Physics-informed machine learning for data-driven identification of governing equations

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2025

Pongpisit THANASUTIVES

List of Publication

The new technical contributions found in this thesis are compiled based on my papers listed in the following.

Journal

1. **Pongpisit Thanasutives**, Takashi Morita, Masayuki Numao, and Ken-ichi Fukui, Adaptive Uncertainty-Penalized Model Selection for Data-Driven PDE Discovery, IEEE Access, volume 12, pp. 13165-13182, January, 2024
2. **Pongpisit Thanasutives**, Takashi Morita, Masayuki Numao, and Ken-ichi Fukui, Noise-aware physics-informed machine learning for robust PDE discovery, Machine Learning: Science and Technology, volume 4(1), February, 2023

International conference and workshop

1. **Pongpisit Thanasutives**, Masayuki Numao, and Ken-ichi Fukui, Adversarial Multi-task Learning Enhanced Physics-informed Neural Networks for Solving Partial Differential Equations, International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, pp. 1-9, July, 2021
2. **Pongpisit Thanasutives** and Ken-ichi Fukui, Adaptation of uncertainty-penalized Bayesian information criterion for parametric partial differential equation discovery, Machine Learning and the Physical Sciences Workshop at the 38th conference on Neural Information Processing Systems (NeurIPS), December, 2024

Domestic conference presentation

1. **Pongpisit Thanasutives** and Ken-ichi Fukui, Uncertainty-penalized Bayesian information criterion for parametric partial differential equation discovery, 名大ISEE研究集会　「情報科学技術との融合による太陽圏物理学の新展開」, Nagoya, Japan, Sepetember, 2024
2. **Pongpisit Thanasutives**, Uncertainty-Penalized Bayesian Information Criterion for Data-Driven Partial Differential Equation Discovery, Machine Learning Summer School (MLSS), Okinawa, Japan, 2024

3. **Pongpisit Thanasutives**, Takashi Morita, Masayuki Numao, and Ken-ichi Fukui, Robust Data-driven PDE Discovery by Forward Best-subset Selection, The 25th Information-Based Induction Sciences Workshop (IBIS), 2022

4. **Pongpisit Thanasutives**, Masayuki Numao, and Ken-ichi Fukui, Learning to Solve Multiple Partial Differential Equations Using Physics-informed Neural Networks, 人工知能学会全国大会論文集, JSAI2021, p.4N4IS1c04-4N4IS1c04, 2021

5. **Pongpisit Thanasutives**, Masayuki Numao, and Ken-ichi Fukui, Adversarial Multi-task Learning Algorithm for Solving Partial Differential Equations, Japan Geoscience Union Meeting (JpGU), 2021

Abstract

Technological advancements over recent centuries have been profoundly influenced by scientific discoveries, ranging from Newton's laws and Navier-Stokes equations to the Black-Scholes model for financial markets. Discoveries in science are achieved in diverse formats using various approaches. This thesis focuses on data-driven methods for deriving certain types of differential equations, which are ubiquitous and interpretable symbolic representations that govern system dynamics in a wide range of fields. Data-driven methods are particularly compelling due to their flexibility and accuracy (over first-principles based methodology), as they can be tailored to fit data with minimal prior knowledge. However, some difficulties still remain in developing an automatic, robust data-driven approach for identifying governing differential equations. For example, optimizing for the true sparsity of system dynamics, which are determined by only a few key physical variables, becomes challenging particularly when the training data is of poor quality, as this naturally leads to inaccuracies in the computed features. Another critical aspect is controlling the size of search space. A large search space may yield an overcomplete set of candidate models but prolong optimization time. Conversely, while a smaller search space facilitates faster optimization of the best model, it risks being incomplete, increasing the possibility of missing the true governing equation. This thesis is crafted with the aim of addressing these problems and other intricate challenges, advancing the field of data-driven discovery and neural-based solutions for partial differential equations (PDEs) through three major contributions in the following.

First, we propose a noise-aware physics-informed machine learning framework that integrates physics-informed neural networks (PINNs) with sparse identification of nonlinear dynamics (SINDy) to discover governing PDEs from noisy measurement data. This framework, which is effective even without a discretized mesh, introduces an interpretable derivative preparation step where feature importance of candidate terms is learned prior to initial PDE identification, mitigating overfitting risks. By incorporating denoising PINNs that utilize projections of noise provided by fast Fourier transform (FFT), the framework enhances data quality for determining PDE coefficients more accurately.

Second, we develop a novel uncertainty-penalized Bayesian information criterion (UBIC) for reliably identifying true governing PDEs with high success rates. Unlike conventional criteria such as AIC (Akaike information criterion) and BIC (Bayesian information criterion), which are demonstrated to favor overly complex PDEs, UBIC leverages a penalty for uncertainty in parameter estimation alongside traditional model complexity, defined by the count of nonzero parameters. Bayesian regression is employed to quantify this uncertainty. Additionally, PINN-based PDE simulations are conducted to validate the UBIC-selected PDE against the other potential PDE. Numerical results demonstrate UBIC's superior performance in achieving robust PDE discovery.

Finally, we present an accurate neural network-based solver for nonlinear PDEs. By employing advanced multi-task training techniques (e.g., uncertainty-weighted loss and gradient surgery) as well as cross-stitch modules, the network effectively leverages shared representations across parameterized PDE families, resulting in improved generalization. Additionally, adversarial training is incorporated to focus on high-loss regions, enhancing the solver's performance in dynamically changing domain regions. Experimental validation demonstrates the solver's success in reducing error on unseen data, excelling in high-dimensional, stochastic, and nonlinear PDE scenarios, and outperforming existing approaches.

In summary, this thesis addresses key challenges in data-driven discovery and numerical solutions of differential equations—such as noise, sparsity, uncertainty, and generalization—by introducing innovations including noise-aware physics-informed machine learning, uncertainty-penalized information criteria, and enhanced neural solvers. These contributions improve upon state-of-the-art approaches and are anticipated to provide foundational tools for diverse scientific and engineering applications, paving the way for future breakthroughs in the automated discovery and solution of governing equations.

Acknowledgment

First and foremost, I would like to express my sincere gratitude to Professor Masayuki Numao for his warm welcome and the opportunity to be a part of his laboratory.

Second, I am deeply grateful to Associate Professor Ken-ichi Fukui, my advisor, for his tremendous support and guidance during my time under his supervision. His patience, encouragement, and advice made the completion of this thesis possible.

Third, I would like to thank Assistant Professor Takashi Morita for his valuable and inspiring feedback during my research presentations.

I am also grateful to all my friends at Osaka University for providing a wonderful atmosphere for research, helping me through difficult times, and offering any advice they have given me.

Lastly, I would like to thank my family for their unwavering support and encouragement throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Data-driven discovery of governing equations and symbolic regression

Scientific discovery has always been a cornerstone of the development of knowledge and technology. With the recent advancements in machine learning and deep learning, the process of scientific discovery has been driven more and more by data, facilitating the conventional practices of first principles and trial and error. Tailored to scientific data, symbolic regression algorithms [1] can extract meaningful insights or knowledge succinctly, offering interpretable models in theoretically arbitrary forms that can vary in complexity, ranging from simple mathematical expressions (see the example in Figure 1.1) found ubiquitously across various fields of physics to ordinary and partial differential equations (ODEs and PDEs) in engineering. There has always been a demand for these interpretable models. In the physical sciences, mathematical models that respect physics enable reasoning about natural phenomena in ways that blackbox models, like deep neural networks, cannot. In high-stakes fields like healthcare, non-interpretable models, regardless of their accuracy, may never be permitted for deployment [2].

Udrescu and Tegmark [3] demonstrated that their AI-Feynman symbolic regression algorithm, which combines neural network learning with techniques exploiting symmetry and separability information to accelerate a brute-force search of mathematical expressions, can accurately identify all 100 well-known equations from the *Feynman Lectures on Physics*. Similarly impressive results in data-driven discovery of governing ODEs and PDEs were also shown by [4], leading to the entirely new research field of SINDy—sparse identification of nonlinear dynamics—where the central assumption is that the governing equation consists of only a few dominant variables. These remarkable results have indeed brought more attention to the field of data-driven scientific discovery.

It is important to understand that equations can be represented by more than just tree-based expressions; scientific discovery can be achieved using a variety of methods applied to different data structures. Figure 1.2 offers a categorization of symbolic regression methods, though this is by no means an exhaustive list. For regression-based methods, governing expressions can be identified by solving systems of linear or nonlinear equations. When using tree-based ex-

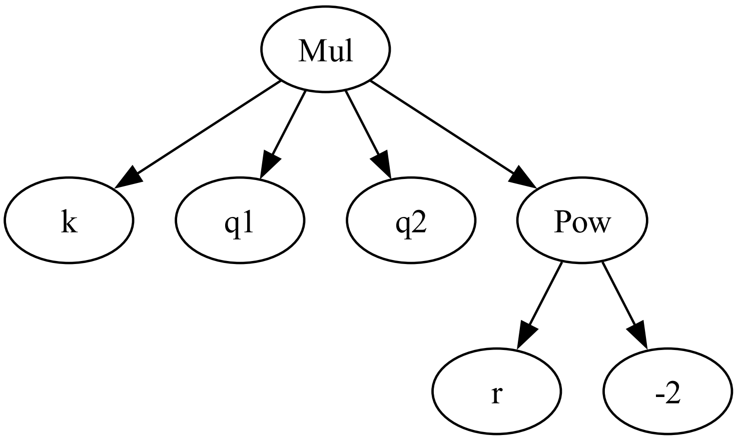


Figure 1.1: Example tree expression for the Coulomb force F between two charged particles (q_1 and q_2) with a constant k, where $F = k\frac{q_1 q_2}{r^2}$. r is the distance between the centers of the two charges. Mul and Pow stands for multiplication and power, respectively. Notes that a binary tree may be used, but the resulting height and number of leaves, which determine the tree's complexity, will change.

pressions, governing equations are discovered through approaches like genetic programming or reinforcement learning, where a powerful neural network, such as a transformer [5], can act as the agent.

1.2 SINDy: Sparse identification of nonlinear dynamics

Given snapshots of a dynamical system, sparse identification of nonlinear dynamics (SINDy) is a data-driven approach for identifying governing equations from the data, requiring minimal prior knowledge about the system. A vanilla SINDy leverages a sparsity-promoting regression, e.g., LASSO (the least absolute shrinkage and selection operator) [6], on a library of nonlinear candidate functions against temporal derivatives to find the governing equations presumably consisting of a few dominant variables that actually dictate the dynamics. Generally speaking, SINDy performs better when the snapshots are not noisy or can be measured accurately. Thus it is a commonly good practice to develop robust SINDy variants. To obtain the true governing equation without any dispensible candidate terms, model selection is a very important step in SINDy and symbolic regression methods in general. For model selection, what criterion that defines the balance between goodness-of-fitting and model complexity to be used and how to practically extract the optimal model out of the criterion is

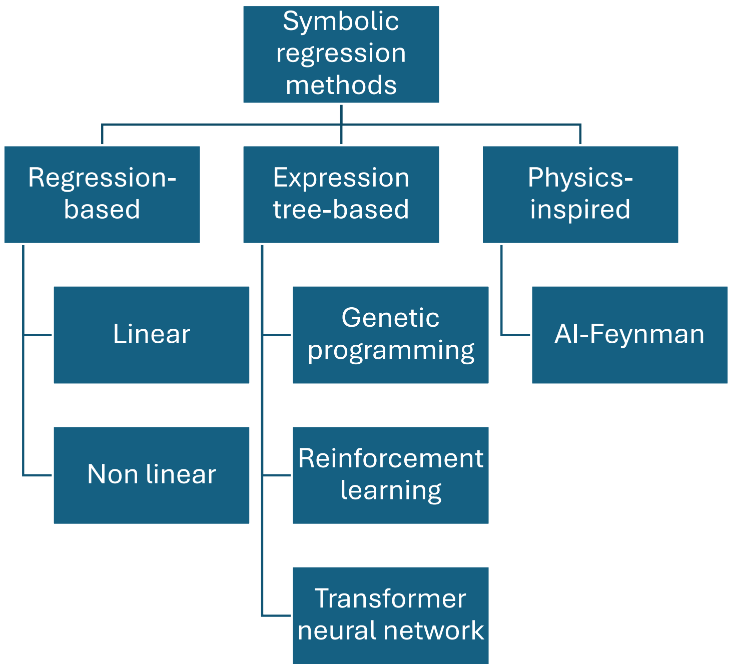


Figure 1.2: Comprehensive taxonomy of symbolic regression methods. Please see [1] for exemplar methods in each category.

very crucial. This is also true for evaluating symbolic regression methods.

1.3 Core challenges in data-driven discovery of governing equations

Noisy data. Generally, underlying dynamics are poorly captured by machine learning algorithms if the training data used to train them are of low quality. Noisy data negatively impacts the process of data-driven discovery, especially for differential equations, mainly because the features—the data and derivatives derived from it—are of poor quality.

Overfitting. Depending on the type of sparsity-promoting regression and model selection criterion used, there can be challenges in setting hyperparameters accurately to eventually select the true model. The issue of overfitting becomes more pronounced when the snapshot data is noisy, making it difficult to perform regression that captures the underlying dynamics with only a few true terms, without including other dispensable variables.

Missing of true variables. This problem is tied to machine learning and centers on the possibility that the *overcompleteness* assumption could be violated. This raises the question: how can we ensure the inclusion of correct, high-quality features as inputs to the algorithms? Without effective features, it is difficult to develop a good model. Enlarging the search space may alleviate this issue but often leads to prolonged training time. Additionally, using a large set of candidate terms is also challenging and may not yield the optimal model efficiently.

Miscellaneous Scientific equation discovery methods have often been designed for specific types of equations, such as ODEs, PDEs, or equations without derivatives as main features. Creating a generalized method that can handle a wide range of equations is challenging but valuable, as it would help scientists identify and understand the governing equations relevant to their research. Furthermore, developing guidelines on which definitions of sparsity to consider and which regularized loss functions work best for various symbolic regression methods would provide researchers with a clear path to extract insights from their data.

1.4 Sparsity-promoting regression

Sparsity-promoting regression optimizes for important terms that capture underlying dynamics using a regularized solver, such as STRidge (sequential threshold ridge) [7] or LASSO, depending on the objective function. This becomes even more difficult in noisy environments, where achieving a robust method is challenging. Accurately identifying these important terms is not an easy task (an NP-hard problem), as it requires searching over a large space of mathematical operations and determining the appropriate constants to best fit the data.

It is natural that early sparsity-promoting regression methods relied on regularized regression, developed to prevent overfitting—examples include Ridge [8] and LASSO. Ridge regression, also known as l_2 regularization, is one form of regularization used with linear regression models to reduce errors from overfitting on training data. Specifically, Ridge regression corrects multicollinearity among candidate terms, which is especially relevant when the terms are derivatives (particularly high-order derivatives). These derivatives often have small values, leading to large coefficients that should be regularized by Ridge penalty.

LASSO, also called regression with l_1 regularization, works by shrinking some coefficients to zero, effectively eliminating certain independent variables from the model. Both LASSO and Ridge regression reduce model complexity, but in different ways. LASSO simplifies the model by reducing the number of independent variables that influence the output, while Ridge regression reduces the influence (or weight/coefficient) each variable has on the output. As a result, Ridge regression itself alone cannot be used to achieve true sparsity. However, LASSO can be sensitive to its regularization hyperparameter, which can leave us unsure about the true governing form of the model. Since we also do not know in advance the number of active terms in a solution provided by LASSO, we may unfortunately miss the actual model complexity.

1.5 Information criterion to prevent overfitting in model selection

At the time of writing this thesis, no perfect information criterion exists that can determine the optimal complexity of the governing equation for every physical system. Researchers typically use data-splitting strategies like k-fold cross-validation or revert to common information criteria, such as the Akaike Information Criterion (AIC) [9] and Bayesian Information Criterion (BIC) [10], which are more suitable when data-splitting strategies are not affordable due to a limited amount of measurement data.

One issue with cross-validation is that the validation set may exhibit a slight distributional shift compared to the training set, raising concerns about whether the resulting distribution sufficiently supports a data-driven discovery process.

A major challenge in defining an information criterion lies in balancing model accuracy with model complexity to accurately identify the true governing equation. Excessive complexity may lead to overfitting, while insufficient complexity may fail to capture the essential structure of the data. Defining an appropriate measure of model complexity is crucial to guiding the selection of models that are both parsimonious and representative of the true dynamics.

Conventional criteria, such as AIC and BIC, struggle with the task of PDE discovery due to two fundamental issues.

Independence assumption AIC and BIC rely on the assumption that data points are independently and identically distributed (i.i.d.). However, data governed by PDEs inherently exhibit dependencies in space and time, violating the i.i.d. assumption.

Disproportionate data size The number of training data points significantly can exceed the number of terms in the PDE due to the dense sampling of spatio-temporal data points. This imbalance can lead to an overestimation of the likelihood in AIC/BIC, compromising their reliability for PDE model selection.

It is worth noting that these criteria are derived from information theory and are not specifically designed to identify sparse governing equations. Since information criteria depend on the candidate terms—which, in the context of this thesis, are mostly derivatives—these less common or unnatural time-series terms may contribute to the ineffectiveness of AIC and BIC.

1.6 New technical contributions

The novel technical contributions in this thesis are presented in 3 folds.

1. The first contribution is a noise-aware physics-informed machine learning framework [11] that combines physics-informed neural network (PINN) [12] learning with SINDy to discover the governing PDE from noisy measurement data, even when it does not originate from a discretized mesh. The derivative preparation step in this framework is interpretable, as the feature importance of candidate terms is learned by the neural network before the initial PDE identification step is applied, reducing the risk

of overfitting during neural network training. Additionally, the framework incorporates denoising PINNs, which use projected (initial) noise extracted via fast Fourier transform (FFT) to produce higher-quality data for determining PDE coefficients.

2. The second technical contribution is the uncertainty-penalized Bayesian information criterion (UBIC) [13, 14, 15]. This criterion is designed to identify the true governing equation with high success rates, without requiring simulations of all possible PDEs, thereby expediting the PDE discovery process. Conventional criteria like AIC and BIC, under similar circumstances, have been shown to favor overly complicated PDEs. To address this, UBIC adopts a more conservative approach by incorporating not only the count of nonzero parameters as a complexity penalty but also the uncertainty associated with them, referred to as PDE uncertainty. Bayesian regression is utilized to quantify this uncertainty. Additionally, we introduce a physics-informed neural network learning framework as a simulation-based method to further validate the UBIC-selected PDE against other potential candidates. Numerical results demonstrate the successful application of UBIC for data-driven PDE discovery from noisy spatio-temporal data.

3. The third contribution advances neural network-based solvers for PDEs addressing performance challenges in highly nonlinear domains. This is achieved through a new framework that integrates multi-task learning techniques—such as uncertainty-weighted loss [16], gradient surgery [17], and cross-stitch modules [18]—to exploit shared representations across related PDEs generated by varying parameterization coefficients, thereby enhancing generalization. Moreover, adversarial training is employed to create high-loss samples that mimic the original training data distribution, enabling the network to prioritize difficult regions with pronounced nonlinearity. Experimental results reveal that the framework's effectiveness, showing substantial error reduction on unseen data across diverse PDE scenarios, including high-dimensional and stochastic cases, and outperforming previous methods.

1.7 Thesis's organization

The rest of this thesis is structured as follows: Chapter 2 reviews foundational literature, including SINDy and its variations, such as PDE-FIND and evolutionary-based PDE discovery frameworks, alongside sparse regression techniques for developing parsimonious models. Since the success of these methods depends on data quality, we explore denoising techniques such as FFT and robust principal component analysis (RPCA) [19]. Additionally, we introduce information criteria, essential for selecting the optimal model from a collection of parsimonious options along the Pareto front. The chapter concludes with a discussion of PINNs and deep operator learning as alternatives for solving or simulating PDEs in a mesh-less manner.

Chapters 3 through 5 present our novel contributions to the field. Chapter 3 proposes a robust framework that integrates PINN learning with SINDy to facilitate PDE discovery from noisy data that may not originate from a discretized mesh. Chapter 4 introduces a new uncertainty-guided information criterion, designed to conservatively uphold the parsimony of governing equations by incorporating quantified uncertainty in PDEs. Chapter 5 explores improving the generalization of PINNs using advanced machine learning techniques, such as adversarial and multi-task learning. Finally, Chapter 6 summarizes the thesis and outlines directions for future research.

Chapter 2

Literature Review

2.1 SINDy and its variants

2.1.1 Mathematical overview of SINDy for ODEs

Here the problem of dynamical system discovery is formulated through the lens of sparse regression and compressed sensing. The central assumption is that most physical systems have only a few relevant terms that dictate the dynamics, making the governing equations sparse in a high-dimensional nonlinear function space. Let us consider dynamical systems in the following ODE form:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(\mathbf{x}(t)); \tag{2.1}$$

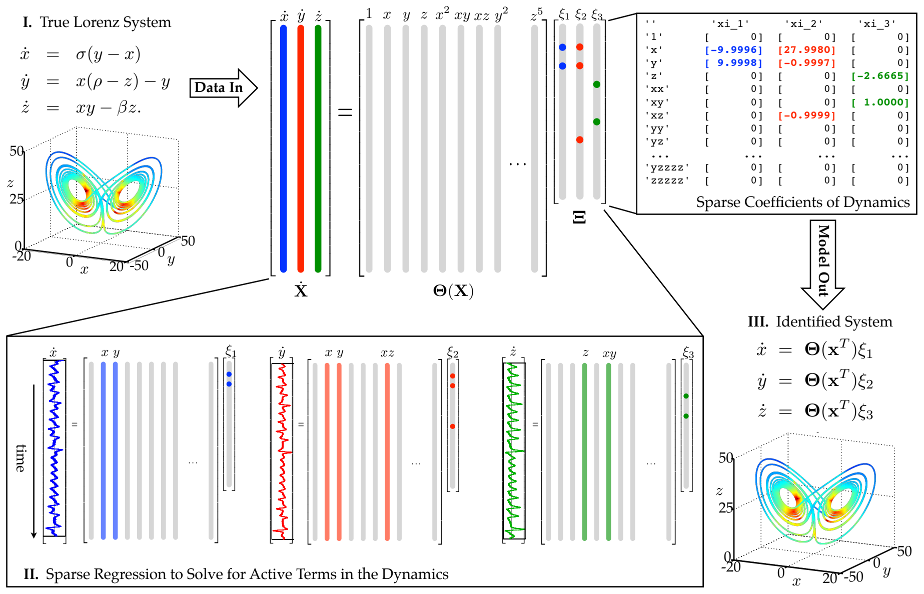
where the vector $\mathbf{x}(t)$ denotes the state of a system at time t, and $f(\mathbf{x}(t))$ is the dynamical constraint defining motion of the system. Throughout this thesis we often assume an access to some measurements of at least the state variables in space and time. The goal is to find the governing function f from the data:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{\intercal}(t_1) \\ \mathbf{x}^{\intercal}(t_2) \\ \vdots \\ \mathbf{x}^{\intercal}(t_m) \end{bmatrix}. \tag{2.2}$$

Based on $\mathbf{X}$ we can compute its corresponding time derivatives $\dot{\mathbf{X}}$ (e.g., using finite difference) and then construct a large candidate library $\mathbf{\Theta}(\mathbf{X})$ consisting nonlinear functions of $\mathbf{X}$, as comprehensively illustrated in Figure 2.1. There is tremendous freedom of choice in constructing the entries in this matrix of nonlinearities. Since we believe that only a few of these nonlinearities are active, we set up a sparse regression to determine the sparse vectors of coefficients as follows:

$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi}. \tag{2.3}$$

In order to solve for the sparse $\mathbf{\Xi}$, LASSO might yield a satisfying solution, however, it may not computationally efficient for very large datasets. Therefore,


I. True Lorenz System
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\rho - z) - y$
$\dot{z} = xy - \beta z.$
Data In
$\dot{x}$ $\dot{y}$ $\dot{z}$
1 x y z x^2 xy xz y^2 z^5
ξ_1 ξ_2 ξ_3
$\dot{\mathbf{X}}$
$\boldsymbol{\Theta}(\mathbf{X})$
$\boldsymbol{\Xi}$
Sparse Coefficients of Dynamics
Model Out
III. Identified System
$\dot{x} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_1$
$\dot{y} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_2$
$\dot{z} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_3$
time
II. Sparse Regression to Solve for Active Terms in the Dynamics

```
''          'xi_1'      'xi_2'      'xi_3'
'1'       [      0]  [      0]  [      0]
'x'       [-9.9996]  [27.9980]  [      0]
'y'       [ 9.9998]  [-0.9997]  [      0]
'z'       [      0]  [      0]  [-2.6665]
'xx'      [      0]  [      0]  [      0]
'xy'      [      0]  [      0]  [ 1.0000]
'xz'      [      0]  [-0.9999]  [      0]
'yy'      [      0]  [      0]  [      0]
'yz'      [      0]  [      0]  [      0]
 ...           ...        ...        ...
'yzzzz'   [      0]  [      0]  [      0]
'zzzzz'   [      0]  [      0]  [      0]
```

Figure 2.1: Schematic of SINDy by [4] demonstrated on the Lorenz system.

