})\left(\tilde{\mathbf{U}}(t_{i+1}) - \mathbf{U}_h^j(t_{i+1}) - \boldsymbol{\epsilon}_1^j\right), \quad \boldsymbol{\epsilon}_1^j \sim (\mathbf{0}, \mathbf{B})$

A.4 Training Details

Additional training hyper parameters used in Sec. 4 is shown in the Tab. 2.

Table 2: Training Details

Case	Van der Pol	Lorenz 96	Advection	Burgers'
ADO Iter	5	5	1	5
ADO Epoch	$20K$	$50K$	$20K$	$20K$
Post Epoch	$1K$	$65K$	$2K$	$0.5K$
SWAG Epoch	$1.5K$	$80K$	$0.5K$	$0.5K$
LR	1×10^{-2}	1×10^{-2}	1×10^{-2}	1×10^{-2}
SWAG LR	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}

A.5 Additional Result

We list additional discovery and UQ results in this section. Fig. 4 shows 4 distributions of the coefficients for Van der Pol system in red box and Lorenz 96 system in blue box. Fig. 5 shows additional UQ result for all the 6 state variables for Lorenz 96 system, incorporating EnKF algorithms. Lastly, Fig. 7 shows additional UQ results for advection results in red box and Burgers' results in blue box.

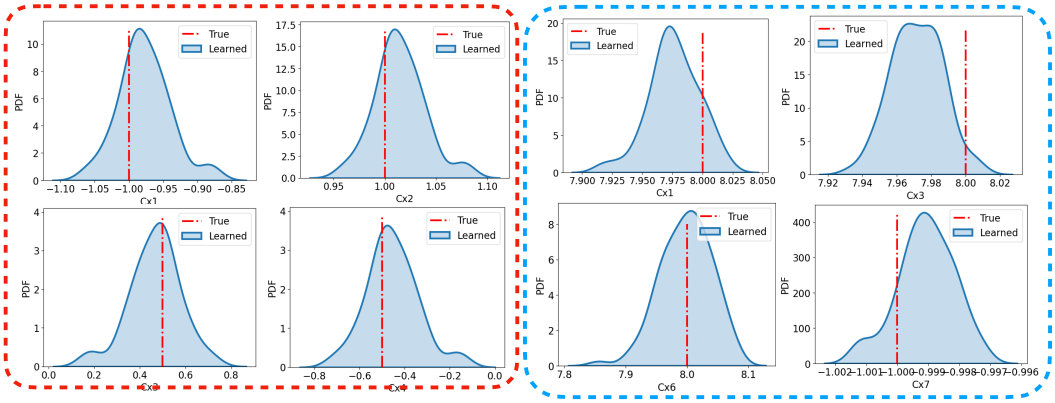

Figure 4: Additional discovery results for ODE systems; the red box shows results coefficients distribution for the Van der Pol system, and the blue box shows the result for Lorenz 96 system.

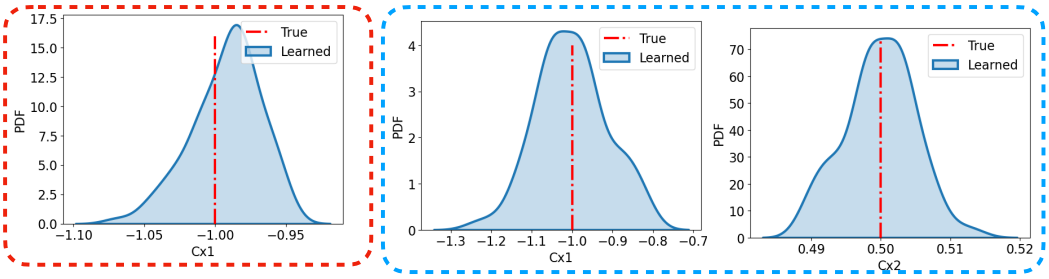

Figure 6: Additional discovery results for PDE systems; the red box shows results coefficients distribution for the advection equation, and the blue box shows the result for Burgers' equation.