Brunton et al. [4] decided to use the sequential thresholded least-squares (STLS) algorithm in Code (2.1).

Code 2.1: Sequential thresholded least squares algorithm for sparse regression in Matlab

```matlab
% Compute Sparse regression: sequential least squares
% initial guess: Least-squares
Xi = Theta \ dXdt;

% Lambda is a sparsifying threshold.
for k=1:10
    % find small coefficients
    smallinds = (abs(Xi) < lambda);

    % thresholding
    Xi(smallinds) = 0;

    for ind = 1:n
        % find big coefficients
        biginds = (abs(Xi) >= lambda);

        % Regress onto remaining terms
        Xi(biginds,ind) = Theta(:,biginds) \ dXdt(:,ind);
    end
end
```

The algorithms starts with a least-squares solution for $\boldsymbol{\Xi}$ and then threshold all coefficients smaller than some cutoff value λ. Once the indices of the

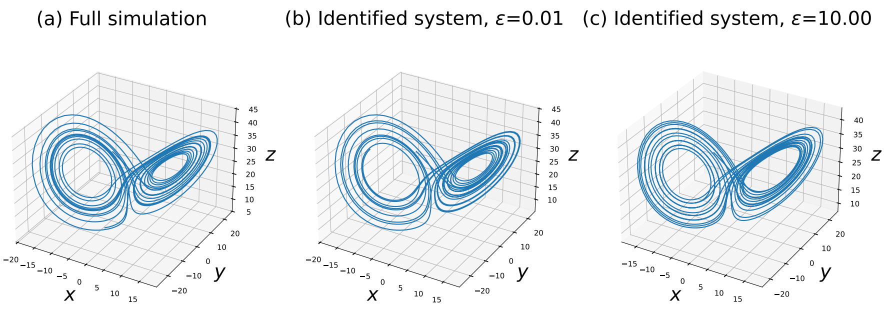


Figure 2.2: Discovery of the Lorenz system where $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ using SINDy [4] with the different noise levels. The leftmost subfigure is the full simulation of the Lorenz system without noise interference. The noise interference is intensified by ϵ, making $\dot{\mathbf{X}}$ noisy but keeping X (relatively) clean to imitate realistic scenarios in which $\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi} + \epsilon\boldsymbol{\mathcal{Z}}$. Each entry of $\boldsymbol{\mathcal{Z}}$ is sampled from the standard Gaussian distribution $\mathcal{N}(0,1)$.

remaining non-zero coefficients are obtained, we reapply another least-squares solution for $\mathbf{\Xi}$ onto the remaining indices. These new coefficients will be thresholded again using λ, and the procedure continues until the non-zero coefficients converge. STLS is computationally efficient, and rapidly converges to in a small number of iterations. STLS requires a single parameter λ to determine the degree of sparsity in $\mathbf{\Xi}$. Successful identification of the Lorenz system is shown in Figure 2.2. We evaluate a (correct) discovery result (for one equation) $\hat{\boldsymbol{\xi}}$ using its percentage coefficient error (%CE) given $\mathbb{E}_{i\in\mathcal{S}}\left[100 \times \left|\hat{\xi}_i - \xi_i^{\text{true}}\right| / \left|\xi_i^{\text{true}}\right|\right]$; where S is a set of indices of the true terms. If the system of interest is composed of multiple equations, we report the average error over the equations. For example, the %CE of the identified system is 1.39×10^{-4} for the case where ϵ is 0.01.

Nevertheless, more recent studies have demonstrated that sensitivity to the threshold λ is quite problematic and cannot be resolved easily. For instance, when $\epsilon = 1$, wrong equations that do not represent the true form of the Lorenz system are selected if the threshold is not set properly, as shown below. A threshold that is too large results in an overly sparse system where true terms are removed, while a threshold that is too small leads to overfitting—the failure to eliminate irrelevant terms. Therefore, selecting an appropriate threshold that identifies the true system is not a trivial task.

Too sparse: $\lambda = 1$	Correct: $\lambda = 0.1$	Overfitting: $\lambda = 0.01$
$\dot{x} = -9.997x + 9.998y$	$\dot{x} = -9.997x + 9.998y$	$\dot{x} = 0.012 - 9.997x + 9.998y$
$\dot{y} = -2.910x$	$\dot{y} = 28.002x - 1.000y - 1.000xz$	$\dot{y} = 28.002x - 1.000y - 1.000xz$
$\dot{z} = -2.667z + 1.000xy$	$\dot{z} = -2.667z + 1.000xy$	$\dot{z} = -0.069 - 2.664z + 1.000xy$

Remark that this sensitivity problem is prevalent, occurring at different noise levels and also in cases where the governing equations are PDEs.

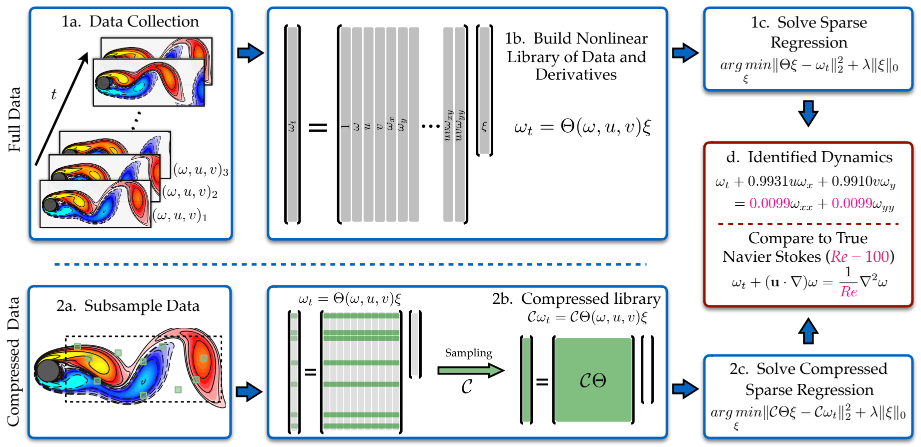

Full Data
1a. Data Collection
t
$(\omega, u, v)_3$
$(\omega, u, v)_2$
$(\omega, u, v)_1$
ω_t
1 ω u v ω_x ω_y $\cdots$ $uv\omega_{xy}$ $uv\omega_{yy}$
ξ
1b. Build Nonlinear Library of Data and Derivatives
$\omega_t = \Theta(\omega, u, v)\xi$
1c. Solve Sparse Regression
$\arg\min_{\xi} \|\Theta\xi - \omega_t\|_2^2 + \lambda\|\xi\|_0$
d. Identified Dynamics
$\omega_t + 0.9931u\omega_x + 0.9910v\omega_y = 0.0099\omega_{xx} + 0.0099\omega_{yy}$
Compare to True Navier Stokes ($Re = 100$)
$\omega_t + (\mathbf{u}\cdot\nabla)\omega = \frac{1}{Re}\nabla^2\omega$
Compressed Data
2a. Subsample Data
$\omega_t = \Theta(\omega, u, v)\xi$
2b. Compressed library
$\mathcal{C}\omega_t = \mathcal{C}\Theta(\omega, u, v)\xi$
Sampling
$\mathcal{C}$
$\mathcal{C}\Theta$
2c. Solve Compressed Sparse Regression
$\arg\min_{\xi} \|\mathcal{C}\Theta\xi - \mathcal{C}\omega_t\|_2^2 + \lambda\|\xi\|_0$

Figure 2.3: Schematic of PDE-FIND by [7] demonstrated on the Navier-Stokes Equations.

2.1.2 PDE-FIND: Data-driven discovery of PDEs

Rudy et al. [7] extended the capability of SINDy to discover governing PDEs from data, proposing the PDE-FIND (PDE functional identification of nonlinear dynamics) framework, also known as SINDy for (nonlinear) PDEs, which have the following parametrized form:

$$u_t = \mathcal{N}(u, u_x, u_{xx}, \ldots, x, \mu). \tag{2.4}$$

$\mathcal{N}$ is an unknown right-hand side (RHS) that is the governing nonlinear operator of u, its derivatives, and parameters in μ. For example, a Burgers' equation may have been formulated as $\mathcal{N} = \mu u_{xx} - u u_x$. The goal is to find $\mathcal{N}$ given time series measurements of the system at a certain number of spatial locations in x. Recall that the central assumption is that $\mathcal{N}$ consists of only a few terms, making the functional form sparse relative to the large space of possible candidate terms.

Upon discretization of (2.4), we construct the following system of linear equations:

$$\mathbf{U_t} = \mathbf{\Theta}(\mathbf{U}, \mathbf{Q})\boldsymbol{\xi}; \tag{2.5}$$

$\mathbf{U_t}$ is the time derivative, $\mathbf{\Theta}(\mathbf{U}, \mathbf{Q})$ is the candidate library constructed by additional information $\mathbf{Q}$, and $\boldsymbol{\xi}$ is the sparse vector of PDE coefficients. Then our goal can be transformed in into a sparse regression problem with a λ_0-controlled penalty on the number of active nonzero terms (support size) as follows:

$$\hat{\boldsymbol{\xi}} = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \left\|\mathbf{U_t} - \mathbf{\Theta}(\mathbf{U}, \mathbf{Q})\boldsymbol{\xi}\right\|_2^2 + \lambda_0 \|\boldsymbol{\xi}\|_0 . \tag{2.6}$$

Recall that the resulting $\hat{\boldsymbol{\xi}}$ is subject to the value of λ_0 chosen. According to [7], λ_0 is set proportional to the conditional number κ of the candidate library, more specifically $\lambda_0 = \lambda_0' \kappa(\mathbf{\Theta}(\mathbf{U}, \mathbf{Q}))$. The entire discovery process is illustrated in Figure 2.3. Since (2.6) is non-convex and the l_0-norm is not differentiable, Rudy et al. [7] has also proposed a relaxed algorithm for solving the sparse regression

Algorithm 1 STRidge($\mathbf{\Theta}, \mathbf{U_t}, \lambda_2, tol, \text{iters}$)

```
$\hat{\boldsymbol{\xi}} = \text{argmin}_{\boldsymbol{\xi}} \left\| \mathbf{U_t} - \mathbf{\Theta}(\mathbf{U}, \mathbf{Q})\boldsymbol{\xi} \right\|_2^2 + \lambda_2 \|\boldsymbol{\xi}\|_2^2$    # Ridge regression
bigcoeffs = $\{j : |\hat{\xi}_j| \geq tol\}$    # Select large coefficients
$\hat{\boldsymbol{\xi}}[\sim\text{bigcoeffs}] = 0$    # Apply hard threshold
# recursive call with fewer coefficients
$\hat{\boldsymbol{\xi}}[\text{bigcoeffs}] = \text{STRidge}(\mathbf{\Theta}[:, \text{bigcoeffs}], \mathbf{U_t}, \lambda_2, tol, \text{iters} - 1)$
return $\hat{\boldsymbol{\xi}}$
```

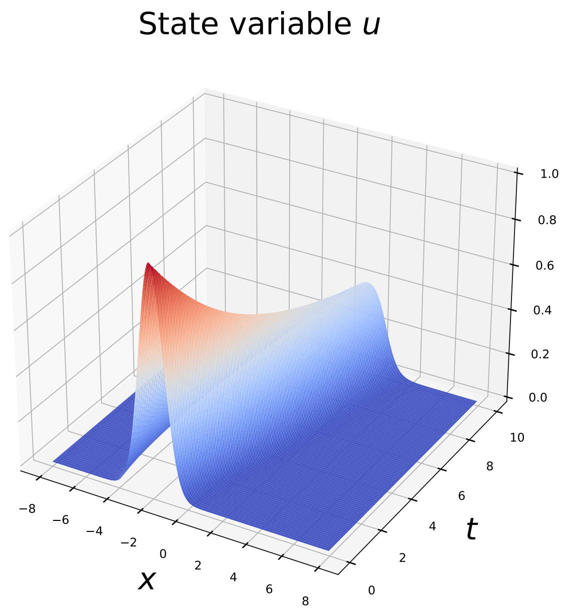


Figure 2.4: Visualizing (clean) Burgers' equation.

problem. The STRidge algorithm apply Ridge solver with hard-thresholding iteratively as detailed in **Algorithm 1**.

A successful discovery result of PDE-FIND, demonstrated on the Burgers' equation (see Figure 2.4 for a visualization of the state variable in space and time), is evaluated in Table. However, there are two major concerns with the PDE-FIND framework. The first concern is that how to properly set λ_0' is unclear. In the noisy case, if we set $\lambda_0' = 10^{-7}$, we get an overfitted PDE (with indispensible candidates like u^2u_x and uu_{xx} in addition to the true ones) as a result. Furthermore, if we set $\lambda_0' = 0.01$ for the clean case, we obtain a too sparse solution with only one term, uu_x. The other issue is related to the noise robustness of the framework. It turns out that when $\lambda_0' = 0.001$, an overly sparse solution (consisting just uu_x) is also identified when our measurements of u are more noisy, with $\epsilon = 2$. Lowing λ_0' does not help, as it failed to explore the true model with two active terms, resulting in a wrong overfitted equation:

Case	Identified PDE ($\lambda'_0 = 0.001$)	%CE
Clean	$u_t = -1.000403uu_x + 0.100145u_{xx}$	0.0927 ± 0.0524
Noisy ($\epsilon = 1$)	$u_t = -1.007779uu_x + 0.103338u_{xx}$	2.057 ± 1.280

Table 2.1: Identified equations (and their performance) obtained using PDE-FIND to discover the governing Burgers' equation. The results are created using code adapted from the implementation provided by [7].

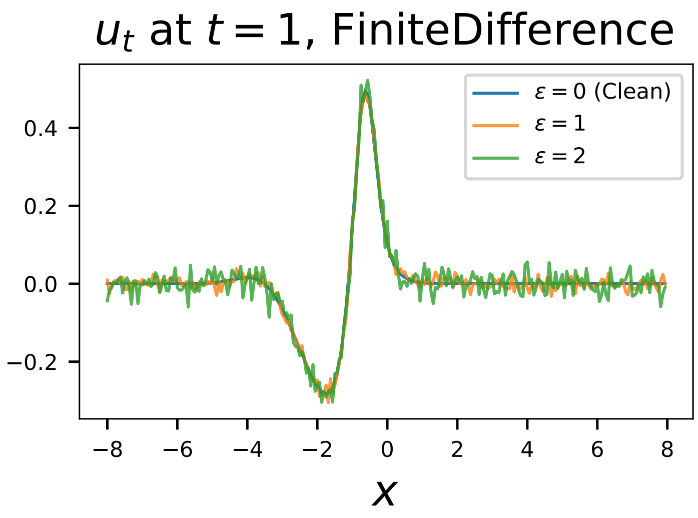


Figure 2.5: Noise interference negatively affects the quality of computed time derivatives [7].

$u_t = -1.053748uu_x + 0.559418uu_{xx} - 0.581526u^2u_{xx}$.

Noise interference affects the quality of computed derivatives, thereby reducing the success rate of data-driven discovery algorithms. As seen in Figure 2.5, noise introduces inaccuracies and fluctuations that distort the true signal, making it difficult to accurately capture the system's underlying dynamics. To build a robust discovery algorithm, a noise-tolerant (or noise-insensitive) and easy-to-use differentiation method is essential. The hyperparameters of the differentiation method should be easy to configure and less sensitive to reliably deal with noise. Note that one must be careful not to apply too much denoising, as it may alter the structure of governing equations or change their parameters substantially.

2.1.3 Evolutionary-based PDE discovery frameworks

Evolutionary-based PDE discovery frameworks were developed to relax the over-completeness assumption, which posits that all true terms are included in the candidate library (i.e., the feature matrix for SINDy and PDE-FIND). In an evolutionary-based framework, a PDE can be represented more flexibly than by relying on a large, typically fixed candidate library. Generally, the larger the candidate library, the more difficult it becomes to optimize for the best sparse

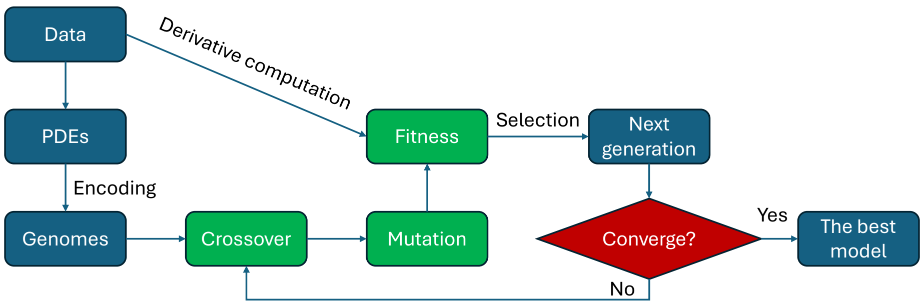


Figure 2.6: Schematic of the DLGA-PDE framework (adapted from [20]) of using deep and genetic learning algorithms to discover the best PDE governing scientific data.

PDE form.

Xu et al. [20] proposed DLGA-PDE, a PDE discovery framework based on deep and genetic learning algorithms. In the framework, a PDE is represented as a linear combination of modules, each being a product of derivative terms (including the 0-order term, which is the data itself), calculated using automatic differentiation applied on a multi-layer perceptron neural network. A PDE is also referred to as a "genome" to fit within the context of genetic algorithms, which are used to optimize the best-fit PDE, as shown in Figure 2.6. Having a population of genomes are indeed more flexible than manually crafting possible candidate terms. We next discuss the essential steps of the DLGA-PDE framework, which include encoding a PDE, crossover, mutation, and the processes of selection and evolution. Encoding a PDE as a genome is based on the principles outlined as follows.

Encoding a PDE

Definition 1 (Encoding) *Numbers are used to represent the corresponding order of derivatives. For example:*

$$u \leftrightarrow 0, \quad \frac{\partial u}{\partial x} \leftrightarrow 1, \quad \frac{\partial^2 u}{\partial x^2} \leftrightarrow 2, \quad \frac{\partial^3 u}{\partial x^3} \leftrightarrow 3.$$

Definition 2 (Module) *Each PDE term is considered as a module. Here, it is assumed that there is only multiplication in a module. In fact, most PDEs can be split into a series of multiplication and addition combinations. For example:*

$$u\frac{\partial u}{\partial x} \leftrightarrow [0,1], \quad u\left(\frac{\partial^2 u}{\partial x^2}\right)^2 \leftrightarrow [0,2,2].$$

Definition 3 (Genome) *Combining these modules yields the genome of a PDE, with modules connected by a plus sign. Note that there are two parts to the genome component of the PDE: one is the module group on the left-hand side, and the other is the module group on the right-hand side, which is enclosed in braces. Since only temporal derivatives are on the left-hand side, the term on the left-hand side is encoded in the same way. We mostly consider cases in which* *the left-hand side has a single term, i.e., a genome length of 1 on the left-hand side. Extending this approach to a broader range of PDEs is straightforward. For example, the genome of the contaminant transport equation is*

$$u_t = -v_x u_x + D_L u_{xx} \leftrightarrow [1], \{[1], [2]\}.$$

Similarly, the genome of the wave equation is

$$u_{tt} = A u_{xx} \leftrightarrow [2], \{[1], [2]\}.$$

Definition 4 (Fitness) *The fitness function for the DLGA-PDE framework is*

$$\text{Fitness} = \text{MSE} + \lambda \cdot len(\text{genome});$$
$$\text{MSE} = \frac{1}{N} \left\| \mathbf{U_t} - \mathbf{\Theta_G}(\mathbf{U}) \boldsymbol{\xi}_{\mathbf{LS}} \right\|_2^2 .$$

$\mathbf{\Theta_G}(\mathbf{U})$ *is is the candidate library (each column is seperated by +.) of RHS in the evaluated PDE.*
$\boldsymbol{\xi}_{\mathbf{LS}}$ *is its corresponding vector of coefficients, computed using least squares.*

Crossover

For each genome, a portion of its modules can be swapped with modules from another genome based on a specified probability to create the next generation.

Example:

$$\text{Input: } [1], \{[1], [2]\} \leftrightarrow [1], \{[1, 3], [0, 2]\}$$
$$\text{Output: } [1], \{[1, 3], [2]\} \quad \text{and} \quad [1], \{[1], [0, 2]\}$$

Here $\leftrightarrow$ refers to an act of crossover. According to [20], the crossover rate is 80%. The process of crossover increases the possibility of different gene combinations.

Mutation

Mutation alters certain genes within a genome, leading to the creation of a new genome. Here, three types of mutation methods are defined.

Definition 5 (Mutation) *There are three types of mutation.*

- *Order mutation: With a certain probability, the order of derivatives in a gene is reduced by 1. Specifically, if the order is 0, it can mutate to any random higher order.*

  $$\textit{Input: } [1], \{[1, 2], [3]\} \rightarrow \textit{Output: } [1], \{[0, 2], [3]\}$$

  $$\textit{Input: } [1], \{[1, 3], [0]\} \rightarrow \textit{Output: } [1], \{[1, 3], [2]\}$$

- *Add mutation: (Add-module Mutation) With a certain probability, a random module is added into the genome.*

  $$\textit{Input: } [1], \{[1, 2], [3]\} \rightarrow \textit{Output: } [1], \{[1, 2], [3], [0, 0]\}$$

Generations	Structure of the best-discovered child
1	$u_t, u^2, u_x, u_{xxx}, uu_{xx}, u^2u_{xx}$
2	$u_t, u, u_x^2, uu_{xx}, uu_x, u_{xx}u_{xxx}$
4	$u_t, u_{xx}, uu_{xxx}, u_{xx}^2, u_{xxx}$
5	$u_t, u_{xx}, uu_{xxx}, u_{xx}^2, uu_{xx}u_{xxx}, u$
6	$u_t, u_{xx}, u_x^2, u_{xx}, u^3$
36	$u_t, uu_{xxx}, u_{xx}^2, u, u^3$
39	$u_t, u_{xx}, u_x^2, uu_{xx}u_{xxx}, u^3$
40	u_t, u_{xx}, u, u^3
100	u_t, u_{xx}, u, u^3

Table 2.2: The best child found in each generation when DLGA-PDE is utilized to discover the Chaffee-Infante equation, which contains exactly u_{xx}, u, and u^3 as the true terms. This table is sourced from [20].

- *Delete mutation: With a certain probability, a module is deleted from the genome.*

$$\textit{Input:}\,[1], \{[1,2],[4],[0,1],[1,3]\} \rightarrow \textit{Output:}\,[1], \{[1,2],[4],[0,1]\}$$

Order Mutation adjusts the order of derivatives within the genome, while Add-module Mutation and Delete-module Mutation modify the genome's length. These three mutation types can occur independently, providing varied avenues for altering the genome. This diversity in mutation methods enhances the potential for diverse transformations within the genome, aiding in the discovery of the best model.

Selection and evolution

In the process of crossover, each parent genome crosses over twice, producing twice as many genomes as the parent genome. For example, 50 parents will produce 100 children via crossover. The children then undergo the process of mutation. The children's fitness will be calculated and sorted from smallest to largest. The top half of the children are chosen as a new generation of parents. We set the number of genomes and the number of generations. After certain epochs, the best model in terms of the fitness is obtained.

Experimental results

Xu et al. [20] presented experimental results demonstrating that the DLGA-PDE framework performs quite well, identifying the true structure of well-known equations such as the KdV Equation, Burgers Equation, and Chaffee-Infante Equation. Table 2.2 lists how the framework gradually finds the true parsimonious form of the Chaffee-Infante PDE over generations of evolutionary learning.

One advantage of the fitness function (in **Definition 4**) is its computational efficiency and thus its ability to quickly yield decent model selection results. However, the selection of the best model is sensitive to the value of λ chosen for the fitness function, as demonstrated in Table 2.3. An excessively large λ lead to an overly sparse best model. At this stage, additional quantities or insights (such as quantified uncertainty of the PDE discussed in Chapter 4) or

$\lambda = 1 \times 10^{-3}$	$\lambda = 1 \times 10^{-4}$
$-0.9053uu_x$ (Incorrect)	$-1.00037uu_x + 0.10118u_{xx}$ (Correct)
$-0.38103u_x$	$-1.00089uu_x + 0.0049u_x^2u_{xx} + 0.10016u_{xx}$
$-1.00037uu_x + 0.10118u_{xx}$	$-1.00157uu_x - 0.00359u_xu_{xx} + 0.1u_{xx}$
$-1.01857uu_x + 0.15675uu_{xx}$	$-0.00574u_xu_{xx} - 1.00291uu_x + 0.10045u_{xx}$
$-1.01392uu_x + 0.03792u_{xx}^3$	$0.01277u^2u_x - 1.00741uu_x + 0.10103u_{xx}$
$-1.01181uu_x - 0.16657u_xu_{xx}$	$0.00246u^3 - 1.00079uu_x + 0.10162u_{xx}$
$-1.39506u^2u_x$	$0.00136u^2 - 1.00069uu_x + 0.10152u_{xx}$
$0.17544u^2u_{xx} - 0.99174uu_x$	$-0.99622uu_x - 0.00191u_x + 0.10095u_{xx}$
$-0.97383uu_x + 0.22702u_x^2u_{xx}$	$-1.00078uu_x + 0.00031u_{xx}^3 + 0.10067u_{xx}$
$-1.63194u^2u_x + 0.11492u_{xx}$	$-0.99833uu_x - 0.00229u_{xx}^3 + 0.10107u_{xx}$

Table 2.3: Over generations of evolutionary learning within the DLGA-PDE framework, the top 10 models in terms of fitness with differnt λs are listed. The results are created using code adapted from the implementation guided by [20].

numerical simulation might be required to assist in selecting the best model. For example, we could simulate the identified PDEs and compare the simulated state variables against our (noisy) measured data.

2.2 Treatment of noisy data

As shown in Figure 2.5, preprocessing noisy data is highly advisable to mitigate the negative effects of noise interference. This step is beneficial as it allows for the derivation of PDEs from higher-quality data. Denoising the data prior to derivative computation increases the chances of identifying the true PDE structure from a set of higher-quality candidate models. Since several methods are applicable to achieve this goal, we discuss some of them relevant to this thesis.

A frequency-domain approach, such as denoising using the Discrete Time Fourier Transform (DFT), is a classic and computationally efficient method for noise filtering. In this method, noise—typically associated with low-power components in the Fourier domain—is removed, while retaining the frequency components that represent the underlying signal. This approach is especially effective when the signal is dominated by high-power frequencies that are easily distinguishable from the remaining components.

One common approach involves polynomial-based techniques for derivative computation. These methods approximate noisy data with polynomials, reducing the impact of noise by smoothing derivative values (e.g., [7]). Similarly, spline-based models have proven effective for denoising by constructing smooth, flexible representations of data that suppress noise while preserving underlying trends [21]. Another notable method is Robust Principal Component Analysis (PCA), which separates noise components from data by decomposing the dataset into low-rank and sparse components [22].

An alternative class of denoising techniques operates with minimal assumptions about noise statistics, setting them apart from traditional approaches like Kalman filtering. Among these, the regularized K-SVD (K-Singular Value Decomposition) algorithm is particularly noteworthy. This dictionary learning

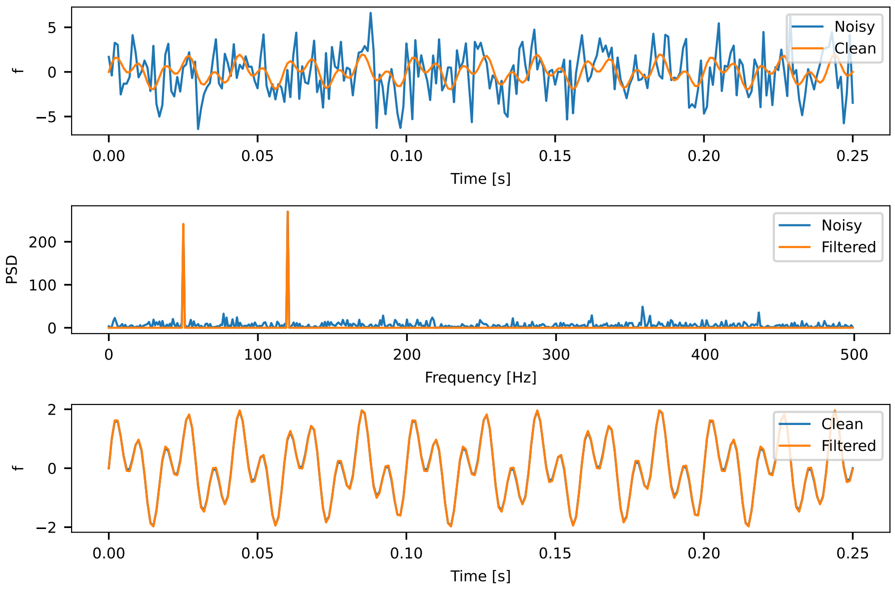


Figure 2.7: Denoising Using FFT: **(Top)** Noise is introduced to the composed of two sine waves. **(Middle)** In the Fourier domain, the prominent peaks can be identified and the noise is filtered out. **(Bottom)** The denoised signal is reconstructed by inverse Fourier transforming on the two dominant peaks. This figure is generated using the Python code provided by Brunton and Kutz [23].

method computes sparse representations of data, which are then used to reconstruct denoised observations effectively. Such techniques are advantageous in applications requiring robust data preprocessing, for example, the data-driven discovery of PDEs.

2.2.1 Discrete Fourier Transform (DFT)

The DFT is essentially a discretized version of the Fourier series for vectors of data $\begin{bmatrix} f_1 & f_2 & \dots & f_N \end{bmatrix}^\intercal$ obtained by discretizing the continuous function $f(x)$ at a regular spacing, Δx. Although the FFT is always used for practical computations due to its efficiency—$\mathcal{O}(N \log N)$, it is helpful to introduce the fundamental formulation of the DFT:

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j e^{-2\pi i \frac{jk}{N}}. \tag{2.7}$$

The inverse discrete Fourier transform (iDFT) is given by

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} \hat{f}_j e^{2\pi i \frac{jk}{N}}, \tag{2.8}$$

The DFT is a linear operator (i.e., a matrix) that maps the data points into the frequency domain: $\{f_1, f_2, \dots, f_n\} \mapsto \left\{\hat{f}_1, \hat{f}_2, \dots, \hat{f}_n\right\}$.

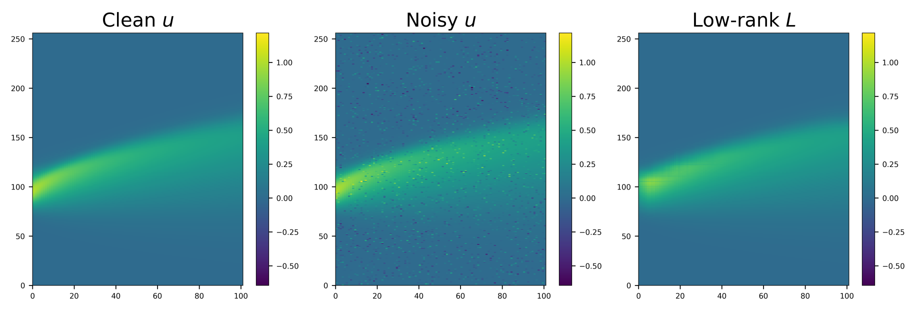


Figure 2.8: Example of denoising data using RPCA. **(Left)** The clean state variable u of the Burgers' PDE. **(Middle)** The noisy state variable with sparse noise added. **(Right)** The low-rank component L decomposed using RPCA. The implementation of RPCA used to create this figure is sourced from [23].

FFT Example: Noise Filtering

We consider an example (by Brunton and Kutz [23]) of applying FFT to denoise a distorted signal of time. The clean function is given as

$$f(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t); \tag{2.9}$$

where $f_1 = 50\,\text{Hz}$ and $f_2 = 120\,\text{Hz}$. We consider $t \in [0,1]$, discretized with $\Delta t = 0.001$. Gaussian white noise is added to this signal (The top panel of Figure 2.7). The FFT of the noisy signal computes its frequency spectrum, revealing the power spectral density (PSD), which indicates the signal power at each frequency. As shown in the middle panel, two prominent peaks at 50 Hz and 120 Hz are observed. Note that the PSD is expressed by $\frac{1}{N}\hat{\mathbf{f}} \odot \text{conj}(\hat{\mathbf{f}})$.

Noise can be removed by zeroing out frequency components with power below a threshold. After applying an inverse FFT to the filtered spectrum, the reconstructed/denoised signal aligns well with the original signal.

2.2.2 Robust Principal Component Analysis

Robust Principal Component Analysis (RPCA) was developed to address the limitations of traditional Principal Component Analysis (PCA), which is highly sensitive to outliers and corrupted data. RPCA decomposes a given data matrix X into two components: $X = L + S$; where L is a low-rank matrix capturing the underlying structure of the data. S is a sparse matrix representing outliers or noise.

The objective of RPCA is to find L and S by solving the following optimization problem:

$$\min_{L,S} \text{rank}(L) + \|S\|_0 \quad \text{subject to } X = L + S. \tag{2.10}$$

Here $\text{rank}(L)$ ensures that L is low-rank. $\|S\|_0$ is the ℓ_0-norm counts the number of non-zero entries in S, ensuring its sparsity.

However, neither the $\text{rank}(L)$ nor the $\|S\|_0$ terms are convex, and this is not a tractable optimization problem. Similar to the compressed sensing problem, it is possible to solve for the optimal L and S with high probability using the convex relaxation:

$$\min_{L,S} \|L\|_* + \lambda\|S\|_1 \quad \text{subject to } X = L + S. \tag{2.11}$$

The nuclear norm $\|L\|_*$ is the sum of singular values, which serves as a proxy for rank. The solution to the relaxed optimization problem converges to the solution of the RPCA objective with high probability if $\lambda = 1/\sqrt{\max(n, m)}$, where n and m are the dimensions of X. This assumes that L is not sparse, S is not low-rank, and the entries are randomly distributed such that they do not have a low-dimensional column space.

The objective of RPCA is known as principal component pursuit (PCP), and may be solved using the augmented Lagrange multiplier (ALM) algorithm. Specifically, an augmented Lagrangian may be constructed:

$$\mathcal{L}(L, S, Y) = \|L\|_* + \lambda\|S\|_1 + \langle Y, X - L - S\rangle + \frac{\mu}{2}\|X - L - S\|_F^2.$$

A general solution would solve for the L_k and S_k that minimize $\mathcal{L}$, update the Lagrange multipliers $Y_{k+1} = Y_k + \mu(X - L_k - S_k)$, and iterate until the solution converges. However, for this specific system, the alternating directions method (ADM) provides a simple procedure to find L and S. First, a shrinkage operator $S_\tau(x) = \text{sign}(x)\max(|x| - \tau, 0)$ is constructed. Next, the singular value threshold operator $\text{SVT}_\tau(X) = US_\tau(\Sigma)V^*$ is constructed.

The use of RPCA to extract the low-rank component from distorted measurement data of the Burgers PDE is demonstrated in the Figure 2.8. This serves as an example of how RPCA can be applied to denoise data prior to PDE identification processes. The Frobenius norm distances from the clean u to the noisy u and to the low-rank component 8.88 are 3.09, respectively. Therefore, it is better to perform PDE discovery algorithms on the low-rank component L, as it is of higher quality.

2.2.3 Weak PDE formulation for robust PDE discovery

Due to the difficulty in accurately computing derivatives especially for high-order ones in noisy scenarios, we can recover the governing equation through its weak formulation [24], bypassing the direct computation of candidate terms, which are mostly derivatives. In fact, this formulation is a general approach for handling noisy measurement data. We will illustrate the procedure with examples where we approximate the single (first-order) temporal derivative of the state variable using a linear combination of terms as follows

$$\partial_t u = \sum_{j=1} \xi_j \mathcal{N}_j(u, \partial_x u, \partial_x^2 u, \dots). \tag{2.12}$$

To setup the weak formulation of the problem, we proceed by multiplying (2.12) by a smooth weight $\mathbf{w}$ and integrate the result over a spatio-temporal domain Ω_k. This procedure is repeated for every k, yielding

$$\mathbf{q}_0 = \sum_{j=1} \xi_j \mathbf{q}_j; \text{ where } \quad q_j^k = \int_{\Omega_k} \mathbf{w} \cdot \mathcal{N}_j \, d\Omega. \tag{2.13}$$

$\mathbf{q}_j$ is a column vector whose height depends on the number of spatio-temporal subdomains. By performing integration by parts, the action of taking derivatives can be transferred from the noisy data u onto the smooth weight $\mathbf{w}$, thereby dramatically reducing the negative effect of noise on the derivative terms. Furthermore, the weight function can be chosen to eliminate the terms with latent variables, yielding a new problem that can be solved using standard regression techniques.

We then demonstrate how to construct the weak formulation of the problem of discovering the Kuramoto-Sivashinsky equation: $\partial_t u + u\partial_x u + \partial_x^2 u + \partial_x^4 u = 0$. The PDE contains a fourth-order spatial derivative, which is extremely sensitive even to small amounts of noise. Considering just a subdoamin Ω_k, the weakly formulated (true) terms are given in the following.

$$q_0^k = \int_{\Omega_k} w\partial_t u \, d\Omega, \; q_1^k = \int_{\Omega_k} wu\partial_x u \, d\Omega, \; q_2^k = \int_{\Omega_k} w\partial_x^2 u \, d\Omega, \;\; q_3^k = \int_{\Omega_k} w\partial_x^4 u \, d\Omega$$

w is a scalar weight. An example of the subdomain would be $\Omega_k = \{(x,t) : |x - x_k| \leq H_x, |t - t_k| \leq H_t\}$, which is centered around randomly chosen points (x_k, t_k). Using integration by parts, we take all derivatives on a smooth noiseless w instead of the noisy data u, yielding

$$\begin{aligned} q_0^k &= -\int_{\Omega_k} u\partial_t w \, d\Omega, & q_1^k &= -\int_{\Omega_k} \frac{1}{2}u^2\partial_x w \, d\Omega, \\ q_2^k &= \int_{\Omega_k} u\partial_x^2 w \, d\Omega, & q_3^k &= \int_{\Omega_k} u\partial_x^4 w \, d\Omega, \end{aligned}$$

provided that w satisfies the conditions required for the boundary terms to vanish. Numerical experiments by Reinbold et al. [24] demonstrated that the weak formulation, which involves integrals of the data rather than derivatives, is significantly more robust to noise than the standard problem formulation. While the weak formulation may not completely eliminate all derivatives in some models, it can reduce the order of the remaining derivatives, a property that is particularly advantageous when dealing with noisy data. The weak formulation presented here will be utilized again in Chapter 4.

2.3 Information criteria for model selection

Model selection is a crucial process in statistical modeling, where one aims to select the best model from a set of candidate models. Two commonly used criteria for model selection are the Akaike Information Criterion (AIC) [9] and the Bayesian Information Criterion (BIC) [10]. Both criteria provide methods for balancing model fit with model complexity, helping to avoid overfitting while capturing the underlying data structure. Here, we discuss the theoretical foundations, interpretations, and comparative aspects of AIC and BIC in the context of statistical and machine learning models.

2.3.1 Akaike Information Criterion (AIC)

The Akaike Information Criterion (AIC) was introduced by Hirotugu Akaike in 1973 as an information-theoretic approach to model selection. AIC is based on the concept of information theory, aiming to minimize the Kullback–Leibler information loss when approximating a true model with a fitted model. The AIC for a model with s parameters, fit to N observations, is given by:

$$\mathrm{AIC} = \frac{1}{N}\left(\mathrm{RSS} + 2s\hat{\sigma}^2\right), \tag{2.14}$$

where RSS represents the residual sum of squares, and $\hat{\sigma}^2$ is an estimate of the variance of the Gaussian error term. Typically, $\hat{\sigma}^2$ is estimated using the full model that includes all predictors. This AIC formulation accounts for the fact that the training error tends to underestimate the test error. The penalty term increases with the number of predictors in the model, counterbalancing the decrease in training RSS. AIC can also be expressed in terms of the maximum value of a model's log-likelihood function:

$$\mathrm{AIC} = 2s - 2\log\left(\hat{L}\right), \tag{2.15}$$

where $\hat{L}$ is the maximum likelihood function of the model. AIC provides a trade-off between goodness of fit and model complexity by adding a penalty term proportional to the number of parameters. This penalty discourages overfitting, as models with more parameters are penalized. In practice, the model with the lowest AIC is generally preferred. In practice, the AIC requires a correction for finite sample sizes given by $\mathrm{AIC_c} = \mathrm{AIC} + \frac{2(s+1)(s+2)}{N-s-2}$. Note that, usually, a lower value of the information criterion indicates a better model.

2.3.2 Bayesian Information Criterion (BIC)

The Bayesian Information Criterion (BIC), also known as the Schwarz criterion, was proposed by Gideon Schwarz in 1978. BIC is derived from a Bayesian perspective (but ends up looking similar to AIC) and can be used to approximate the Bayes factor for model selection. It is defined as:

$$\mathrm{BIC} = \frac{1}{N}\left(\mathrm{RSS} + \log(N)s\hat{\sigma}^2\right), \tag{2.16}$$

or, in terms of likelihood, as:

$$\mathrm{BIC} = s\log(N) - 2\log\left(\hat{L}\right). \tag{2.17}$$

Unlike AIC, BIC imposes a heavier penalty for model complexity due to the $\log(N)$ term, which grows with the sample size. This makes BIC more conservative, often favoring simpler models. The model with the lowest BIC is generally selected.

2.3.3 Comparison of AIC and BIC

AIC and BIC are similar in form but differ in their theoretical foundations and implications for model selection. AIC is derived from an information-theoretic approach and focuses on minimizing prediction error, aiming to select the model that best describes an unknown, high-dimensional reality. BIC, on the other hand, is derived from a Bayesian framework and incorporates sample size into its penalty term, favoring simpler models as the sample size increases. This

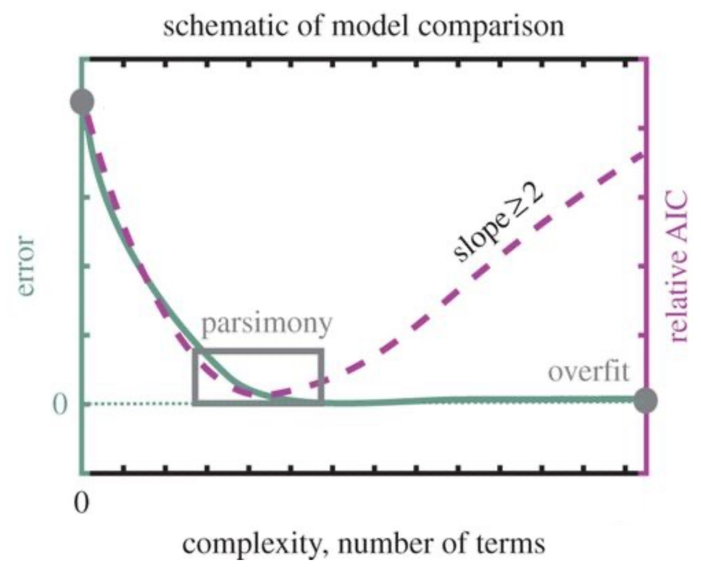


Figure 2.9: Schematic of Pareto front for evaluating the number of terms [25]. Note that the standard AIC score has an asymptotic penalty of $2s$ for the number of terms, resulting in a slope of at least 2 for large s.

distinction means that BIC is more likely to select models closer to the true parsimonious model, particularly with large datasets.