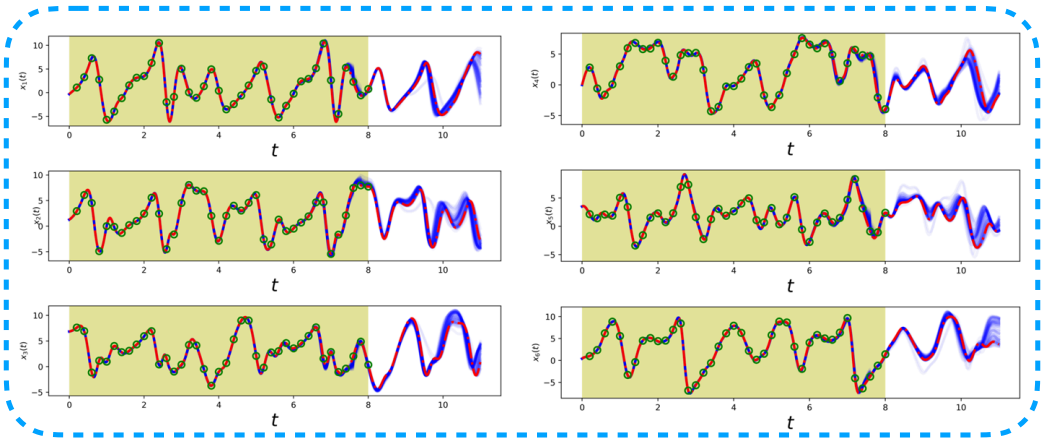

Figure 5: Additional UQ results for ODE systems; the blue box shows the all the states prediction for Lorenz 96 system with ensemble Kalman filter.

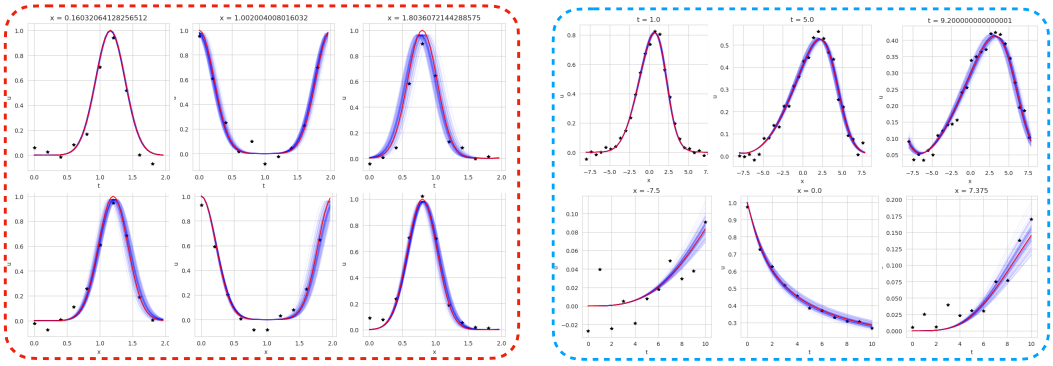

Figure 7: Additional UQ results PDE; the red box shows the cross section UQ results for Advection equation. The blue box shows the cross section UQ results for Burgers' equation.

A.6 Additional Discussion: PDE with source term

In this section, we chose to investigate the discovery performance when a source term exists in the governing equation. For example, we chose to discover Burgers' equation with a source term, with the form:

$$u_t + uu_x - 0.1u_{xx} = \sin(x)\sin(t) \tag{29}$$

To account for the source term, we add source terms to the library, as shown in Eq. 30

$$[\sin(x), \sin(t), \cos(x), \cos(t), \sin(x)\sin(t), \cos(x)\cos(t), \sin(x)\cos(t)] \tag{30}$$

The discovery result is with 20% Gaussian noise is shown in Tab. 3. The error is calculated by the same metric in Eq. 17 It can be seen that our model is more accurate than the PINN-SR model, with approximately the same number of training parameters and the training time. The SINDy algorithm failed at this large and cannot give the parsimonious form of the governing equation. Finally the contour plot and the cross section result are shown in Fig. 8.