In practice, AIC tends to select slightly more complex models, while BIC often favors more parsimonious ones. Both criteria, however, can help in selecting models that balance fit and complexity, with AIC being especially useful in predictive modeling contexts and BIC in contexts where model simplicity is prioritized.

Both AIC and BIC are widely used across various fields, including econometrics, machine learning, and bioinformatics. In regression analysis, for instance, these criteria assist in selecting the optimal number of predictors by penalizing overly complex models that may lead to overfitting. However, practitioners should consider the limitations of each criterion: AIC may select overly complex models in small-sample scenarios, whereas BIC may favor overly simplistic models if the sample size is limited. Thus, model selection often involves evaluating both criteria while considering the specific goals and constraints of the modeling task.

2.3.4 Application of AIC and BIC to PDE discovery

The application of information criteria in the data-driven discovery of PDEs is relatively straightforward. After obtaining multiple candidate PDEs, for instance, through the sparse regression methods discussed earlier, each potential PDE is ranked using AIC or BIC. Generally, a lower information criterion value indicates a better ranking for the corresponding PDE. At this stage, the RSS terms in (2.14) and (2.16) can take the form of $\left\|\mathbf{U}-\hat{\mathbf{U}}\right\|_F^2$ or $\left\|\mathbf{U_t}-\hat{\mathbf{U}}_\mathbf{t}\right\|_F^2$, where $\hat{\mathbf{U}}$ is the simulated state variable (an approximation of the measurement data)

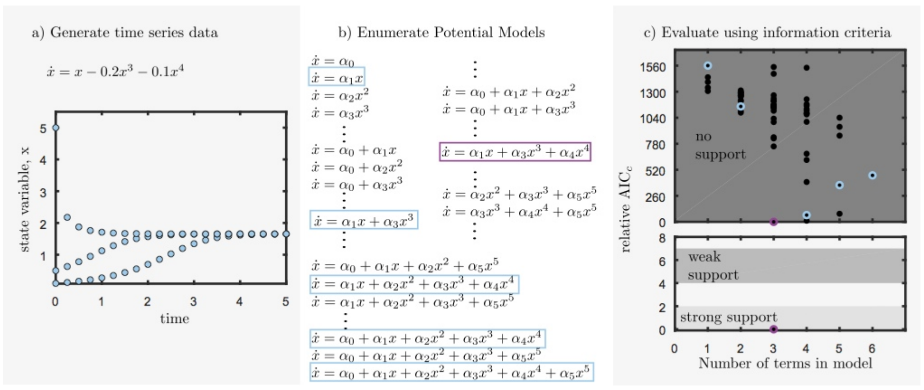


Figure 2.10: Model selection using relative AICc for single variable, x, polynomial system. a) 3 computationally generated time series. b) Combinatorial model possibilities with with those selected by SINDy highlighted in blue. c) Relative AICc criteria for all possible models (black dots), and those found by SINDy (blue circles). Lower plot magnifies strong and weakly supported AICc range, containing only correct model (magenta circle). This figure is sourced from [25].

for a potential PDE, and $\hat{\mathbf{U}}_\mathbf{t}$ is an estimation of the computed time derivative. The distinction lies in their evaluation focus: the first choice [25, 26] directly assesses how closely the simulated state variable aligns with the measurement data (because the governing PDE should describe the system state over time), while the second choice (ours) [11, 13] evaluates the regression model that represents the PDE instead. The second approach is computationally faster, as it does not require PDE simulations. However, it can be more challenging to identify the best model among a cluster of multiple parsimonious PDEs. In contrast, the first approach, being a direct comparison, often allows the best model to stand out more clearly. Nonetheless, this method necessitates PDE simulations, which are currently infeasible for arbitrary PDE forms and domains.

The complexity penalty remains unchanged for both choices, and we essentially ask the same question: *Does the PDE explain the measured data parsimoniously, using only the necessary variables in line with the principle of Occam's razor?* Figure 2.9 highlights the trade-off between model complexity and error. Simple models with no terms have high error, while adding more terms reduces error as the model better fits the data. However, as the number of terms approaches the number of free parameters, overfitting becomes a concern, especially in noisy data, leading to poor predictions for validation experiments. The goal is to find the best model among parsimonious models—those with minimal terms that sufficiently reduce error—aligning with Occam' s razor. However, interpreting the Pareto front is not an easy task, as there may not be a clear "elbow" point to select but rather a cluster of models to choose from.

Mangan et al. [25], an early work on using sparse regression and information criteria for SINDy, presents a complete discovery process guided by the corrected AIC, as demonstrated in Figure 2.10. This figure provides an example application of how to use the information criterion for ODE discovery (extendable to PDE discovery). However, their model selection process can be computationally expensive, even when simulating tens of potential models to discover the Burgers' PDE. It is important to stress again that these potential models are essentially represented by regression models that approximate temporal derivatives (e.g., first-order, second-order, etc.). If we can evaluate these regression models in such a way that the best model is indeed the correct governing equation, we could avoid simulating all potential models, expediting the discovery process substantially. This problem will be addressed by the new uncertainty-guided information criterion in Chapter 4.

2.4 Physics-informed machine learning

Although significant advances have been made in simulating multiphysics problems using numerical methods to solve PDEs, challenges remain. Traditional solvers often struggle to integrate noisy data effectively due to the complexity of discretized mesh generation. Solving inverse problems, particularly those involving hidden physical processes, also requires unique approaches and costly computation. Machine learning offers an alternative approach, though training deep neural networks usually requires extensive data, which may not always be available in scientific applications. Training these networks can, nonetheless, be guided by enforcing physical laws as additional priors. This method, known as physics-informed learning, combines (noisy) data with mathematical models and implements them through neural networks. Moreover, specialized network architectures can be designed to obey certain physical invariants, potentially improving accuracy, training efficiency, and generalization compared to traditional numerical methods. How much domain-specific knowledge or physics needed to develop a physics-informed machine learning method depends on the data size of the physical problem, which can be classified into the three categories shown in Figure 2.11. The more data available, the less need there is to explicitly specify the governing physics when training neural networks. Nonetheless, some consideration should also be given to maintaining the interpretability of the build model.

2.4.1 Physics-informed neural networks (PINNs)

Physics-informed neural networks (PINNs) [12] combine information from measurement data and governing equations in the form of PDEs by embedding these PDEs (directly) into the neural network's loss function. This approach is possible because numerical derivatives can be computed using automatic differentiation, allowing the entire network to be trained via gradient-based optimization.

To better understand how the PINN's loss function is formulated, we provide an example using a PINN to solve a forward problem governed by Burgers' PDE, which appears in many applied mathematics fields, such as fluid mechanics, nonlinear acoustics, and traffic flow. This fundamental PDE can be derived from the Navier-Stokes equations by omitting the pressure gradient term. With low viscosity values, Burgers' equation can lead to shock waves that are challenging for traditional numerical methods to resolve. We consider the following Burgers' equation with Dirichlet boundary conditions in one spatial dimension $x \in [-1, 1]$:

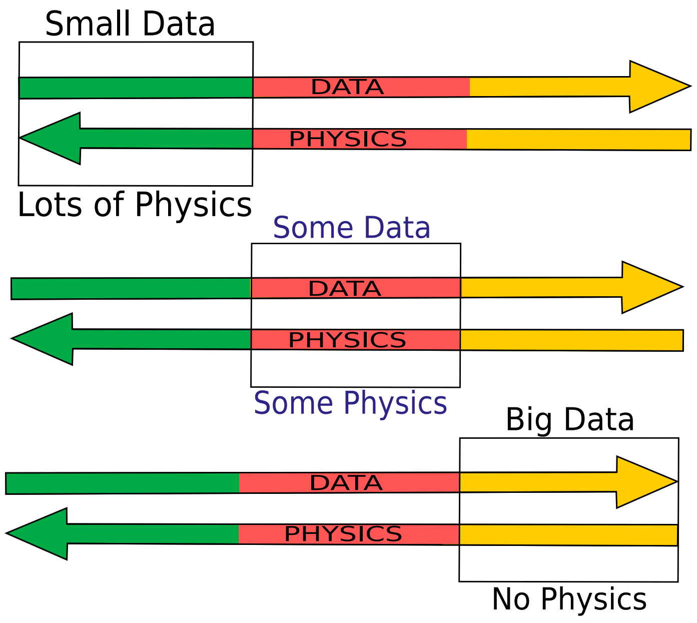


Figure 2.11: Schematic to illustrate three possible scenarios of physical problems categorized by the amount of data available, as described by Raissi et al. [27].

$$
\begin{aligned}
&u_t + uu_x - \frac{0.01}{\pi}u_{xx} = 0,\\
&u(0,x) = -\sin(\pi x),\\
&u(t,-1) = u(t,1) = 0
\end{aligned}
$$

Suppose that $t \in [0,1]$. We proceed by approximating u by a neural network $\mathcal{F}$ parameterized by θ. Then to update θ such that the network solves the equation, we sample collocation points (t_f, x_f) within the spatial-temporal domain of interest then put the PDE optimization constraint:

$$\mathcal{L}_f^i = \partial_t \mathcal{F}_\theta^i + \mathcal{F}_\theta^i \partial_x \mathcal{F}_\theta^i - \frac{0.01}{\pi}\partial_x^2 \mathcal{F}_\theta^i = 0;$$

where $\mathcal{F}_\theta^i = \mathcal{F}_\theta(t_f^i, x_f^i)$. This constraint naturally gives rise to the PDE-constrained loss function, defined as:

$$\mathrm{MSE}_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|\mathcal{L}_f^i\right|^2 \tag{2.18}$$

Another important loss function to satisfy boundary and initial conditions is also formulated as a simple squared loss:

$$\mathrm{MSE}_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u^i - \mathcal{F}_\theta(t_u^i, x_u^i)\right|^2; \tag{2.19}$$

where (t_u, x_u) represents data sampled from the boundaries or at the initial time step. By combining (2.18) and (2.19), the PINN is trained by minimizing

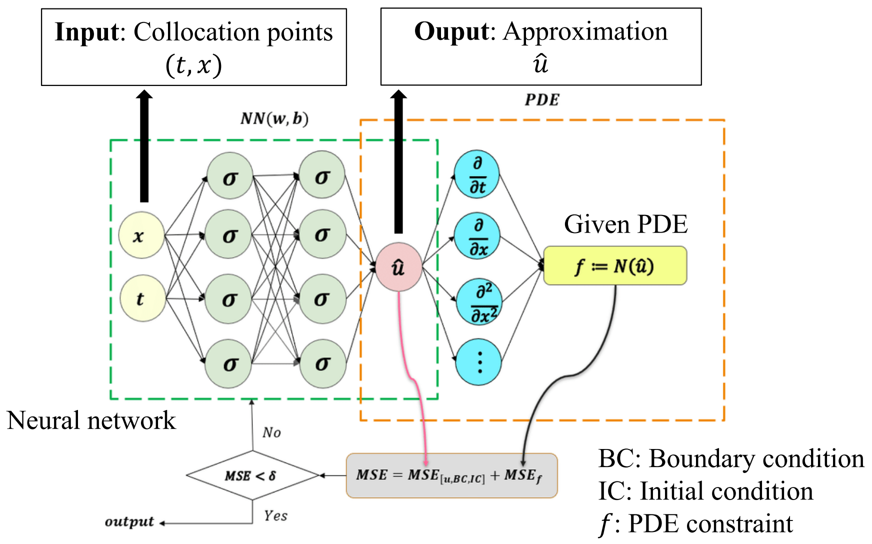


Figure 2.12: Overview of how the physics-informed loss function is formulated. This figure is adapted from [28].

the mean squared loss function $\text{MSE}_f + \text{MSE}_u$ Note that constructing physics-informed loss functions is highly flexible in practice, allowing PINNs to be extended to solve other equations, such as ODEs or higher-dimensional PDEs.

The experimental results of using the PINN to solve Burgers' PDE are visualized in Figure 2.13. The top panel displays the predicted spatio-temporal solution, $\hat{u}(t, x)$, along with the locations of initial and boundary training data points. Unlike traditional numerical methods for solving partial differential equations, this prediction is achieved without discretizing the spatio-temporal domain. A detailed evaluation of the predicted solution is shown in the bottom panel of Figure 1, where we compare the exact and predicted solutions at different time points: $t = 0.25, 0.50$, and 0.75. With only a handful of initial and boundary data, the PINN accurately captures the complex nonlinearity of Burgers' equation, a sharp internal layer (around $t = 0.4$), which is notoriously difficult for classical numerical methods to resolve accurately, as it typically requires extensive spatio-temporal discretization.

2.4.2 Deep operator learning

In the previous section, we discuss PINNs, originally proposed to numerically solve a specific PDE. In this section, we introduce a new neural network architecture called DeepONet (deep operator network) [29], which can learn nonlinear continuous operators for complex systems. While the success of PINNs is grounded in the theory of neural networks as universal function approximators, DeepONet relies on a lesser-known but powerful theory: a neural network with a single hidden layer can accurately approximate any nonlinear continuous operator. DeepONet has a small generalization error and is suitable for a wide range of applications. Additionally, DeepONets can be integrated with the physics-informed principles embedded in PINNs (see the physics-informed DeepONet

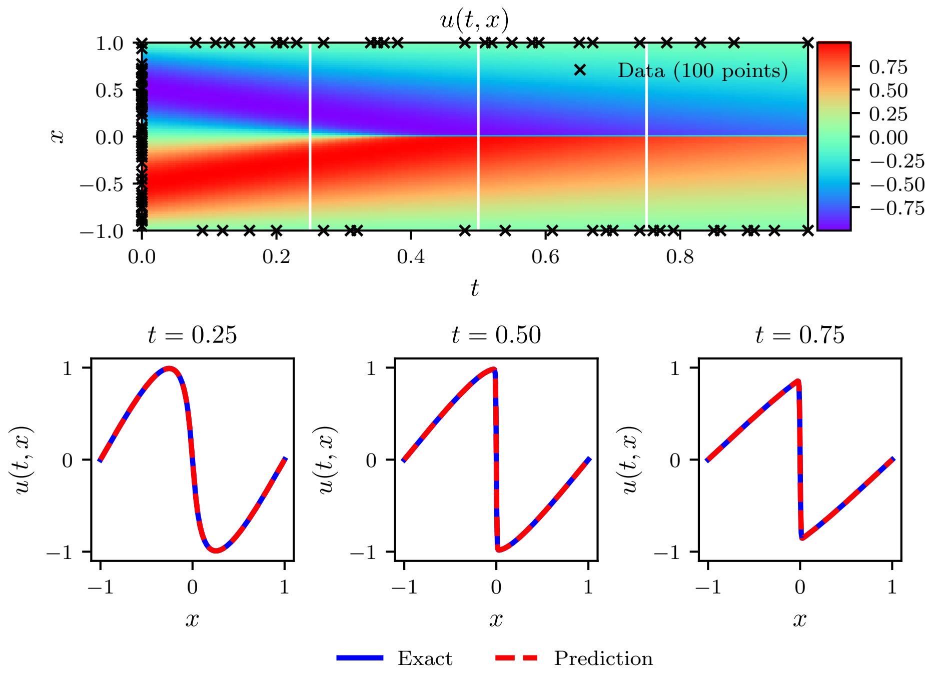


Figure 2.13: Top panel: Predicted solution $u(t,x)$ along with the initial and boundary training data. In addition we are using $10,000$ collocation points generated using a Latin Hypercube Sampling strategy. Bottom panel: Comparison of the predicted and exact solutions corresponding to the three temporal snapshots. The relative l_2-error for this case is 6.7×10^{-4}. This figure was created by Raissi et al. [12].

[30] for more details) to enable real-time, accurate predictions and handle extrapolation in multiphysics applications. It is worth noting that, loosely speaking, an operator (acting on a function space) maps functions to other functions. In contrast, a functional maps functions to numerical values, such as an integral over a function or a derivative evaluated at a specific point.

DeepONet architecture is constructed upon the universal approximation theorem for operator **Theorem 1** [31], which has been generalized in **Theorem 2** [29] to account for diverse classes of neural networks. **Theorem 1** suggests the potential for neural networks to learn nonlinear operators from data, similar to how standard neural networks learn functions from data. However, this theorem does not provide guidance on how to learn operators efficiently. The overall accuracy of neural networks can be analyzed by categorizing the total error into three primary types: approximation, optimization, and generalization errors. While the universal approximation theorem guarantees a small approximation error for a sufficiently large network, it does not address the critical issues of optimization and generalization errors, which often dominate the total error in practical applications. For a neural network to be effective, it should not only achieve low approximation error but also be easy to train (yielding low optimization error) and generalize well to unseen data (yielding low generalization error).

Theorem 1 (Universal Approximation Theorem for Operator) *Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \dots, n$, $k = 1, \dots, p$, $j = 1, \dots, m$, such that*

$$\left| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} \right| < \epsilon$$

holds for all $u \in V$ and $y \in K_2$. Here $C(K)$ is the Banach space of all continuous functions defined with $\|f\|_{C(K)} = \max_{x \in K} |f(x)|$.

Theorem 2 (Generalized Universal Approximation Theorem for Operator) *X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a nonlinear continuous operator, which maps V into $C(K_2)$—$G : V \mapsto C(K_2)$. Then, for any $\epsilon > 0$, there exists positive integers m, p, continuous vector functions $\mathbf{g} : \mathbb{R}^m \mapsto \mathbb{R}^p$, continuous vector functions $\mathbf{f} : \mathbb{R}^d \mapsto \mathbb{R}^p$, and $x_1, x_2, \dots, x_m \in K_1$, such that*

$$\left| G(u)(y) - \langle \mathbf{g}(u(x_1), u(x_2), \dots, u(x_m)), \mathbf{f}(y) \rangle \right| < \epsilon$$

holds for all $u \in V$ and $y \in K_2$. $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathbb{R}^p$. Furthermore, the functions $\mathbf{g}$ and $\mathbf{f}$ can be chosen as diverse classes of neural networks, which satisfy the classical universal approximation theorem of functions, for example, (stacked/unstacked) fully connected neural networks, residual neural networks, and convolutional neural networks (CNNs).

DeepONet is designed to achieve low total errors, as it significantly enhances generalization through a dual-subnetwork design: the branch network, which processes the input function, and the trunk network, which processes the locations at which the output function is evaluated. The core innovation is our formulation of a new operator, G, as a neural network capable of inferring quantities of interest from new, unseen data. We can interpret the nature of operator G by projecting the results of $G(u)(y)$ onto a dictionary of derivatives and applying sparse regression techniques, such as SINDy or PDE-FIND, to obtain the underlying system in the form of ordinary or partial differential equations (ODEs or PDEs). Figure 2.14 illustrates two versions of DeepONet. The stacked version is inspired by **Theorem 1**, and the unstacked version is motivated by **Theorem 2**. Note that one data point is a triplet $(u, y, G(u)(y))$, and thus a specific input u may appear in multiple data points with different values of y. $u(x)$ is generated as the value of a Gaussian random field (GRF) at a point x.

A straightforward application of DeepONets is to use them for learning an explicit operator in a ODE system. Consider the following pedagogical example:

$$\frac{\mathrm{d}s(x)}{\mathrm{d}t} = g(s(x), u(x), x); \quad x \in (0, 1].$$

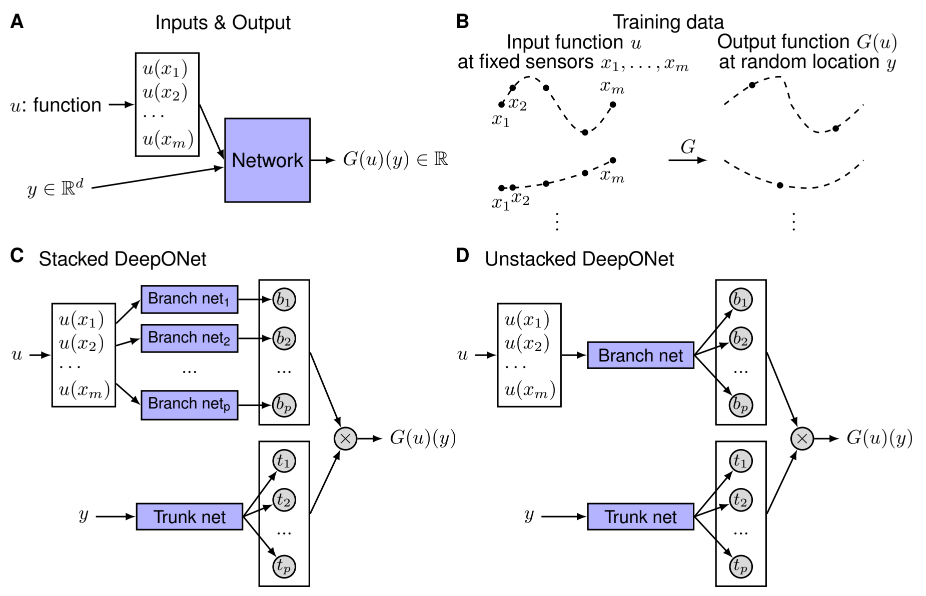


Figure 2.14: Schematic of DeepONet architectures for enhancing generalization [29]. **(A)** Operator learning setup: To enable the network to learn an operator $G : u \mapsto G(u)$, the network takes two inputs: $[u(x_1), u(x_2), \dots, u(x_m)]$ and random location y. **(B)** Training data illustration: For each input function u, we require evaluations at the same scattered sensors $x_1, x_2, \dots, x_m$. However, we impose no constraints on the number for evaluating the output functions, allowing flexibility in the output data structure. **(C)** Stacked DeepONet: The network consists of one trunk network and p stacked branch networks. The trunk network is designed as a single-layer network of width p. Each branch network having one hidden layer of width n. **(D)** Unstacked DeepONet: The network includes a single trunk network and a single branch network. The unstacked DeepONet is a special case of the stacked DeepONet, where all branch networks share the same set of parameters, reducing the model complexity while maintaining generalization performance.

Given an initial condition $s(0) = 0$, our goal is to predict $s(x)$ over the entire domain $[0, 1]$ for any input function $u(x)$. We start by considering a linear problem where we choose $g(s(x), u(x), x) = u(x)$, which is equivalent to learning the antiderivative operator:

$$G : u(x) \mapsto s(x) = s_0 + \int_0^x u(\tau) \mathrm{d}\tau.$$

In Figure 2.15, DeepONets are trained to learn the antiderivative operator, showing small generalization errors and, consequently, low test errors. The training trajectory of an unstacked DeepONet with bias is plotted in the left panel, demonstrating negligible generalization error. For both stacked and unstacked DeepONets, adding bias to the branch networks reduces both training and test errors. DeepONets with bias also display reduced uncertainty, indicating greater stability during training from random initialization. Although unstacked DeepONets exhibit slightly higher training error than stacked Deep-

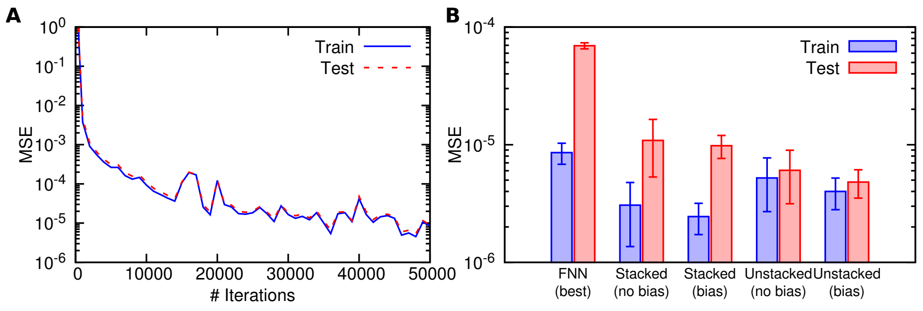


Figure 2.15: Errors in DeepONets trained to learn the antiderivative operator are examined [29]. **(A)** Shows the training progress of an unstacked DeepONet with bias. **(B)** Compares training/test errors for stacked and unstacked DeepONets, with and without bias, against the best error achieved by Feedforward Neural Networks (FNNs).

ONets, they achieve lower test error due to their smaller generalization error, making unstacked DeepONets with bias the most effective configuration. Additionally, unstacked DeepONets have fewer parameters than stacked DeepONets, allowing for faster training with reduced memory usage.

Chapter 3

Noise-aware physics-informed machine learning (nPIML) for robust data-driven discovery of PDEs

3.1 Overview of the nPIML framework

I proposed a noise-aware physics-informed machine learning (nPIML) framework [11] to close the gap in the applicability of PINNs for discovering governing PDEs when the exact terms in the PDEs are not exactly known. This approach is distinct from using PINNs to solve inverse problems, as described in the previous section. The framework is composed of three main steps: (I) derivative preparation, (II) initial PDE identification, and (III) denoising and fine-tuning using PINN. The mechanism involves using a solver network to prepare derivative candidate terms/features. Then, we apply PDE-FIND to identify the equation structure. Finally, we train PINNs with a denoising mechanism to fine-tune PDE coefficients (previously initialized by PDE-FIND), obtaining the optimal governing equation. The schematic of the nPIML framework for robust PDE discovery is illustrated in Figure 3.1.

Our new contributions are as follows: We introduce training with a preselector network to impose the weakly physics-informed constraint on the solver network. The constraint is calculable without labeled supervision. The preselector is capable of extracting feature importance scores to provide an auxiliary view for candidate selection. We also propose dPINNs (denoising PINNs) built based on discrete Fourier transform and projection networks to handle both noisy (x, t) and u, hence the noise-robustness property of the framework. While a majority of previous works only experiment with noise in u, we also consider noise in x and t, which can occur in GPS coordinate measurements [32] and manual, non-digital timing in physical experiments [33], respectively.

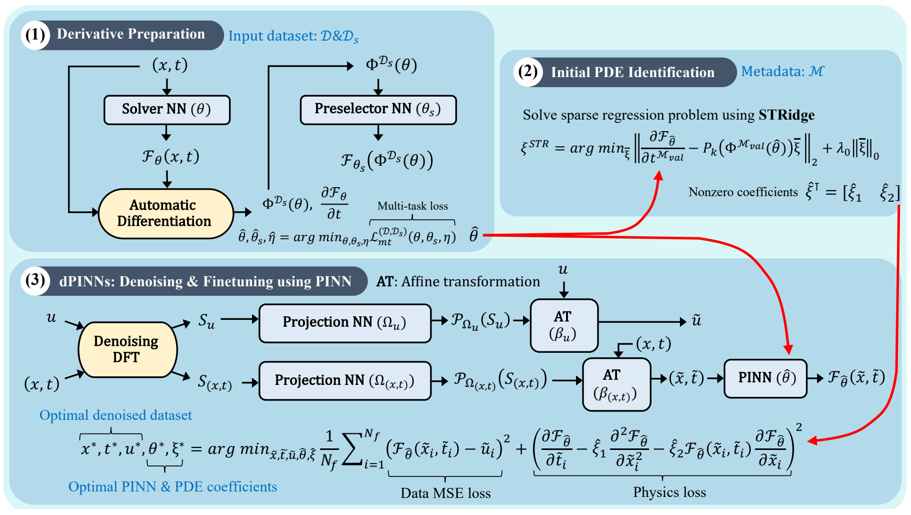

(1) Derivative Preparation
Input dataset: $\mathcal{D}\&\mathcal{D}_s$
(x,t)
Solver NN (θ)
$\mathcal{F}_\theta(x,t)$
Automatic Differentiation
$\Phi^{\mathcal{D}_s}(\theta)$
Preselector NN (θ_s)
$\mathcal{F}_{\theta_s}(\Phi^{\mathcal{D}_s}(\theta))$
$\Phi^{\mathcal{D}_s}(\theta),\ \frac{\partial \mathcal{F}_\theta}{\partial t}$
Multi-task loss
$\hat{\theta},\hat{\theta}_s,\hat{\eta} = arg\ min_{\theta,\theta_s,\eta}\mathcal{L}_{mt}^{(\mathcal{D},\mathcal{D}_s)}(\theta,\theta_s,\eta)$
$\hat{\theta}$
(2) Initial PDE Identification
Metadata: $\mathcal{M}$
Solve sparse regression problem using STRidge
$\xi^{STR} = arg\ min_{\bar{\xi}}\left\|\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial t^{\mathcal{M}_{val}}} - P_k(\Phi^{\mathcal{M}_{val}}(\hat{\theta}))\bar{\xi}\right\|_2 + \lambda_0\|\bar{\xi}\|_0$
Nonzero coefficients $\hat{\xi}^\intercal = [\hat{\xi}_1 \quad \hat{\xi}_2]$
(3) dPINNs: Denoising & Finetuning using PINN
AT: Affine transformation
u
(x,t)
Denoising DFT
S_u
$S_{(x,t)}$
Projection NN (Ω_u)
Projection NN $(\Omega_{(x,t)})$
$\mathcal{P}_{\Omega_u}(S_u)$
$\mathcal{P}_{\Omega_{(x,t)}}(S_{(x,t)})$
AT (β_u)
AT $(\beta_{(x,t)})$
$\tilde{u}$
$(\tilde{x},\tilde{t})$
PINN $(\hat{\theta})$
$\mathcal{F}_{\hat{\theta}}(\tilde{x},\tilde{t})$
Optimal denoised dataset
$x^*,t^*,u^*,\theta^*,\xi^* = arg\ min_{\tilde{x},\tilde{t},\tilde{u},\hat{\theta},\hat{\xi}}\frac{1}{N_f}\sum_{i=1}^{N_f}(\mathcal{F}_{\hat{\theta}}(\tilde{x}_i,\tilde{t}_i)-\tilde{u}_i)^2 + \left(\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial \tilde{t}_i} - \hat{\xi}_1\frac{\partial^2 \mathcal{F}_{\hat{\theta}}}{\partial \tilde{x}_i^2} - \hat{\xi}_2\mathcal{F}_{\hat{\theta}}(\tilde{x}_i,\tilde{t}_i)\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial \tilde{x}_i}\right)^2$
Optimal PINN & PDE coefficients
Data MSE loss
Physics loss

Figure 3.1: Exemplary discovery scheme of the noise-aware physics-informed machine learning (nPIML) framework: (1) physics-regularized derivative preparation by multi-task learning of the solver and preselector. (2) Initial identification of the hidden PDE by STRidge. (3) Applying the denoising DFT to $(x,t)\&u$ then finetuning the initial PDE coefficients on the denoised variables using PINN. Here, we assume that the discovered governing equation is Burgers' PDE.

We focus on the following general form of nonlinear PDE in the dynamical system perspective:

$$u_t = \mathcal{N}_\xi[\Theta];\ \Theta = \begin{bmatrix} u & u_x & u_{xx} & \cdots & x \end{bmatrix}.$$

$\mathcal{N}_\xi$ is the governing function parameterized by the vector of coefficients ξ. The function depends on Θ, which may consist of the spatial variable x, the derivatives and any indispensable terms. In regards to $\mathcal{N}_\xi$, Θ is the smallest possible library, merely composed of the necessary terms.

Figure 3.1 conceptualizes the three principal procedures for uncovering ξ preferably in a low-dimensional space by walking through an example of discovering Burgers' PDE. Step $\textcircled{\textbf{1}}$, we numerically equivalizes u and $\mathcal{N}_\xi[\Theta]$ to the solver and preselector neural network's output $\mathcal{F}_\theta(x,t)$ and $\mathcal{F}_{\theta_s}(\Phi^{\mathcal{D}_s}(\theta))$. $\Phi^{\mathcal{D}_s}(\theta) \in \mathbb{C}^{(N_f+N_r)\times C}$ is the library of C linearly independent atomic/basis candidates from which the preselector learns to embed physics by inferring the system evolution. The candidates are evaluated on a multiset $\mathcal{D}_s = \left\{(x_i,t_i)_{i=1}^{N_f+N_r}\right\}$. Step $\textcircled{\textbf{2}}$, the well-fitted networks, $\hat{\theta}$ and $\hat{\theta}_s$, put together a larger library of potential k-degree polynomial features $P_k(\Phi^{\mathcal{M}_{val}}(\hat{\theta}))$ of which an initial analytical expression of Burgers' PDE, worked out approximately by STRidge [7], is made. Step $\textcircled{\textbf{3}}$, $\hat{\theta}$ is henceforth transferred to the PINN that is optimally finetuned with the PDE, initialized by nonzero coefficients $\hat{\xi}$, on the denoised variables $\tilde{x}$, $\tilde{t}$ and $\tilde{u}$, offered by the projection networks $\mathcal{P}_{\Omega_{(x,t)}}$ and

$\mathcal{P}_{\Omega_u}$. The noise-reduction mechanism functions as a series of affine transformations, controlled by $\beta_{(x,t)}$ and β_u, of the dataset with the projected noises $\mathcal{P}_{\Omega_{(x,t)}}(S_{(x,t)})$ and $\mathcal{P}_{\Omega_u}(S_u)$, after applying the frequency-based denoising DFT.

3.1.1 Derivative preparation

The solver network ($\mathcal{F}_\theta$) is weakly physics-constrained via joint training with the preselector network ($\mathcal{F}_{\theta_s}$). Facilitating the co-training, the solver network is pretrained on the dataset $\mathcal{D} = \left\{(x_i, t_i, u_i)_{i=1}^{N_f}\right\}$ to approximate the mapping function. Therefore, the partial derivative candidate values are assured of becoming close to the valid values. At the pretraining stage, the solver network minimizes the mean square error (MSE)

$$\mathcal{L}_{sup}^{\mathcal{D}}(\theta) = \frac{1}{N_f}\sum_{i=1}^{N_f}(\mathcal{F}_\theta(x_i, t_i) - u_i)^2;\ (x_i, t_i, u_i) \in \mathcal{D}, \tag{3.1}$$

where N_f is the number of labeled subsamples. If u is complex-valued, the sum of the MSEs from the real and imaginary parts is taken as the supervised loss function. Since a futile search over infinitely feasible $\Phi^{\mathcal{D}_s}(\theta)$ setups would be intractable, we presume the ability to build an overcomplete basis candidate library given to the preselector network for deciding the predictive set of features by minimizing

$$\begin{aligned} \mathcal{L}_{unsup}^{\mathcal{D}_s}(\theta, \theta_s) &= \frac{1}{N_f + N_r}\sum_{i=1}^{N_f+N_r}(\frac{\partial \mathcal{F}_\theta}{\partial t_i} - \mathcal{F}_{\theta_s}(\Phi_i^{\mathcal{D}_s}(\theta)))^2; \\ \Phi_i^{\mathcal{D}_s}(\theta) &= \begin{bmatrix}\mathcal{F}_\theta(x_i, t_i) & \frac{\partial \mathcal{F}_\theta}{\partial x_i} & \frac{\partial^2 \mathcal{F}_\theta}{\partial x_i^2} & \cdots & x_i\end{bmatrix}, \end{aligned} \tag{3.2}$$

where N_r is the number of random unsupervised subsamples within the domain that may disjoint the supervised set $\mathcal{D}$. We attain $\mathcal{D}_s$ by fusing up the spatio-temporal measurements without supervision. Each derivative term's input is usually omitted for notational convenience. Inspired by the assumption that low-order partial derivatives are commonly included more than the higher ones, we embed the thresholded self-gated mechanism, parameterized by W^b, to the preselector forward pass, emphasizing the priority of simple models as follows:

$$\begin{aligned} \mathcal{F}_{\theta_s}(\Phi^{\mathcal{D}_s}(\theta)) &= \mathcal{F}_{\theta_s^r}(\mathcal{F}_{W^b}(\Phi^{\mathcal{D}_s}(\theta))), \\ \mathcal{F}_{W^b}(\Phi^{\mathcal{D}_s}(\theta)) &= \Phi^{\mathcal{D}_s}(\theta) \odot \mathcal{A}^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b), \\ \mathcal{A}_j^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b) &= \max(\mathcal{A}_j(\Phi^{\mathcal{D}_s}(\theta), W^b) - \mathcal{T}, 0), \\ \mathcal{A}_j(\Phi^{\mathcal{D}_s}(\theta), W^b) &= \frac{\sum_{i=1}^{N_f+N_r}\sigma(\sum_{k=1}^{C}\Phi_{ik}^{\mathcal{D}_s}(\theta)W_{kj} + b_j)}{N_f + N_r}. \end{aligned} \tag{3.3}$$

$\odot$ refers to Hadamard product (broadcast multiplication). $\mathcal{A}^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b)$ is interpreted as the thresholded vector-valued feature importance the preselector perceive. The self-gated mechanism utilizes the activation function σ to compute the expected importance of each candidate in terms of (unnormalized) probability across $N_f + N_r$ samples. Note that we only consider the real part of $\Phi^{\mathcal{D}_s}(\theta)W + b$ in the case of complex-valued PDEs. $\mathcal{T}$ is a threshold for allowing the effective basis candidates. The threshold is initialized to be surely less than the minimal candidate importance, specifically we set $\mathcal{T} = \kappa \min_j \mathcal{A}_j^{(1)}(\Phi^{\mathcal{D}_s}(\theta), W^b)$, where $0 < \kappa < 1$, before the joint gradient update at the first epoch (superscript (1)). The parameter W^b consists of $W \in \mathbb{C}^{C \times C}$ and $b \in \mathbb{C}^{1 \times C}$ (weights and biases of the linear layer), serving as the share of the preselector's parameters:

$$\theta_s = (W^b, \theta_s^r); \; \mathcal{F}_{\theta_s} = \mathcal{F}_{\theta_s^r} \circ \mathcal{F}_{W^b}. \tag{3.4}$$

Excluding W^b, the rest of the preselector network's parameters get referred to as θ_s^r. We devise $R^{\mathcal{D}_s}(\theta, W^b)$ as a L_0-regularization on $\mathcal{A}^{\mathcal{T}}$ for selecting the expressive subset with priority to lower-order candidates in favor of Occam's razor principle. The regularization, encouraging the sparse and simple preselector learned representations, reads

$$\begin{aligned} R^{\mathcal{D}_s}(\theta, W^b) = \lambda_1 \Big(& \left\| \mathcal{A}^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b) \right\|_0 \\ & + \lambda_2 \sum_{j=1}^{C} w_j \mathcal{A}_j^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b) \Big). \end{aligned} \tag{3.5}$$

w is the weighting by derivative orders directly applied to the feature importance. For instance, suppose that j^{th} basis candidate associates to the second-order derivative u_{xx}. Then we have $w_j = 2$. For nonderivative terms, we assign $w_j = 1$. λ_1 is the parameter that controls the regularization intensity. λ_2 closes the gap between the derivative orders such that the high-order derivatives are not always deselected. To practically minimize $R^{\mathcal{D}_s}(\theta, W^b)$ with $\mathcal{L}_{unsup}^{\mathcal{D}_s}(\theta, \theta_s)$ by a gradient-based optimizer, we have to overcome the obstacle that the L_0-norm is not yet readily differentiable with respect to its input vector. Unlike how [34] mask candidates (with 0 or 1) before automatic differentiation, we require the smooth approximation of L_0 for achieving the thresholded feature importance. Adapted from SL0 algorithm [35], we estimates

$$\left\| \mathcal{A}^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b) \right\|_0 \approx C - \sum_{j=1}^{C} \exp\left(\frac{-(\mathcal{A}_j^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b))^2}{2(\eta \mathbb{V}(\mathcal{A}^{\mathcal{T}}(\Phi^{\mathcal{D}_s}(\theta), W^b)))^2} \right), \tag{3.6}$$

where $\mathbb{V}$ is the unbiased variance estimator over the C basis candidates. η determines the trade-off between the accuracy and smoothness: the smaller η gives the closer approximation, and the larger η gives the smoother approximation. η is initialized at 1.0 and learned with the gradients. We now denote the differentiable regularization function as $R_\eta^{\mathcal{D}_s}(\theta, W^b)$. Combining (3.1), (3.2), (3.5) and (3.6), we view the multi-task learning of the weakly physics-informed solver and the coordinating simplicity-guided preselector inherently as the semi-supervised multi-objective optimization formulated as follows:

$$\begin{aligned} \hat{\theta}, \hat{\theta}_s, \hat{\eta} &= \underset{\theta, \theta_s, \eta}{\operatorname{argmin}} \, \mathcal{L}_{mt}^{(\mathcal{D}, \mathcal{D}_s)}(\theta, \theta_s, \eta); \\ \mathcal{L}_{mt}^{(\mathcal{D}, \mathcal{D}_s)}(\theta, \theta_s, \eta) &= MT(\mathcal{L}_{sup}^{\mathcal{D}}(\theta), \mathcal{L}_{unsup}^{\mathcal{D}_s}(\theta, \theta_s) + R_\eta^{\mathcal{D}_s}(\theta, W^b)). \end{aligned} \tag{3.7}$$

The parameters of both networks are concurrently updated with the expectancy that the preselector network distills the hidden PDE function $\mathcal{N}_\xi$, and informs physics back to the solver. MT is a function that reasonably manipulates learning by multiple losses.

3.1.2 Initial PDE identification

Depicted by $\textcircled{\textbf{2}}$ of Figure 3.1, we train STRidge [7] on top of the candidates and their polynomial features up to k degree: $P_k(\Phi^{\mathcal{M}}(\hat{\theta}))$, which is evaluated on metadata $\mathcal{M}$. For example, assume that $k = 2$, the nonconstant interaction-only polynomial features of u, u_x and u_{xx} are formed as

$$P_2(\begin{bmatrix} u & u_x & u_{xx} \end{bmatrix}) = \begin{bmatrix} u & u_x & u_{xx} & uu_x & uu_{xx} & u_x u_{xx} \end{bmatrix}. \tag{3.8}$$

The metadata $\mathcal{M} = \left\{ (x_i^{\mathcal{M}}, t_i^{\mathcal{M}})_{i=1}^{N_{\mathcal{M}}} \right\}$ can be samples from a desired domain of interest, e.g., linearly discretized samples within a bounded rectangle domain are generated with the equal spaces as follows: $\Delta x = \min_{i,j;(i \neq j)} \left| x_i - x_j \right|$ and $\Delta t = \min_{i,j;(i \neq j)} \left| t_i - t_j \right|$. If x is in a higher dimension, the equal space is computed separately for each spatial direction. In fact, naively equating $\forall i \in \left\{ 1, 2, \ldots, N_f \right\}, (x_i^{\mathcal{M}}, t_i^{\mathcal{M}}) = (x_i, t_i)$ is also viable for identifying the governing PDE as $\hat{\mathcal{N}}_{\hat{\xi}}[P_k(\Phi^{\mathcal{M}_{val}}(\hat{\theta}))\mathcal{E}]$, where $\hat{\xi}$ and $\mathcal{E}$ are found by the following selection criterion:

$$\begin{gathered} \xi^{STR} = \underset{\bar{\xi}}{\operatorname{argmin}} \left\| \frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial t^{\mathcal{M}_{val}}} - P_k(\Phi^{\mathcal{M}_{val}}(\hat{\theta}))\bar{\xi} \right\|_2 + \lambda_0 \left\| \bar{\xi} \right\|_0; \\ \lambda_0 = \mu \lambda_{STR} \varepsilon,\, E = \left\{ f_{i+1} \mid i \in \mathbb{N}_{\left\| \xi^{STR} \right\|_0} \wedge \xi^{STR}_{f_{i+1}} \neq 0 \right\}, \\ \hat{\xi} = \begin{bmatrix} \xi^{STR}_{f_1} & \cdots & \xi^{STR}_{f_{|E|}} \end{bmatrix}^{\intercal},\, \mathcal{E} = \begin{bmatrix} \boldsymbol{e}_{f_1} & \cdots & \boldsymbol{e}_{f_{|E|}} \end{bmatrix}. \end{gathered} \tag{3.9}$$

$\varepsilon = \varepsilon(P_k(\Phi^{\mathcal{M}}(\hat{\theta})))$ is the significand of the condition number (written in the scientific notation) of the polynomial candidate library. $\mathcal{M}_{val}$ is a 20% split of the full $\mathcal{M}$. For a tolerance tol, $\bar{\xi}$ is estimated by solving a relaxed λ_{STR}-regularized ridge regression problem on $P_k(\Phi^{\mathcal{M}}(\hat{\theta}))$, whose column is normalized by its L_2-norm unless noted otherwise, with hard thresholding. To attain ξ^{STR}, tol is iteratively refined with respect to different values of $\lambda_0 \propto \lambda_{STR}$ using a variable d_{tol} that initializes tol. $\mu > 0$ is assigned data-dependently. After applying STRidge, $\hat{\mathcal{N}}_{\hat{\xi}}$ is the linear combination of the effective polynomial candidates chosen by $\mathcal{E}$. $\mathbb{N}_{\left\| \xi^{STR} \right\|_0}$ denotes $\left\{ 0, 1, \ldots, \left\| \xi^{STR} \right\|_0 - 1 \right\}$. E is an indexed set, and $\boldsymbol{e}_j$ is an elementary column vector whose entries are all zero except for the nonzero j^{th} polynomial candidate. The matrix $\mathcal{E}$ reduces the dimensionality such that we focus solely on the effective candidates, which hopefully capture the ideal Θ. $\hat{\xi}$ successively stores the nonzero coefficients in ξ^{STR}. If the library is overcomplete under the evaluation on $\mathcal{M}$, there exists $\mathcal{E}$ such that $\Theta^{\mathcal{M}} \approx P_k(\Phi^{\mathcal{M}}(\hat{\theta}))\mathcal{E}$.