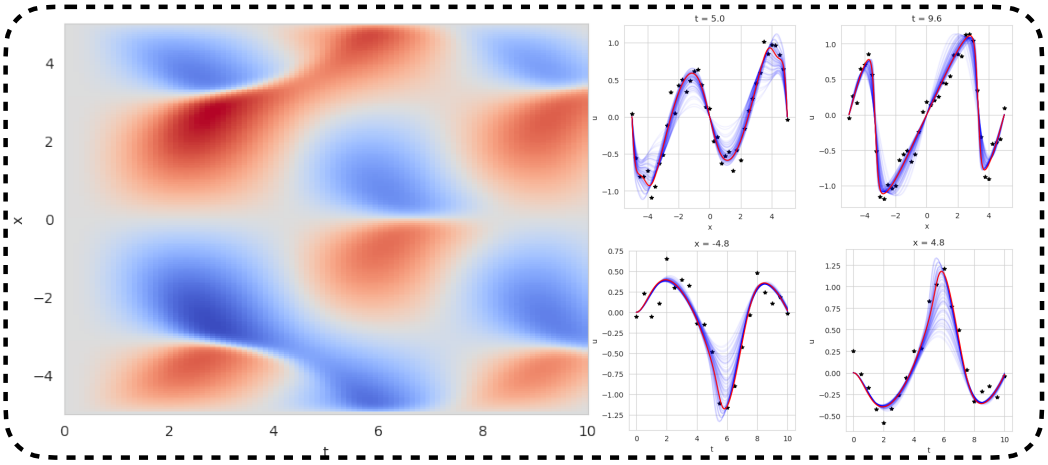

Figure 8: Additional UQ results for PDE; the black box shows the cross section UQ results for Burgers' equation with source.

A.7 Additional Discovery result

In this section, we list the full table that includes all the experiments made for current work, as attached in Tab. 3.

Table 3: ODE and PDE discovery comparison

Name	**rmse**(0%)	**rmse**(1%)	**rmse** (large[5])	$\mathbf{M_P}$	$\mathbf{M_R}$[6]	Training Cost[7]
			Van der Pol Oscillator			
BSL(Ours)	**0.2**	2.82	**18.04**	**1**	**1**	$\sim 133(+3)s$
PINN-SR	Fail[8]	Fail	Fail	0.214	0.75	$\sim 1213s$
SINDy	1.0	**1.93**	Fail	0.267	1.0	$\sim 10s$
RVM	1.0	2.54	27.46	**1**	**1**	$\sim 10s$
			Lorenz 96			
BSL(Ours)	**0.269**	1.47	**13.0**	**1**	**1**	$\sim 1654(+438)s$
PINN-SR	Fail	Fail	Fail	0.5	0.22	$\sim 10788s$
SINDy	0.4	0.64	Fail	0.75	**1**	$\sim 10s$
RVM	0.4	**0.6**	49.7	**1**	**1**	$\sim 25s$
			Advection Equation			
BSL(Ours)	**0.26**	**1**	**1.9**	**1**	**1**	$\sim 946(+233)s$
PINN-SR	5.9	4.5	30.4	**1**	**1**	$\sim 650s$
SINDy	2.3	8.2	38.9	**1**	**1**	$\sim 10s$
RVM	0.77	6.76	Fail	0.2	**1**	$\sim 4s$
			Burgers' Equation			
BSL(Ours)	3.62	4.13	**6.38**	**1**	**1**	$\sim 117(+74)s$
PINN-SR	10.2	**3.3**	10.3	**1**	**1**	$\sim 512s$
SINDy	0.826	Fail	Fail	**1**	0.5	$\sim 10s$
RVM	**0.754**	Fail	Fail	0.1429	0.5	$\sim 4s$
Name	**rmse**(0%)	**rmse**(0.1%)	**rmse** (large)	$\mathbf{M_P}$	$\mathbf{M_R}$	Training Cost
			Burgers' with Source			
BSL(Ours)	11	**12.4**	**13.4**	**1**	**1**	$\sim 396(+340)s$
PINN-SR	**10.5**	15	34.6	1	1	$\sim 600s$
SINDy	26.2	Fail	Fail	**1**	0.33	$\sim 10s$
RVM	27.6	Fail	Fail	0.5	0.67	$\sim 10s$
			Heat Equation			
BSL(Ours)	19	19	**38.9**	**1**	**1**	$\sim 71(+8)s$
PINN-SR	Fail	Fail	Fail	0	0	$\sim 285s$
SINDy	**1.9**	**17**	Fail	0.25	1	$\sim 10s$
RVM	2.8	6	Fail	0	0	$\sim 6s$
			Poisson Equation			
BSL(Ours)	$\mathbf{1.18 \times 10^{-2}}$	**0.133**	**16.7**	**1**	**1**	$\sim 92(+9)s$
PINN-SR	Fail	Fail	Fail	0	0	$\sim 3737s$
SINDy	1.15	87	962	**1**	**1**	$\sim 10s$
RVM	1.15	232	968	**1**	**1**	$\sim 10s$