The pair values of $(\lambda_1, \lambda_{STR})$ are grid searched with BIC as the guidance score. The pairs whose PDEs are in agreement with the corresponding preselectors, according to **Definition 6**, are expected.

Definition 6 (Agreement) *If P_k is regarded as the candidate building function and every nonzero f_{i+1}^{th} term can be written as a polynomial of certain j^{th}*

candidates whose j^{th} is taken from the set of threshold-passing basis candidate indices $\{j \mid I_j > \frac{1}{C}\}$, we determine that the initial discovered PDE of a particular pair of $(\lambda_1, \lambda_{STR})$ is in the "agreement" with the λ_1-trained preselector network.

The likely models, from which we can voluntarily choose one as the initial discovered PDE, are conceived of being in their agreements and relatively sparse (small $\left\|\xi^{STR}\right\|_0 = |E|$) while conveying sufficiently low BIC scores defined as follows:

$$\begin{aligned} BIC(\xi^{STR}, \hat{\theta}) &= \left\|\xi^{STR}\right\|_0 \log N_{\mathcal{M}} - 2 \log \hat{L}(\xi^{STR}, \hat{\theta}); \\ \log \hat{L}(\xi^{STR}, \hat{\theta}) &= \frac{-N_{\mathcal{M}}}{2} \left(1 + \log 2\pi + \log \frac{RSS(\xi^{STR}, \hat{\theta})}{N_{\mathcal{M}}} \right), \\ RSS(\xi^{STR}, \hat{\theta}) &= \sum_{i=1}^{N_{\mathcal{M}}} \left| \frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial t_i^{\mathcal{M}}} - P_k(\Phi_i^{\mathcal{M}}(\hat{\theta}))\xi^{STR} \right|^2 . \end{aligned} \tag{3.10}$$

$\log \hat{L}(\xi^{STR}, \hat{\theta})$ is the maximized (natural) log-likelihood of the $\hat{\theta}$-produced model parameterized by ξ^{STR}. RSS denotes the real-valued residual sum of squares because the absolute value of each (complex-valued) residual term is considered. Let us mention that our BIC terminology can be treated as pseudo-BIC in the sense that the calculation compares to the estimated system evolution $\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial t^{\mathcal{M}}} \approx u_t$ not u, which the simulated solution of the initial discovered PDE should closely approximate. With that said, we can optionally calculate an real-valued RSS for a complex-valued PDE by the comparison to $\left\|\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial t^{\mathcal{M}}}\right\|_2$ instead.

We shall see that the heuristics search for the agreed PDEs with the minimal BIC score is questionable in terms of future applications, where there may be no expert to supervise an acceptable range of $(\lambda_1, \lambda_{STR})$. Nevertheless, we observe that the BIC decay rate (per one increasing candidate), e.g., $\frac{\Delta BIC}{\Delta\left\|\xi^{STR}\right\|_0}$ between different estimated PDEs from λ_{STR}-varying STRidge with a fixed λ_1, can assist as an explicit information metric, which inspires us to design the automatic complexity selection algorithm of the optimal number of effective candidates.

Pedagogically, suppose that the preferred initial PDE exemplifies Burgers' PDE; we write the effective candidate matrix concerning the training set of labeled subsamples $\mathcal{D}$ as

$$\Phi_{\mathcal{E}}^{\mathcal{D}}(\hat{\theta}) = P_k(\Phi^{\mathcal{D}}(\hat{\theta}))\mathcal{E} = \begin{bmatrix} \frac{\partial^2 \mathcal{F}_{\hat{\theta}}}{\partial x^2} & \mathcal{F}_{\hat{\theta}}(x,t)\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial x} \end{bmatrix}. \tag{3.11}$$

3.1.3 dPINNs: Denoising and finetuning using PINNs

As illustrated by $\textcircled{\textbf{3}}$ of Figure 3.1, we introduce the denoising PINNs (dPINNs) for achieving the precise recovery of PDE coefficients ξ^* under uncertainties. We take the weakly physics-constrained solver $\mathcal{F}_{\hat{\theta}}$ and the initial PDE $\hat{\mathcal{N}}_{\hat{\xi}}$ to build the dPINNs, minimizing the vigorous physics-informed loss $\mathcal{L}_{sup}^{\tilde{\mathcal{D}}}(\hat{\theta}) +$

$\mathcal{L}^{\tilde{\mathcal{D}}'}_{unsup}(\hat{\theta}, \hat{\mathcal{N}_{\xi}})$ on the denoised dataset $\tilde{\mathcal{D}} = \left\{(\tilde{x}_i, \tilde{t}_i, \tilde{u}_i)_{i=1}^{N_f}\right\}$. The physics loss is generally given by

$$\mathcal{L}^{\tilde{\mathcal{D}}'}_{unsup}(\hat{\theta}, \hat{\mathcal{N}_{\xi}}) = \frac{1}{N_f}\sum_{i=1}^{N_f}(\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial \tilde{t}_i} - \hat{\mathcal{N}_{\xi}}[(\Phi^{\tilde{\mathcal{D}}'}_{\mathcal{E}}(\hat{\theta}))_i])^2, \tag{3.12}$$

where the unsupervised set $\tilde{\mathcal{D}}' = \left\{(\tilde{x}_i, \tilde{t}_i)_{i=1}^{N_f}\right\}$ is viewed simply as the slice of $\tilde{\mathcal{D}}$ without the supervision. Let us now continue the Burgers' example, we can derive the physics-constraint as

$$\begin{aligned}\hat{\mathcal{N}_{\xi}}[(\Phi^{\tilde{\mathcal{D}}'}_{\mathcal{E}}(\hat{\theta}))_i] &= P_k(\Phi^{\tilde{\mathcal{D}}'}_i(\hat{\theta}))\mathcal{E}\hat{\xi} \\ &= \hat{\xi}_1\frac{\partial^2 \mathcal{F}_{\hat{\theta}}}{\partial \tilde{x}_i^2} + \hat{\xi}_2\mathcal{F}_{\hat{\theta}}(\tilde{x}_i, \tilde{t}_i)\frac{\partial \mathcal{F}_{\hat{\theta}}}{\partial \tilde{x}_i}.\end{aligned} \tag{3.13}$$

To continually denoise $\mathcal{D}$ during the dPINNs' learning, we subtract the transformed noises, initially precomputed by the Discrete Fourier Transform (DFT) algorithm, from both (x, t) and u. The denoising mechanism is formulated as the double affine transformations of the entire training dataset given by

$$\begin{aligned}(\tilde{x}, \tilde{t}) &= (x, t) - \beta_{(x,t)} \odot \mathcal{P}_{\Omega_{(x,t)}}(S_{(x,t)}); \quad S_{(x,t)} = (S_x, S_t), \\ \tilde{u} &= u - \beta_u \odot \mathcal{P}_{\Omega_u}(S_u),\end{aligned} \tag{3.14}$$

where $\mathcal{P}_{\Omega_{(x,t)}}$ and $\mathcal{P}_{\Omega_u}$ are the projecting functions parameterized by $\Omega_{(x,t)}$ and Ω_u, capturing the unknown noise distributions. $\beta_{(x,t)}$ and β_u are updated proportional to the unbiased standard deviations ($\sqrt{\mathbb{V}(x)}, \sqrt{\mathbb{V}(t)}$) and $\sqrt{\mathbb{V}(u)}$, controlling the relevant comparable intensity of the noise corrections. The denoising DFT algorithm, which considers power spectrum density (PSD), is meant to deduct small power frequencies components. The starting noises S_u and $S_{(x,t)}$ are obtained by limiting frequencies whose power is less than the threshold ζ. To attain the low-PSD noise for the signal $\psi \in \{x, t, u\}$, we compute the following quantities:

$$\begin{aligned}S_\psi &= \psi - DFT^{-1}(DFT^{\zeta}(\psi)); \\ DFT^{\zeta}_k(\psi) &= \begin{cases} DFT_k(\psi); & \text{if} \quad PSD_k(\psi) > \zeta \\ 0; & \text{otherwise,}\end{cases} \\ PSD_k(\psi) &= \frac{1}{N_f}\left\|DFT_k(\psi)\right\|_2^2, \\ \widetilde{PSD}_k(\psi) &= \frac{PSD_k(\psi) - \mathbb{E}(PSD(\psi))}{\sqrt{\mathbb{V}(PSD(\psi))}}, \\ \zeta &= \mathbb{E}(PSD(\psi)) + \alpha \max_k(\widetilde{PSD}_k(\psi))\sqrt{\mathbb{V}(PSD(\psi))}.\end{aligned} \tag{3.15}$$

Here, k denotes an index in the frequency domain. ζ is defined according to the α portion of the maximal normalized PSD. $\mathbb{E}$ and $\mathbb{V}$ calculates the sample mean and variance over k. We precompute $S_{(x,t)}$ and S_u since the gradients cannot flow to α.

We remark that, in the nPIML framework, noise in x and t is treated in the same manner as noise in u. Our so-called "noise-reduced" variables are optimized to minimize the physical constraint loss, rather than the ideal denoising objective, which aims to bring variables closer to their clean counterparts (a difficult task without access to the ground truth), partly because x and t, when sorted, are essentially increasing functions, leaving us little to no meaningful nonlinearity to exploit.

3.2 Experiments and discussion on different canonical PDEs

We experimented with 5 canonical PDEs, including 3 ordinary PDEs and 2 complex-valued PDEs, to investigate the accuracy and robustness of our proposed method. We present the results of $\textcircled{\textbf{1}}$ Derivative preparation and $\textcircled{\textbf{2}}$ Initial PDE discovery and discuss the regularization hyperparameter effects on finding the appropriate initial PDE. Later, we show the tolerance of $\textcircled{\textbf{3}}$ dPINNs against noise in both $(x, t)\&u$ for each PDE. Please refer to [11] for full details on the experimental setting.

In the noisy experiments, we presume that a matrix, say z, gets perturbed, right after the time of its subsampling, by the $p\%$ biased (no Bessel's correction) standard deviation (std) of Gaussian noise Z simulated as follows:

$$noise(z, p) = \frac{p \cdot std(z)}{100} \times Z;\ \forall i, j(Z_{ij} \sim \boldsymbol{\mathcal{N}}(0, 1)). \tag{3.16}$$

Suppose that 1% noise is exerted, subsampled u and (x, t) get polluted in turn with $noise(u, 1)$ and $(\frac{noise(x,1)}{\sqrt{2}}, \frac{noise(t,1)}{\sqrt{2}})$.

The metric to measure how far an estimate ξ^{est} from the ground truth ξ is $mean(\delta) \pm std(\delta)$ over all j effective coefficients in ξ^{est}. If only the correct candidates are identified, $\delta_j = \delta_j(\xi^{est}, \xi)$ is the %coefficient error (%CE):

$$\delta_j = \left|\xi_j^{est} - \xi_j\right| / \left|\xi_j\right| \times 100\%;\ j \in \left\{1, \ldots, cols(\Theta)\right\}. \tag{3.17}$$

$cols(\Theta)$ represents the number of columns of Θ.

3.2.1 Canonical PDEs

Burgers' PDE

The equation [36] arises in sub-areas of applied mathematics, such as fluid mechanics and traffic flow. We consider the following Burgers' equation dataset simulated with Dirichlet boundary conditions.

$$u_t + uu_x - \nu u_{xx} = 0; \quad \nu = \frac{0.01}{\pi},\ x \in [-1, 1],\ t \in [0, 1]. \tag{3.18}$$

The viscosity of fluid ν was set to $\frac{0.01}{\pi}$ is so small that the shock wave emerges. Spectral methods and standard finite differences are used to accurately simulate the PDE with this small viscosity.

Korteweg–De Vries (KdV) PDE

The KdV equation [37] is a nonlinear dispersive PDE for describing the motion of unidirectional shallow water surfaces. For a function $u(x,t)$ the actual form of KdV we consider is expressed as

$$u_t + 6uu_x + u_{xxx} = 0; \quad x \in [0, 50],\, t \in [0, 50]. \tag{3.19}$$

KdV was known to have soliton solutions, representing two one-way moving waves with different amplitudes. Such characteristics challenge discovery methods to distinguish and yield the sparsest governing PDE that generalizes the situation. The PDE is also an excellent prototypical example to test discovering the relatively high-order spatial derivative u_{xxx}.

Kuramoto–Sivashinsky (KS) PDE

The KS or flame equation is a chaotic nonlinear PDE with a spatial fourth-order derivative term, primarily to model the diffusive instabilities in a laminar flow. The PDE reads

$$u_t + uu_x + u_{xx} + u_{xxxx} = 0; \quad x \in [0, 100],\, t \in [0, 100]. \tag{3.20}$$

The solution was generated with an initial condition $u(x,0) = \cos\big(\frac{x}{16}\big)(1 + \sin\big(\frac{x}{16}\big))$, integrated up to the wide temporal bound of $[0, 100]$ using a spectral method. As a result, the PDE solution we obtained is chaotic and complex, making it difficult for a vanilla neural network to learn the entire solution accurately while also minimizing the residual physics loss. When encountering the whole chaotic domain of KS, the PDEs produced by STRidge could be inaccurate and unstable, especially with the complication of noise.

Quantum Harmonic Oscillator (QHO) PDE

The quantum harmonic oscillator is the Schrodinger equation with a parabolic potential $0.5x^2$. The PDE is given by

$$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0; \quad x \in [-7.5, 7.5],\, t \in [0, 4]. \tag{3.21}$$

We construct the basis candidate matrix that includes the parabolic potential.

Nonlinear Schrodinger (NLS) PDE

The nonlinear Schrodinger equation is used to study nonlinear wave propagation. The PDE and its true discretization read as

$$iu_t + \frac{1}{2}u_{xx} + u\|u\|_2^2 = 0; \quad x \in [-5, 5],\, t \in [0, \frac{\pi}{2}]. \tag{3.22}$$

We include candidate terms depending on the magnitude of the solution, e.g., $\|u\|_2^2$, which may appear in the correct identification of the dynamics of the complex-valued function.

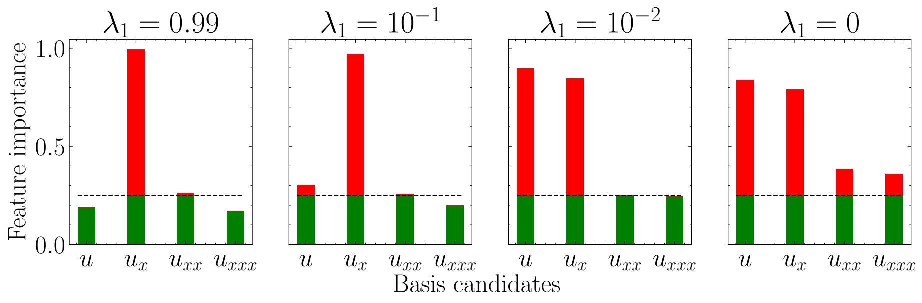


Figure 3.2: Burgers: Learned feature importance with varied λ_1

3.2.2 Effect of Regularization Hyperparameters on Initial PDE Identification

For each canonical PDE, we specify the domain of interest from which the meta-data $\mathcal{M}$ is generated for the initial PDE extraction. We then concentrate on the multi-perspective assessment of the different discovered PDEs by STRidge while varying the two major regularization hyperparameters: λ_1 of the preselector network and λ_{STR} of STRidge algorithm. Before the finetuning process, we present how accurate the initial discovered PDEs concerning the following three cases distinguished by the noise conditions: noiseless dataset, noiseless (x, t) but noisy u, and noisy $(x, t)\&u$ in which the spatial-temporal (x, t) becomes mesh-free.

Initial Discovered Burgers' PDE

We trained the preselector network with varying λ_1 to perceive the significance of each candidate. The distributed feature importance values (I_j for each j^{th} basis candidate) are presented in Figure 3.2. Although several choices of the expressive subset of passing-threshold candidates are contributed, identifying the optimal set is still not obvious by merely adjusting λ_1. Hence, STRidge was subsequently employed multiple times with diverse levels of regularization intensity λ_{STR}. For convenience, we simply set $\forall i \leqslant N_f + N_r, (x_i^{\mathcal{M}}, t_i^{\mathcal{M}}) = (x_i, t_i)$ for all Burgers' experimental cases that differed in the noise conditions. The cross results, Table 3.1, are assessed for obtaining the initial discovered PDE that agrees with the corresponding preselector and sparse with a sufficiently low BIC score.

Assigning the $\lambda_1 = 0.99$ is so high that the true candidate, i.e., u, is lacking from the passing-threshold candidates. Accordingly, the resulting PDEs cannot match the particular importance scores. The preselector properly focuses on the true candidates when λ_1 is set to 10^{-1} and 10^{-2}. Notice that u_{xx} consistently passes the threshold with marginal values, conveying the small viscosity estimates. As seen in Table 3.1, for $\lambda_1 > 0$, 10^{-2} gave the best initial result, covering the sparse PDE with the lowest BIC among the agreed models. We shall examine to find out what BIC level should be considered sufficiently low.

Deciding on the value of λ_{STR} requires an akin principle: the values that are too low or high are likely to yield incorrect forms. For example, $\lambda_{STR} = 10^0$ is immensely high, outputting the too sparse and noninformative PDE with

λ_1/λ_{STR}	10^{-6}	10^{-3}	10^{0}
0.99	$[u_{xx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-8,723.69**)	$[u_{xx}, uu_x]$ (-7,636.39)	$[uu_x]$ (15,823.14)
10^{-1}	$[u_{xx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-8,456.28**)	$[u_{xx}, uu_x]$ (-7,154.65) ✓	$[uu_x]$ (15,824.98)
10^{-2}	$[u_{xx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-8,294.55**)	$[u_{xx}, uu_x]$ (-7,178.84) **✓**	$[uu_x]$ (15,824.29)
0 (Supplement)	$[u_{xx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-8,437.81**) ✓	$[u_{xx}, uu_x]$ (-7,243.32) ✓	$[uu_x]$ (15,827.68)

Table 3.1: **Burgers regularization hyperparameter selection**: Concerning the coefficient selection criteria, STRidge's λ_0, controlling the L_0-penalty, is set to $10^4\lambda_{STR}\varepsilon$, and d_{tol} equals 2 for the three noise conditions. The assignment of $(\mu, \lambda_{STR}, d_{tol})$ is purely for gathering the likely different PDEs. Each PDE is accompanied by the "(BIC)" score. Blue indicates the agreement. **Bold** means the lowest BIC score, compared to the scores acquired by the same λ_1. Among the agreed models, we check (✓) the sparse PDEs with $\left\|\xi^{STR}\right\|_0 \leqslant 4$, which demonstrate sufficiently low BIC score. The PDE with **✓** is regarded as the initial guess.

the single effective uu_x, delivering the high BIC scores. $\lambda_{STR} = 10^{-3}$ is more suitable, suggesting the sparse models, which conform with the preselectors and offer the low BIC scores that vastly improve from those given by $\lambda_{STR} = 10^0$. Conditioned by $\lambda_1 > 0$, $u_t = 0.003063u_{xx} - 0.986174uu_x$ contains the few terms and offers the minimal BIC among the acceptable PDEs; thus, taken as our initial guess (**✓**) to be finetuned. By the disagreements, the sparsity-promoting preselectors, trained with $\lambda_1 > 0$, all entail that $\lambda_{STR} = 10^{-6}$ gives overly parameterized models with the minor improvements per the increased independent candidates. If we were to independently have the mere consideration on $\lambda_1 = 0$ or technically diminutive to a certain value, none of the basis candidates would probably get deselected, and the resulting PDEs would be all in their agreements. The justification, whether including uu_{xxx} and u_xu_{xx} worth the reduction in BIC, would turn ambiguous, though the PDE outcome by $(\lambda_1, \lambda_{STR}) = (0, 10^{-3})$: $u_t = 0.003063u_{xx} - 0.985882uu_x$ captures the ground on a par with our PDE guess (**✓**). If the preselector were not at all constructed, the concern would still persist.

To shed light on how we might achieve automatic model selection, we examine the Pareto front between the relative BIC and model complexity, as shown in Figure 3.3. The decrease in the relative BIC plateaus (with only small, insignificant improvements observed thereafter) once the true form of the Burgers' PDE is reached. These empirical numerical results suggest that the governing equation can be identified automatically by applying a knee detection algorithm to the trade-off between the relative BIC and the number of active terms in the PDE coefficient vector.

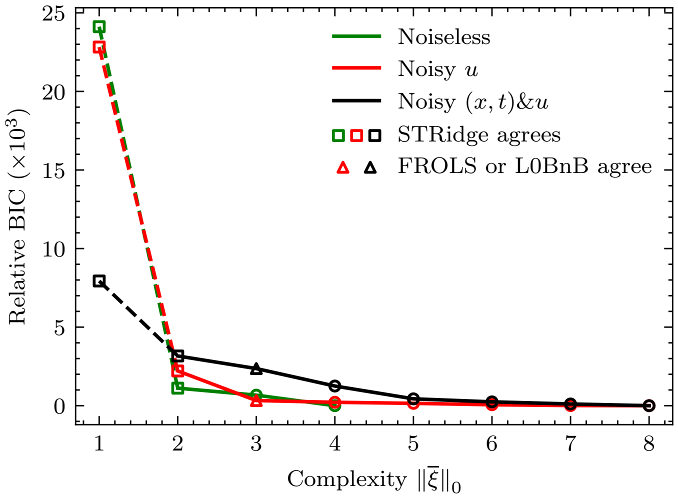


Figure 3.3: Burgers: Pareto front between the relative BIC and model complexity. The PDEs formable using the threshold-passing candidates of the preselector network trained with $\lambda_1 = 10^{-2}$.

Initial discovered KdV PDE

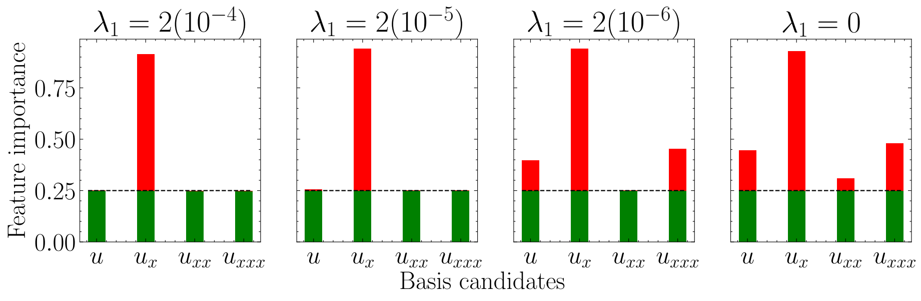


Figure 3.4: KdV: Learned feature importance with varied λ_1

We inspect how the preselector weights each basis candidate in Figure 3.4. Trained with $\lambda_1 = 2(10^{-5})$ or $2(10^{-6})$, the preselector can capture the true candidates while the relatively high value of $\lambda_1 = 2(10^{-4})$ solely let u_x pass the threshold. u and u_{xxx} barely pass the threshold if $\lambda_1 = 2(10^{-5})$, nonetheless their effectivenesses become vivid when $\lambda_1 \leqslant 2(10^{-6})$.

STRidge was leveraged multiple times on the candidate library built on $\mathcal{M}$. For KdV, we regarded the metadata as the linear discretization of the entire spatio-temporal domain; $N_{\mathcal{M}} = 64, 128$, facilitating the disambiguation of the different wave amplitudes. The found PDEs for the several pairs of $(\lambda_1, \lambda_{STR})$

λ_1/λ_{STR}	10^{-5}	10^{-3}	10^{-1}
$2(10^{-4})$	$[u_x, u_{xxx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-651,496.23**)	$[u_{xxx}, uu_x]$ (-593,260.84)	$[u_x]$ (-493,869.28)
$2(10^{-5})$	$[u_x, u_{xxx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-651,650.73**)	$[u_{xxx}, uu_x]$ (-593,259.27) ✓	$[u_x]$ (-493,885.29)
$2(10^{-6})$	$[u_x, u_{xxx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-651,782.07**)	$[u_{xxx}, uu_x]$ (-593,389.01) ✓	$[u_x]$ (-493,868.73)
0 (Supplement)	$[u_x, u_{xxx}, uu_x, uu_{xxx}, u_xu_{xx}]$ (**-651,733.37**)	$[u_{xxx}, uu_x]$ (-593,275.19) ✓	$[u_x]$ (-493,851.71)

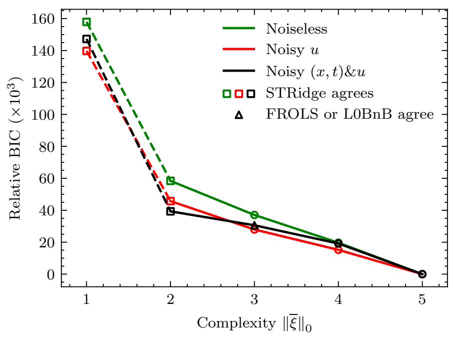
Table 3.2: **KdV regularization hyperparameter selection**: STRidge's λ_0 is set to $10^2\lambda_{STR}\varepsilon$, and d_{tol} equals 1 for the three noise conditions.

are listed in Table 3.2. By pondering the PDEs that harmonize with $\lambda_1 > 0$, we neglect the selection of the PDEs with the minimal BIC (for a particular λ_1) because they neither agree with the L_0-penalized feature importance nor be sparse as expected. The reduced BIC per an increasing effective term of transition from $\lambda_{STR} = 10^{-3}$ to $\lambda_{STR} = 10^{-5}$ is much less when compared with moving from $\lambda_{STR} = 10^{-1}$ to $\lambda_{STR} = 10^{-3}$, signifying the inefficiency of including the unnecessary terms. Remark that setting $\lambda_{STR} = 10^{-1}$ gives the PDEs, each describing a one-way traveling wave which can be considered as the relaxed form of KdV PDE, still not well fit the overall character of the dataset. Based on the mentioned justification, we thus prefer $\lambda_{STR} = 10^{-3}$, and choose the agreed PDE with the better BIC, taking the form of $u_t = -0.989065u_{xxx} - 5.961087uu_x$ as our initial guess (✓). The selected PDE is noticed as a more precise to the ground truth than the PDE based $\lambda_1 = 0$, which is $u_t = -0.988350u_{xxx} - 5.959614uu_x$. Also, just naively, the BIC cannot elucidate the overfitting hurdle without the auxiliary knowledge gained by varying $\lambda_1 > 0$.

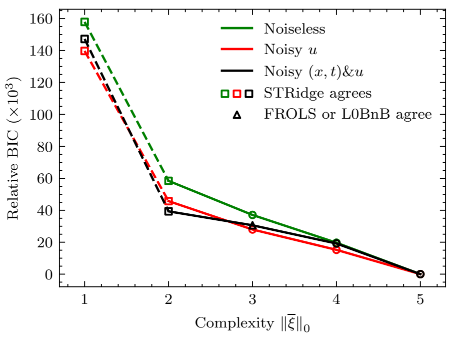
In Figure 3.5, we plot the KdV Pareto front created via the aid of the best-subset regressors: FROLS and L0BnB analogous to what is explained in the previous example. Visually, we are inclined to stop after the transition to the agreed PDE between the preselector and STRidge at an elbow point, where the BIC decay rate is minimized. This observation becomes even more pronounced when the weak-formulation representation is used with STRidge, as shown in the right panel.

Initial discovered KS PDE

In this example, we focus on on the samples from a more stable sub-region at the beginning of the evolution, where the solver can accurately approximate as indicated by the relative L_2 error plots in Figure 3.6. We assumed that the



(a) Polynomial library



(b) Weak formulation: an ensemble of 3 instances, each having $N_{\mathcal{M}} = 64,128$ integration domain centers.

Figure 3.5: KdV: Pareto front between the relative BIC and model complexity. The preselector network is trained with either $\lambda_1 = 2(10^{-5})$ or $2(10^{-6})$.

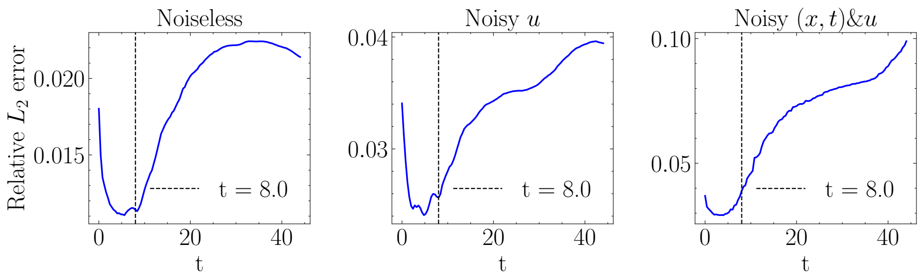


Figure 3.6: KS: Training relative L_2 error of the learned (from $N_f = 80,000$) solver $\hat{\theta}$ against temporally varying sub-regions of the KS training set bounded by $[0, 100] \times [0, 44]$, revealing a local optimum around the stability domain at the beginning of the evolution.

unknown PDE governs persistently throughout the evolution; nevertheless, the presumption does not universally hold, since specific coefficients of the chaotic behavior can be distinct over time. Based on the encountered evidences, as a result, the first 21,504 (1,024×21) discretized points within $[0, 100] \times [0, 8]$, were instead used with randomly generated nonoverlapping 10,752 unsupervised points for the training in the noiseless experiment. The temporally-wise increased number of training samples to be the first 30,000 polluted discretized points, where $t \leqslant 11.6$, were used with randomly generated disjoint 15,000 unsupervised points for both the noisy experiments. The validation sets were commonly left unaffected.

We investigate the learned feature importance of the preselector for ranking each potential atomic candidate, helping us choose the right PDE as presented in Figure 3.7. It is intriguing to discern that u_{xxxx} is one of the essential terms for every choice of λ_1, despite its order being 4, implying the possibility of including the high-order derivative.

We list the possible PDEs provided by STRidge for the various set of regularization hyperparameters in Table 3.3. The metadata was specified as the

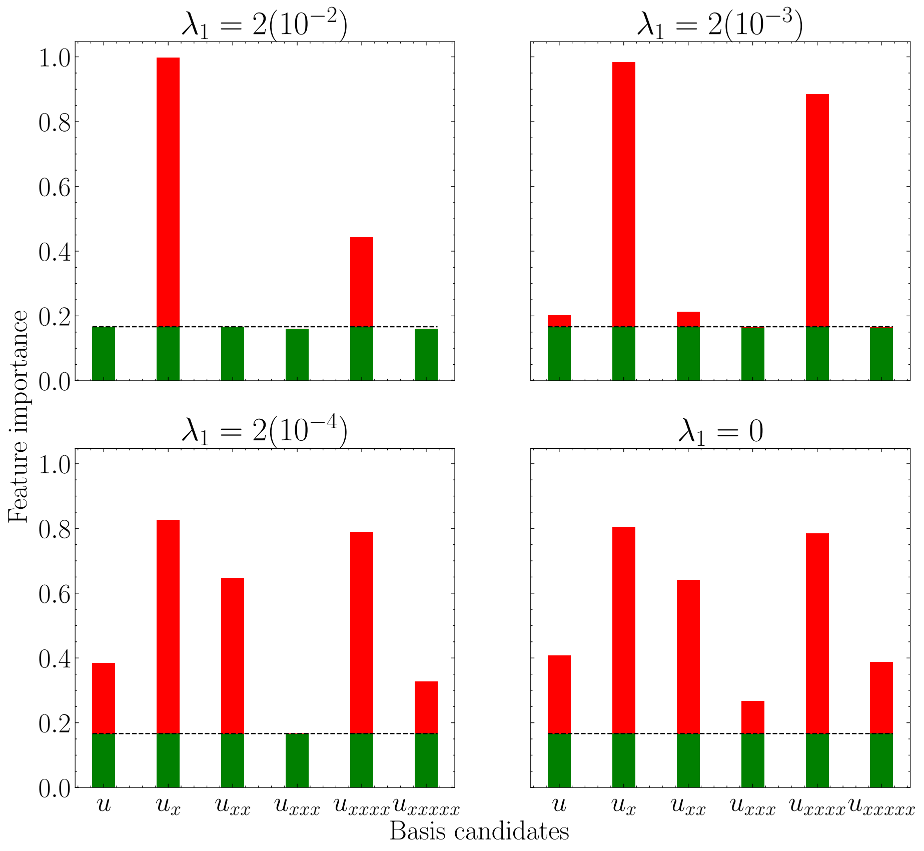


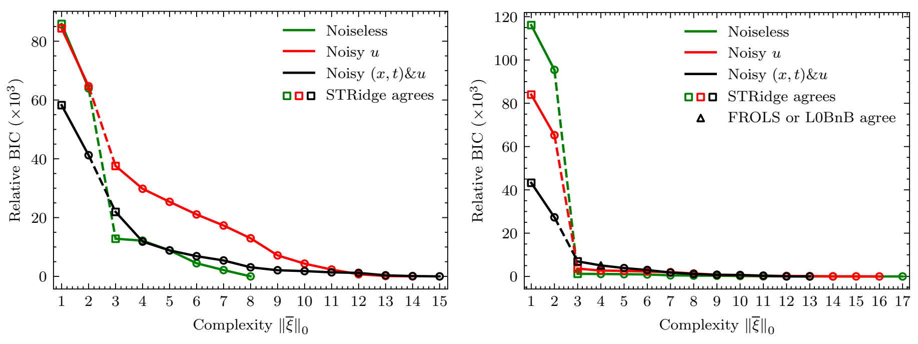
Figure 3.7: KS: Learned feature importance with varied λ_1

λ_1/λ_{STR}	10^{-5}	10^{-3}	10^{-1}
$2(10^{-2})$	$[u_{xx}, u_{xxxx}, uu_x, uu_{xxx}, uu_{xxxxx}, u_xu_{xx}, u_{xx}u_{xxx}, u_{xx}u_{xxxxx}]$ (**-153,326.24**)	$[u_{xx}, u_{xxxx}, uu_x]$ (-141,117.36)	$[uu_x]$ (-67,989.30)
$2(10^{-3})$	$[u_{xx}, u_{xxxx}, uu_x, uu_{xxx}, uu_{xxxxx}, u_xu_{xx}, u_{xx}u_{xxx}, u_{xx}u_{xxxxx}]$ (**-153,661.00**)	$[u_{xx}, u_{xxxx}, uu_x]$ (-141,032.21) **✓**	$[uu_x]$ (-67,956.02)
$2(10^{-4})$	$[u_{xx}, u_{xxxx}, uu_x, uu_{xxx}, uu_{xxxxx}, u_xu_{xx}, u_{xx}u_{xxx}, u_{xx}u_{xxxxx}]$ (**-151,328.93**)	$[u_{xx}, u_{xxxx}, uu_x]$ (-138,842.33) ✓	$[uu_x]$ (-68,022.47)
0 (Supplement)	[1]$[u_{xx}, u_{xxxx}, uu_x, uu_{xxx}, uu_{xxxxx}, u_xu_{xxxx}, u_{xx}u_{xxx}, u_{xx}u_{xxxxx}, u_{xxx}u_{xxxx}]$ (**-146,610.75**)	$[u_{xx}, u_{xxxx}, uu_x]$ (-135,102.20) ✓	$[uu_x]$ (-67,942.84)

[1]To avoid the minor details of cluttered discoveries, STRidge gets recursively reiterated with small magnitude coefficient removal until $\forall j, \left|\hat{\xi}_j\right| > 10^{-1}$.

Table 3.3: **KS regularization hyperparameter selection**: STRidge's μ is set to $(2(10^2), 5(10^3), 5(10^3))$, and d_{tol} equals $(1, 1, 50)$ for the three noise conditions. For the noisy $(x,t)\&u$ case, each polynomial candidate is normalized by its L_1-norm to get the better three-term PDE in terms of the BIC score.

21,000 samples ($N_{\mathcal{M}}$) within the $[0, 100] \times [0, 8]$ boundary generated by a Latin Hypercube Strategy. It alludes to us that the $\lambda_{STR} = 10^{-5}$ founded PDEs cannot correspond to any specified $\lambda_1 > 0$ feature importance because of the inclusion of u_{xxx}, which may be inessential. Conversely, if we were to solely contemplate on the resulting PDEs associated with $\lambda_1 = 0$, we would suspect that some terms are missing from $[u_{xx}, u_{xxxx}, uu_x]$, as the big PDE model comprising $[uu_{xxx}, uu_{xxxxx}, \ldots, u_{xxx}u_{xxxx}]$ whose coefficient magnitudes were all comparable in size, e.g., of order $> 10^{-1}$, demonstrated the lowest BIC score. The dilemma signifies that the unaided BIC, whose value varies dominantly by the changing log-likelihood term, cannot righteously balance the model complexity and accuracy, partly because no parsimonious governing PDE is involved behind the criterion assumption. In fact, the well-matched BIC is achievable by the simpler model built on the three correct candidates in $\lambda_1 = 2(10^{-3})$. We mark the correct PDE expression $u_t = -0.989019u_{xx} - 0.962360u_{xxxx} - 0.966931uu_x$ found by $\lambda_1 = 0$ as inferior to the selected model (**✓**) in terms of discovery precision. $\lambda_{STR} = 10^{-1}$ offers us the sparse PDEs, still, their BIC scores are much higher along with the clear BIC worthy enhancements observed when comparing against $\lambda_{STR} = 10^{-3}$, thus designated as the con-



(a) Polynomial library

(b) Weak formulation: an ensemble of 3 instances, each having $N_{\mathcal{M}} = 21,000$ integration domain centers.

Figure 3.8: KS: Pareto front between the relative BIC and model complexity. The preselector network is trained with $\lambda_1 = 2(10^{-3})$.

dition giving the underfitted models. We take the PDE with the lowest BIC $u_t = -0.989305u_{xx} - 0.970189u_{xxxx} - 0.978123uu_x$ as our starting PDE (✓), after assessing the agreed models for each $\lambda_1 > 0$ row. On the subsequent learning $\textcircled{\textbf{3}}$ of Figure 3.1, the first (repolluted, if noisy) 21,504 data points were employed to finetune dPINNs.

Figure 3.8 reveals that transitioning to the discovered PDE of the actual form still results in a satisfactory (and best in the KS example) BIC improvement per candidate over the optimal 2-complexity PDE built on $[uu_x, uu_{xxx}]$. After that, progress immediately stagnates (undoubtedly for the weak-formulation-based library), suggesting that 3 is the optimal number of nonzero terms to select.

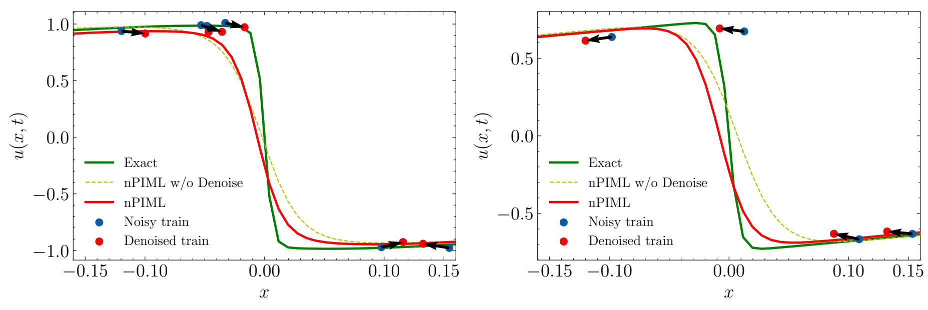
3.2.3 Finetuning PDE coefficients by dPINNs

Based on the results in Table 3.4, nPIML establishes superior results over nPIML without the denoising DFT and projection networks for the noisy cases, especially when both (x,t) and u are contaminated. For the clean dataset, the denoising mechanism seems to not over perturb backwardly through converging $\beta_{(x,t)}, \beta_u \to 0$, maintaining the effectiveness of the dPINNs' learning, on a par to the nPIML without the denoising that exactly matches the noiseless hypothesis. Indeed, nPIML can outperform nPIML without the denoisers since the shifting to the more propitious finite set, e.g., $\left\{(x_i^*, t_i^*, u_i^*)_{i=1}^{N_f}\right\}$, is still technically probable. In Burgers' example, nPIML surpasses vanilla PINN for all experimental cases regardless of the denoising modules, implying the superiority and benefits of the precomputed initialization followed by finetuning $\hat{\theta}$ and $\hat{\xi}$. Moreover, if the genuine PDE is known beforehand, training PINN from scratch eventually leads to the good close-formed discovery accuracy on a par with CWF, better than PDE-FIND (STRidge), DLrSR and WF. The accuracy enhancement points out the usefulness of automatic differentiation and physics-informed learning. Still, in KS example, IPI+CWF sets an impressive baseline error that is better than the dPINNs, which finetunes the 4th-order derivative

Dataset	Method	# Train samples (N_f)	Noiseless	u + Noise$_u$	u + Noise$_u$ & (x,t) + Noise$_{(x,t)}$
Burgers	PDE-FIND (STRidge)	256×100[1]	19.2070±19.0686	Failed ($-0.0698uu_x$)	Not applicable[2]
	DLrSR	256×100	19.2070±19.0686	Failed ($-0.0698uu_x$)	Not applicable
	WF[3]	256×100	18.7103±17.7589	18.5517±17.6983	Not applicable
	CWF	256×100	0.3135±0.2825	0.3316±0.1370	Not applicable
	CWF (Subsampled)	128×50[4]	0.5283±0.4245	0.1476±0.0220	Not applicable
	PINN[5]	3,000	0.3256±0.1921	0.9212±0.8589	4.0893±2.9622
	nPIML: IPI[6]	3,000	2.5730±1.1904	7.0093±2.6069	55.2051±15.8919
	nPIML: IPI+WF	3,000	18.5244±17.6550	19.2048±18.1213	24.9787±24.9340
	nPIML: IPI+CWF	3,000	0.7741±0.6189	0.7253±0.7113	7.8921±5.0705
	nPIML w/o Denoise[7]	3,000	0.1264±0.0605	0.4271±0.2451	2.9920±2.2222
	nPIML	3,000	**0.0557±0.0170**	**0.3360±0.1251**	**0.8546±0.4806**
KdV	PDE-FIND (STRidge)	128×501	0.5194±0.1733	Failed ($-5.4128uu_x$)	Not applicable
	DLrSR	128×501	0.5194±0.1733	Failed ($-5.3521uu_x$)	Not applicable
	WF	128×501	1.5526±0.9140	1.5529±0.8999	Not applicable
	CWF	128×501	<(0.0001±0.0001)	0.0203±0.0027	Not applicable
	CWF (Subsampled)	26×101	Failed ($-5.4090uu_x$)	Failed ($-5.4412uu_x$)	Not applicable
	nPIML: IPI	2,000	0.8710±0.2224	2.9887±1.1612[8]	3.7460±1.4158[8]
	nPIML: IPI+WF	2,000	0.9660±1.5963	1.8076±2.0034	1.4701±1.7390
	nPIML: IPI+CWF	2,000	0.2347±0.0693	**0.1177±0.1143**	0.9487±0.4943
	nPIML w/o Denoise	2,000	0.6413±0.3904	1.2547±0.8369	2.9378±1.6140
	nPIML	2,000	**0.0890±0.0568**	0.2845±0.2463	**0.4344±0.2696**
KS	PDE-FIND (STRidge)	1024×251	0.7557±0.5967	52.2843±1.4005	Not applicable
	DLrSR	1024×251	0.7571±0.5966	Failed[9]	Not applicable
	WF	1024×251	0.1521±0.0598	0.1487±0.0658	Not applicable
	CWF	1024×251	0.0004±0.0004	0.0128±0.0038	Not applicable
	CWF (Subsampled)	1024×21	Failed ($-1.2218uu_x$)	Failed ($-0.9u_{xx} - uu_x$)	Not applicable
	nPIML: IPI	⩽30,000	2.0794±0.7842	10.7558±3.3449	14.0475±4.0048
	nPIML: IPI+WF	⩽30,000	0.7265±0.5199	**0.3557±0.3068**	2.1058±1.7677
	nPIML: IPI+CWF	⩽30,000	**0.1913±0.1347**	0.5944±0.5621	**1.2546±1.488**
	nPIML w/o Denoise	⩽30,000	1.7417±1.1171	8.8925±5.2704	9.2365±6.5974
	nPIML	⩽30,000	0.4775±0.2751	2.9320±1.4401	3.6493±3.968

[1]All the discretized points are shown in the mesh representation: # in $x \times t$. [2]Because a mesh is required for taking polynomial derivatives used in PDE-FIND. [3]An instance of WF, having 10,000 integration domain centers, is used with STRidge whose $(\lambda_0, \lambda_{STR}, d_{tol}) = (10^{-5}, 10^{-2}, 5)$. [4]We subsample every 2nd point (5th point for KdV) in both x and t. For KS, we take the first 21,504 (1,024×21) discretized points. [5]$\hat{\xi}$ is initialized at $\left[exp(-7.0), 1.0\right]^\intercal$ before training PINN. [6]The results until **(2)** of Figure 3.1, Initial PDE Identification, with polynomial library. [7]The results from **(3)** of Figure 3.1, dPINNs, but without the denoising DFT module and projection networks. [8]λ_1 is assigned to $2(10^{-5})$ instead of $2(10^{-6})$. [9]DLrSR with the original and unvarying λ_0 discovers the following mismatched PDE: $u_t = -0.60uu_x - 0.39u_{xx} - 0.10uu_{xxx} - 0.49u_{xxxx}$.

Table 3.4: **Summary of the robust discovery results by nPIML**: The noise is 1% of standard deviation. Generally, the adopted λ_1s for the noisy experiments are identical to the noiseless condition unless noted otherwise. Underline and **bold** indicate the best error among the mesh-based and mesh-free methods.