[5]Large noise for different cases: Van der Pol: 5%, Lorenz 96: 10%, Advection: 20%, Burgers: 10%,Burgers' with source: 20%, Heat: 15%, Poisson: 5%

[6]$\mathbf{M_P}$, $\mathbf{M_R}$ are only reported for the largest noise cases

[7]All cases are running on a Nvidia 2070 Ti GPU card

[8]Fail means failure in discovery of the parsimonious ODE/PDE forms.

A.8 Analytical forms of the discovered system

Table 4: Analytical forms of ODE

Name	ϵ (large)
	Vander Pol Oscillator
True	$\frac{dx}{dt} = y, \frac{dy}{dt} = -x - 0.5x^2y + 0.5y$
BSL(Ours)	$\frac{dx}{dt} = 1.0096(\pm 0.024)y, \frac{dy}{dt} = -0.9858(\pm 0.037)x - 0.4801(\pm 0.114)x^2y + 0.4889(\pm 0.111)y$
PINN-SR	$\frac{dx}{dt} = 1.3079 - 0.1151x + 0.9982y + 0.1939x^2 - 0.4101xy + 0.8559y^2 - 0.3577x^3$ $+0.6764x^2y - 0.1207xy^2 + 0.6261y^3,$ $\frac{dy}{dt} = -0.6718x + 1.6035y + 1.6339y^2 + 0.4803y^3$
SINDy	$\frac{dx}{dt} = 0.1197 + 0.169x + 0.9975y - 0.0292x^2 + 0.0232xy - 0.0274y^2 - 0.0393x^3 + 0.0208x^2y - 0.0463xy^2,$ $\frac{dy}{dt} = -1.0909x + 0.149y + 0.0294x^3 - 0.4x^2y - 0.0356xy^2 + 0.0762y^3$
RVM	$\frac{dx}{dt} = 0.9957(\pm 0.073)y, \frac{dy}{dt} = -0.9943(\pm 0.0597)x - 0.4777(\pm 0.0809)x^2y + 0.4714(\pm 0.1204)y$
	Lorenz 96
True	$\frac{dX_1}{dt} = (X_2 - X_5)X_6 - X_1 + 8, \frac{dX_2}{dt} = (X_3 - X_6)X_1 - X_2 + 8,$ $\frac{dX_3}{dt} = (X_4 - X_1)X_2 - X_3 + 8, \frac{dX_4}{dt} = (X_5 - X_2)X_3 - X_4 + 8,$ $\frac{dX_5}{dt} = (X_6 - X_3)X_4 - X_5 + 8, \frac{dX_6}{dt} = (X_1 - X_4)X_5 - X_6 + 8$
BSL(Ours)	$\frac{dX_1}{dt} = 1.0033(\pm 2.37 \times 10^{-4})X_2X_6 - 0.9926(\pm 2.03 \times 10^{-4})X_5X_6,$ $-0.9991(\pm 9.04 \times 10^{-4})X_1 + 7.9773(\pm 2.08 \times 10^{-2}),$ $\frac{dX_2}{dt} = 0.9963(\pm 5.54 \times 10^{-5})X_1X_3 - 0.9942(\pm 2.47 \times 10^{-4})X_1X_6$ $-1.0054(\pm 2.31 \times 10^{-4})X_2 + 7.8719(\pm 7.05 \times 10^{-4}),$ $\frac{dX_3}{dt} = 1.0106(\pm 2.19 \times 10^{-4})X_2X_4 - 1.0029(\pm 2.03 \times 10^{-4})X_1X_2$ $-0.9979(\pm 5.3 \times 10^{-4})X_3 + 7.9938(\pm 1.48 \times 10^{-2}),$ $\frac{dX_4}{dt} = 1.0067(\pm 1.01 \times 10^{-4})X_3X_5 - 1.0103(\pm 9.09 \times 10^{-5})X_2X_3$ $-0.9926(\pm 2.36 \times 10^{-4})X_4 + 8.055(\pm 2.41 \times 10^{-3}),$ $\frac{dX_5}{dt} = 0.9953 \pm (4.39 \times 10^{-4})X_4X_6 - 0.9922 \pm (3.88 \times 10^{-4})X_3X_4$ $-1.0072 \pm (4.1 \times 10^{-4})X_5 + 7.8538(\pm 2.1 \times 10^{-3}),$ $\frac{dX_6}{dt} = 1.0095(\pm 2.71 \times 10^{-4})X_1X_5 - 0.9967(\pm 2.83 \times 10^{-4})X_4X_5$ $-0.9772(\pm 2.46 \times 10^{-4})X_6 + 7.9964(\pm 4.15 \times 10^{-2})$
PINN-SR	N/A
SINDy	$\frac{dX_1}{dt} = 8.0985 - 0.9423X_1 - 0.1007X_4 - 0.1659X_6 + 0.9582X_2X_6 - 0.944X_5X_6$ $\frac{dX_2}{dt} = 7.9372 - 0.1268X_1 - 0.9607X_2 + 0.9636X_1X_3 - 0.9558X_1X_6$ $\frac{dX_3}{dt} = 8.1523 - 0.0983X_1 - 0.1652X_2 - 0.9952X_3 - 0.9541X_1X_2 + 0.9651X_2X_4$ $\frac{dX_4}{dt} = 7.4958 - 0.9754X_4 + 0.1515X_5 - 0.973X_2X_3 + 0.9206X_3X_5,$ $\frac{dX_5}{dt} = 7.8556 - 0.1146X_4 - 0.9457X_5 - 0.9559X_3X_4 + 1.0023X_4X_6$ $\frac{dX_6}{dt} = 7.6026 + 0.0934X_2 - 0.927X_6 + 0.9711X_1X_5 - 0.9754X_4X_5$
RVM	$\frac{dX_1}{dt} = 0.9613(\pm 1.21)X_2X_6 - 0.962(\pm 1.27)X_5X_6,$ $-0.8983(\pm 2.67)X_1 + 7.518(\pm 5.88),$ $\frac{dX_2}{dt} = 0.9578(\pm 1.31)X_1X_3 - 0.9693(\pm 1.24)X_1X_6$ $-0.9217(\pm 2.8)X_2 + 7.6442(\pm 6.1575),$ $\frac{dX_3}{dt} = 0.9535(\pm 1.48)X_2X_4 - 0.9803(\pm 1.38)X_1X_2$ $-0.9468(\pm 2.96)X_3 + 7.5323(\pm 6.56),$ $\frac{dX_4}{dt} = 0.9365(\pm 1.24)X_3X_5 - 0.9748(\pm 1.15)X_2X_3$ $-0.9114(\pm 2.59)X_4 + 7.67(\pm 5.49),$ $\frac{dX_5}{dt} = 0.9981 \pm (1.26)X_4X_6 - 0.9667 \pm (1.2)X_3X_4$ $-0.9216 \pm (2.75)X_5 + 7.6146(\pm 5.96),$ $\frac{dX_6}{dt} = 0.9677(\pm 1.42)X_1X_5 - 0.9757(\pm 1.38)X_4X_5$ $-0.8986(\pm 2.86)X_6 + 7.6657(\pm 6.57)$

Table 5: Analytical forms of unsteady PDE

Name	ϵ (large)
	Advection Equation
True	$u_t = -u_x$
BSL(Ours)	$u_t = -0.9988(\pm 0.024)u_x$
PINN-SR	$u_t = -0.997u_x$
SINDy	$u_t = -0.9961u_x$
RVM	$u_t = -0.5148(\pm 0.106)u_x - 0.9797(\pm 0.384)uu_x$
	$-0.018(\pm 0.208)u^2u_x - 0.075(\pm 0.148)u^2u_{xxx} + 0.049(\pm 0.111)u^3u_{xxx}$
	Burgers' Equation
True	$u_t = -uu_x + 0.5u_{xx}$
BSL(Ours)	$u_t = -0.9929(\pm 0.086)uu_x + 0.4993(\pm 0.005)u_{xx}$
PINN-SR	$u_t = -1.0103uu_x + 0.5051u_{xx}$
SINDy	$u_t = -0.8179uu_x$
RVM	$-0.0809(\pm 0.041)u_x - 1.6684(\pm 0.2618)uu_x + 4.1835(\pm 0.571)u^2u_x$
	$-3.9068(\pm 0.391)u^3u_x + 0.1916(\pm 0.064)uu_{xx} - 1.0314(\pm 0.197)u^2u_{xx} + 1.5504(0.156)u^3u_{xx}$
	Burgers' Equation with Source
True	$u_t = -uu_x + 0.1u_{xx} + sin(x)sin(t)$
BSL(Ours)	$-0.9882(\pm 0.246)uu_x + 0.105(\pm 0.022)u_{xx} + 0.9859(\pm 0.005)sin(x)sin(t)$
PINN-SR	$u_t = -0.9576uu_x + 0.1168u_{xx} + 1.0179sin(x)sin(t)$
SINDy	$u_t = 0.8052sin(x)sin(t)$
RVM	$-0.0234(\pm 0.145)uu_x + 0.8318(\pm 0.142)sin(x)sin(t)$
	$-0.0789(\pm 0.105)sin(x) + 0.3558(\pm 0.156)sin(x)cos(t)$

Table 6: Analytical forms of steady PDE

Name	ϵ (large)
	Heat Equation
True	$u_{yy} = -u_{xx}$
BSL(Ours)	$u_{yy} = -0.9611(\pm 0.059)u_{xx}$
PINN-SR	$u_{yy} = 0.5544u_x$
SINDy	$u_{yy} = -0.069u_{xx} + 13.8988uu_x - 19.493u_x + 0.2468uu_{xx}$
RVM	$u_{yy} = -7.5861(\pm 202.6)u_x$
	Poisson Equation
True	$u_{yy} = -u_{xx} - sin(x)sin(y)$
BSL(Ours)	$u_{yy} = -0.9788(\pm 8.75 \times 10^{-4})u_{xx} - 0.9897(\pm 4.63 \times 10^{-4})sin(x)sin(y)$
PINN-SR	$u_{yy} = 0.13611752uu_x + 0.29748484uu_{xxx}$
SINDy	$u_{yy} = 0.2316u_{xx} - 0.4221sin(x)sin(y)$
RVM	$u_{yy} = 0.2284(\pm 0.007)u_{xx} - 0.3954(\pm 0.01)sin(x)sin(y)$

A.9 Implementation detail for the spline

In the current work, we apply direct tensor product to extend spline for solving spatial-temporal field and the relevant statistics are attached in Tab. 7, where the control points (trainable weights) θ in 1-d is marked by red and the total number of parameters are listed in the last column. We only store the non-zero elements for two-dimensional basis to leverage the sparsity (local support) of the spline. However, we must claim that it is *not* an optimal way to extend spline for higher spatial dimensions. In that case, a spline kernel can be defined and the tensor-product is only processed in the subdomain, as shown in [43].

Table 7: Direct tensor-product spline for PDE

basis x	basis $t(y)$	basis$(x, t(y))$	sparsity	trainable params
Advection Equation				
50×54	50×54	62500	0.0086	2916
Burgers' Equation				
128×13	101×19	92872	0.058	247
Burgers' Equation with Source				
201×103	101×103	251415	0.0012	10609
Heat Equation				
51×11	51×11	41209	0.1309	121
Poisson				
101×53	101×53	162409	0.0057	2809

A.10 Relevant Terminologies

Table 8: Terminologies

Name	Explanation
Control points	Trainable weight θ for spline basis.
Measurement points	Sparse spatio-temporal points with labels
Collocation points	Dense spatio-temporal points without labels
Library candidates	A collection of polynomial terms that the system identification algorithm can choose parsimonious terms from it. e.g., $\{x, y, x^2y, ...\}$ (for ODE) or $\{u, uu_x, u_{xx}...\}$ for (PDE)
ADO iteration	Alternating direction optimization to update the trainable parameters of control points θ, weight of library candidates $\mathbf{W}$ and covariance matrices.
Aleatoric Uncertainty	Due to intrinsic randomness by nature, which is irreducible
Epistemic Uncertainty	Because of a lack of knowledge, which can be reduced by adding more information.