(a) Snapshot around $t = 0.46$ (b) Snapshot around $t = 0.97$

Figure 3.9: Close visualization of how the projection networks react to the high noise at $x \in [-0.16, 0.16]$, around the abrupt transition caused by the shock waves.

directly and thus risks suffering from local optima of the coefficients. CWF offers the precise approximation of found coefficients, better than WF, for the mesh data with fine resolution but struggles to uncover the genuine governing PDE in the scarce subsampled mesh data. This observation naturally allows for incorporating our dPINNs' learning with the (convolutional) weak formulation to get the best from both methods in future investigations.

3.2.4 Denoising Visualization

We sought to apprehend how the projection networks respond to high noise visually by letting dPINNs expose the strongly contaminated dataset, where u and (x,t) are polluted with $noise(u,5)$ and $noise((x,t),5)$. Specifically, we finetuned the dPINNs pretrained by $\hat{\theta}$, taken from the 1%Noise+(x,t)&u case of Burgers' PDE. The initialized PDE was resolved by LS based on the intentionally uplifted 5% noisy (x,t), expressing the form as follows: $u_t = 0.000606u_{xx} - 0.403049uu_x$. For such high noise, we find it is useful that $\mathcal{P}_{\Omega_{(x,t)}}(x,t)$ and $\mathcal{P}_{\Omega_u}(u)$ should not be only activated by the final Tanh but also unbiased standardized and then scaled down to be 0.01 times the values to denoise gradually from small to larger noise magnitude since denoising the considerable amount at the beginning of the dPINNs' learning can ultimately cause the divergence. α and $(\beta'_{(x,t)}, \beta'_u)$ are initialized at 0.1 and $(10^{-3}, 10^{-3})$. We display how the projection networks denoise closely around $t = 0.46, 0.97$ in Figure 3.9. By the proximate examination near the dynamically changing region, where there are only a few supervised samples, the naive PDE estimation: $u_t = 0.012378u_{xx} - 0.948156uu_x$ neglecting the noise effect is observed when the denoising components are ablated. In comparison, the projection networks can shift the polluted samples towards the direction that drives the approximated solution by dPINNs to better captures the exact characteristics of Burgers' PDE when the denoising components are utilized. For example, the noisy samples get redirected (mostly) to the right in Figure 3.9a and left in Figure. 3.9b. With the denoising process, the optimized PDE carries the better form of $u_t = 0.008550u_{xx} - 0.972390uu_x$.

3.3 Summary

We have introduced the interpretable nPIML framework for deriving the nonlinear PDE governing a physical system as an analytical expression. The proposed method addresses challenges related to suboptimal derivatives, sensitivity to regularization hyperparameters, and noisy datasets. The weakly physics-informed solver network is the primary building block for derivative computation. Giving rise to the automatic PDE selection algorithm, multi-perspective assessment of the diverse sets of regularization hyperparameters is feasible through the physics-learning preselector network and the sparse regression. Additionally, dPINNs are introduced to fine-tune the PDE coefficients using an affine-transformed, noise-reduced dataset provided by projection networks. Numerical results demonstrate that the proposed method is both accurate and robust, even in scenarios with limited labeled samples and noisy data across five classic canonical PDEs.

Nonetheless, the proposed framework exhibits some limitations. For instance, there is no explicit denoising mechanism at the early derivative preparation and sparse regression stages; thus, particular noise of an unknown distribution may fake those initial processes and let the entire framework fail. The predicament that underlying physics remains mysterious initially causes the projection networks to be inoperable, as the affine transformation can yield the unwanted $\tilde{u} \approx \vec{0}$, and solely assigning an appropriate threshold for denoising DFT is not either trivial or readily beneficial. Towards future improvements, researchers may conduct extensive studies on grounded topics such as the effect of parameter initialization on the discovery stability or a border class of inferable PDEs that is not restricted by the linear assumption.

Chapter 4

Uncertainty-penalized Bayesian information criterion (UBIC)

4.1 Overview

Model selection from a set of potential PDEs often relies on metrics such as AIC and BIC, which evaluate model fit based on point estimates of their parameters. However, numerical results in Chapter 3 indicate that these metrics tend to favor more complex models as they minimize the information criteria, potentially leading to overfitting and the unnecessary inclusion of terms in the selected PDE. More specifically, when adjusting the number of nonzero terms in a linear model fitted to an overcomplete candidate library, AIC and BIC values typically decrease as model complexity increases, even beyond the parsimonious region.

To address this issue, we propose the uncertainty-penalized Bayesian information criterion (UBIC), a novel BIC-based metric that accounts not only for typical model complexity (the number of nonzero terms) but also for the quantified uncertainty of the estimated coefficients. Unlike traditional criteria, UBIC employs Bayesian linear regression to obtain a posterior distribution for the coefficients of each candidate PDE, offering a more robust measure of uncertainty. This method calculates the posterior mean and covariance of the coefficients and uses them to derive their coefficient of variation (CV) as a measure of *PDE uncertainty*. By integrating this uncertainty, UBIC adaptively balances the trade-off between overfitting and underfitting, enabling the identification of parsimonious and stable governing PDEs and consistently outperforming AIC and BIC.

The proposed framework begins with data denoising and progresses through identification of potential PDEs and model selection to ultimately identify the optimal PDE, as illustrated in Figure 4.1. Furthermore, the selected PDE can be validated using physics-informed (simulation-based) model selection, leveraging a PINN as a differentiable PDE solver. PINNs enable flexible solutions by embedding physical laws as learning constraints, providing a supplemental method for validating the UBIC-selected PDE by comparing its simulated state

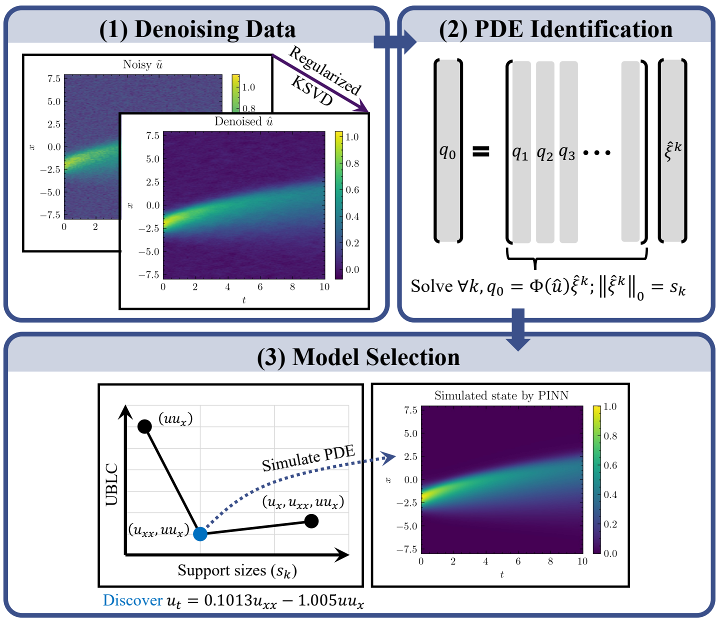


Figure 4.1: Schematic of data-driven discovery process of PDEs using UBIC

solutions against observed data.

The integration of uncertainty quantification into the model selection process through UBIC represents a significant advancement in distinguishing between overfitted and underfitted models. It offers a systematic, reliable, and efficient approach for discovering governing equations.

4.2 UBIC for data-driven identification of PDEs and ODEs

4.2.1 Problem Formulation

Let us assume without loss of generality that the system state u in a two-dimensional (2D) spatio-temporal grid of spatially distributed physical systems satisfies

$$\partial_t u = \mathcal{N}(u, \partial_x u, \partial_x^2 u, \ldots; \xi). \tag{4.1}$$

We aim to discover $\mathcal{N}$, a linear or nonlinear operator involving spatial derivatives of the state variable u only. The parametric dependency ξ is a constant vector-valued coefficient. For convenience, we consider $\partial_x u \equiv u_x$ and other notations alike. Since our observed input $\tilde{u}$ may be disturbed with noise, or mathematically given as $\tilde{u}_{ij} = u(x_i, t_j) + z(x_i, t_j)$, we begin our PDE discovery approach by denoising on $\tilde{u}$. Noise $z(x_i, t_j) \sim \frac{\epsilon \sigma_u}{100} \mathcal{Z}(0, 1)$ is drawn from standard Gaussian distribution, and scaled proportionally to $\epsilon\%$ of the standard deviation (sd) σ_u calculated over the domain.

4.2.2 Denoising Data

Suppose the spatio-temporal grid is in a 2D space, we turn $\tilde{u}$ into a zero-mean array stacking flattened patches, regarded as signals $S_p(\tilde{u}) \in \mathbb{R}^{p^2 \times f}$; where p and f determine the patch size and the number of features. We seek the dictionary $D \in \mathbb{R}^{p^2 \times c}$, whose each column is denoted by d_j, along with corresponding sparse code A to approximate $S_p(\tilde{u})$ by its sparse representation $DA \approx S_p(\tilde{u})$. To achieve the approximation, we solve the ρ-regularized dictionary learning problem:

$$\begin{aligned} \min_{D,A} \big\| S_p(\tilde{u}) - DA \big\|_F^2 + \rho \|A\|_F^2 \\ \text{subject to } \big\| d_j \big\|_2 = 1,\, j = 1, \dots, c \\ \|a_l\|_0 \le L,\, l = 1, \cdots, f. \end{aligned} \tag{4.2}$$

A couple of optimized D and A is obtain through regularized K-SVD training iterations. With the final fixed D, the ultimate sparse code A is then found using the orthogonal matching pursuit (OMP) algorithm with $\lfloor \frac{p^2}{10} \rfloor$ transforming sparsity to reconstruct the denoised observed data $\hat{u} = S_p^{-1}(DA)$ from the patches, via the inverse function S_p^{-1}. $\|\cdot\|_F$ denotes the Frobenius matrix norm. We define $\mathcal{G}_{\hat{u}}(x_i, t_j) = \hat{u}_{ij}$ as the denoised state function.

If we encounter 3D or 4D spatio-temporal data, the denoised $\hat{u}$ is instead achieved efficiently by applying 2D Savitzky-Golay filters, which can be used with SVD (singular value decomposition).

4.2.3 PDE Identification

Best-subset regression is subsequently used to recover a sequence of potential parsimonious PDEs (with their corresponding coefficients) represented by best-subset solutions with a maximal bound on support sizes (i.e., the number of nonzero terms).

We presume an overcomplete library $\Phi(\hat{u})$ collecting candidate terms (with a maximum derivative order) of the denoised observed data. The library $\Phi(\hat{u})$ is supposed to embed the information on the initial and boundary conditions. According to the weak formulation [24], i-th numerical value of j-th candidate (column-wise) in $\Phi(\hat{u}) \in \mathbb{R}^{N_\Omega \times N_q}$ is given by integrating over a local spatio-temporal subdomain Ω_i, whose (rectangular) lengths are H_x and H_t.

$$\begin{aligned} \Phi(\hat{u}) &= \begin{bmatrix} \cdots & q_j & \cdots \end{bmatrix},\, j = 1, \dots, N_q; \\ q_j^i &= \int_{\Omega_i} w \phi_j \, d\Omega,\, i = 1, \dots, N_\Omega. \end{aligned} \tag{4.3}$$

ϕ_j is regarded as a candidate function, for example, $\mathcal{G}_{\hat{u}}^2$ and $\partial_x^2 \mathcal{G}_{\hat{u}}$. N_Ω is the number of domain centers; $\forall i$, (x_i^c, t_i^c). The smooth weight, e.g., $w = (\underline{x}^2 - 1)^2(\underline{t}^2 - 1)^2$; where $\underline{x} = (x - x_i^c)/H_x, \underline{t} = (t - t_i^c)/H_t$ conditioned by $(\underline{x}, \underline{t}) \in [-1, 1]^2$, is a viable function for discovering the Burgers' PDE as it vanishes along the boundary $\partial\Omega_i$. Note that higher polynomial orders are possible. By integration by parts on Equation (4.3), numerical noisy derivative evaluation of ϕ_j is supposed to be carried out on the noiseless w instead. We use the implementation provided in the PySINDy package. Remark that the noise-tolerant representation by the convolutional weak formulation (CWF) [38] could be leveraged at the library construction stage as well.

We attain an estimate $\hat{\xi}^k$ of the PDE coefficients with its support set, $\mathrm{supp}(\hat{\xi}^k) = \{\hat{\xi}_j^k \mid \left|\hat{\xi}_j^k\right| > 0\}$ of a s_k support size (cardinality), by solving the best-subset selection problem:

$$\hat{\xi}^k = \underset{\xi^k}{\mathrm{argmin}} \left\| q_0 - \sum_{j=1}^{N_q} q_j \xi_j^k \right\|_2^2, \text{ subject to } \left\| \xi^k \right\|_0 = s_k; \tag{4.4}$$

where $q_0^i = \int_{\Omega_i} w \partial_t \mathcal{G}_{\hat{u}} \, d\Omega$ and $q_0 \approx \sum_{j=1}^{N_q} q_j \xi_j^k = \Phi(\hat{u})\xi^k$. Best-subset solvers, we experiment with to yield potential PDEs for an increasing sequence of support sizes $(s_k)_{k=1}^{N_s}$ $(N_s \leq N_q)$, are based on MIO, SOS-1-formulated (type-1 specially ordered sets) [39] MIO-SINDy, FROLS (forward regression with orthogonal least squares) [40, 41] and L0BnB (branch-and-bound framework for sparse regression) [42].

4.2.4 Uncertainty-penalized Bayesian Information Criterion (UBIC) for Adaptive Model Selection

We now find the best support size presented in Equation (4.4) within a given range. The base information criterion, on which we rely to penalize the maximized log-likelihood value of a regression model by its complexity, is the BIC:

$$\begin{aligned} \mathrm{BIC}(\hat{\xi}^k) &= -2 \log L(\hat{\xi}^k) + \log(N_\Omega) s_k; \\ \log L(\hat{\xi}^k) &= -\frac{N_\Omega}{2} \log\left(\frac{2\pi}{N_\Omega} \left\| q_0 - \Phi(\hat{u})\hat{\xi}^k \right\|_2^2 \right) - \frac{N_\Omega}{2}. \end{aligned} \tag{4.5}$$

L is the model likelihood function. Our motivating assumption is that the true governing PDE is reliable and thus parameterized by the stable vector-valued coefficient whose uncertainty is relatively lower (a good parsimony indicator) than those that characterize the other potential PDEs. Addressing the issue that the PDE with the lowest BIC is not necessarily the best PDE [11], we define the UBIC by penalizing the BIC formula by tunable quantified uncertainty as follows:

$$\begin{aligned} \mathrm{UBIC}(\xi^k, \lambda_{\mathrm{U}}) &= \mathrm{BIC}(\xi_\mu^k) + \lambda_{\mathrm{U}} \log(N_\Omega) \mathrm{U}^k \\ &= -2 \log L(\xi_\mu^k) + \log(N_\Omega)(s_k + \lambda_{\mathrm{U}} \mathrm{U}^k). \end{aligned} \tag{4.6}$$

U^k represents an estimated total uncertainty for ξ^k, scaled proportionally to $\log(N_\Omega)$, similar to the penalizing complexity in the BIC, for convenient unification. Like $L(\xi_\mu^k)$ and s_k, U^k is considered as an indicator of the PDE's parsimony. The data-dependent λ_{U} controlling influence of U^k on model selection is adaptively adjusted by **Algorithm 2**. A lower UBIC conveys a better-discovered PDE. A Bayesian linear regression probabilistic view is placed on ξ^k with Gaussian conjugate prior $\mathcal{N}(\xi^k \mid \xi_0^k, \mathrm{V}_0^k)$. Using Bayes rule for linear Gaussian systems, we derive the posterior as follows:

$$\begin{aligned}
p(\xi^k \mid \Phi(\hat{u}), q_0, \sigma_q^2) &\sim \mathcal{N}(\xi^k \mid \xi_0^k, \mathrm{V}_0^k)\mathcal{N}(q_0 \mid \Phi^k(\hat{u})\xi^k, \sigma_q^2\mathbf{I}_{N_\Omega}) \\
&= \mathcal{N}(\xi^k \mid \xi_\mu^k, \mathrm{V}^k); \\
\xi_\mu^k &= \mathrm{V}^k(\mathrm{V}_0^k)^{-1}\xi_0^k + \frac{1}{\sigma_q^2}\mathrm{V}^k\,\Phi^k(\hat{u})^T q_0, \\
\mathrm{V}^k &= \sigma_q^2(\sigma_q^2(\mathrm{V}_0^k)^{-1} + \Phi^k(\hat{u})^T\Phi^k(\hat{u}))^{-1}.
\end{aligned} \tag{4.7}$$

Later presented experimental results are produced with $\xi_0^k \in \mathbb{R}^{s_k}$ (a vector containing nonzero terms in $\hat{\xi}^k$) and $\mathrm{V}_0^k = \mathbf{I}_{s_k}$ as an identity matrix of size $s_k = \left\|\hat{\xi}^k\right\|_0$. Note that $\xi_0^k = \vec{0}$, reducing the posterior mean to ridge estimate, is also a feasible option. Columns of $\Phi^k(\hat{u})$ correspond to s_k effective candidates, which are used to calculate $\mathrm{BIC}(\xi_\mu^k)$. By maximum likelihood estimation (MLE) for the error variance, we set $\sigma_q^2 = \mathbb{E}[(q_0 - \Phi(\hat{u})\hat{\xi}^k)^2]$. Based on the obtained posterior, the uncertainty U^k is defined as follows:

$$\mathrm{U}^k = \frac{\mathrm{CV}^k}{\min_k \mathrm{CV}^k};\ \mathrm{CV}^k = \frac{\left\|\mathrm{diag}(\mathrm{V}^k)^{\circ\frac{1}{2}}\right\|_1}{\left\|\xi_\mu^k\right\|_1} = \frac{\sum_{i=j}^{s_k}\sqrt{\mathrm{V}_{ij}^k}}{\left\|\xi_\mu^k\right\|_1}. \tag{4.8}$$

We compute CV^k (the relative standard deviation) of the covariance matrix V^k by taking an element-wise square (the $\circ\frac{1}{2}$ exponent) root on its diagonal vector (diag) and then the l_1-norm division by $\left\|\xi_\mu^k\right\|_1$. When all the true terms are included, the Bayesian linear model relies on them to approximate q_0, leaving the contribution of unnecessary terms on improving the approximate error diminished and uncertain with potentially high-variance coefficients. As $\sum_{i=j}^{s_k}\sqrt{\mathrm{V}_{ij}^k}$ sums the posterior standard deviation of every effective candidate, the more unnecessary candidates get included, the more the PDE risks becoming uncertain and overfitted and getting penalized more in Equation (4.6). Each CV^k is rescaled by the minimum $\min_k \mathrm{CV}^k$, resulting in U^k whose value is comparable to s_k.

Once every U^k is obtained, we converge the UBIC by **Algorithm 2**, iteratively decreasing λ_U from its maximum bound $\lambda_\mathrm{U}^{\max}$ derived to maintain the influence of the log-likelihood value (by not overly penalizing $-2\log L(\hat{\xi}^k)$ in Equation (4.6)) for all the discovered PDEs. We compute $\lambda_\mathrm{U}^{\max}$ based on the following constraint:

$$\begin{aligned}
\forall k \le N_s,\ \log N_\Omega(s_k + \lambda_\mathrm{U}\mathrm{U}^k) &\le \left|-2\log\hat{L}(\xi_\mu^k)\right|;\ \lambda_\mathrm{U} \ge 0, \\
\lambda_\mathrm{U}^{\max} &= \max_k \frac{1}{\mathrm{U}^k}\left(\frac{2\left|\log\hat{L}(\xi_\mu^k)\right|}{\log N_\Omega} - s_k\right).
\end{aligned} \tag{4.9}$$

Algorithm 2 finds a proper $\lambda_\mathrm{U} = 10^\lambda$ by reducing λ iteratively. We track the current and competitive optimal support sizes (s_{k^*}, s_{k^c}), and test the stopping condition at line 12, which essentially checks whether we have the increased complexity with unsatisfactory improvement (see τ), or the decreased complexity with already satisfying improvement. $\tau = \tau_0$ might be included in the stopping condition by choice. Also, τ_0 can be set adaptively, yet offering

Algorithm 2 Find the optimal complexity s_{k^*} by tuning λ_{U}

```
Input: $\Phi(\hat{u}), q_0$ and $\hat{\xi}^k$
Parameter: $\tau_0$: Improvement threshold (default = 0.02) and $N_\delta$: maximum number of iterations (default = 3)
Output: The optimal support size $s_k^*$ and tuned UBIC's hyperparameter $\lambda_{\mathrm{U}}$
1: Compute $\forall k \leq N_s$, $\mathrm{V}^k, \xi_\mu^k$ and $\mathrm{U}^k$,
   with Gaussian prior $\mathcal{N}(\xi^k \mid \hat{\xi}^k, \mathbf{I}_{\|\hat{\xi}^k\|_0})$
2: Assign $\lambda \leftarrow \log_{10} \max(\lambda_{\mathrm{U}}^{\max}, 0)$ {$-\infty$ if $\lambda_{\mathrm{U}}^{\max} \leq 0$}
3: Assign $\delta \leftarrow \frac{\lambda}{N_\delta}$ and $\lambda^c \leftarrow \lambda - \delta$ {next trial value of $\lambda$}
4: Compute $\forall k$, $\mathcal{I}_k \leftarrow \mathrm{UBIC}(\xi^k, 10^\lambda)$ using $\xi_\mu^k$ and $\mathrm{U}^k$
5: Find $s_{k^*}$ where $k^* \leftarrow \mathrm{argmin}_k \mathcal{I}_k$
6: while $\lambda^c \geq 0$ do
7:    Compute $\forall k$, $\mathcal{I}_k^c \leftarrow \mathrm{UBIC}(\xi^k, 10^{\lambda^c})$
8:    Find $s_{k^c}$ where $k^c \leftarrow \mathrm{argmin}_k \mathcal{I}_k^c$
9:    Assign $\Delta s \leftarrow s_{k^c} - s_{k^*}$
10:   Assign $\Delta\mathrm{BIC} \leftarrow \mathrm{BIC}(\xi_\mu^{k^c}) - \mathrm{BIC}(\xi_\mu^{k^*})$
11:   Assign $\tau \leftarrow \tau_{k^*}^{k^c}$; $\tau_{k^*}^{k^c} = \left|\Delta\mathrm{BIC}/(\mathrm{BIC}(\xi_\mu^{k^*})\Delta s)\right|$
12:   if ($\Delta s > 0$ but ($\Delta\mathrm{BIC} > 0$ or $\tau < \tau_0$)) or
      ($\Delta s < 0$ but $\Delta\mathrm{BIC} > 0$ and $\tau > \tau_0$) then
13:      break{stopping condition detected}
14:   end if
15:   Assign $\lambda \leftarrow \lambda^c$ and $\lambda^c \leftarrow \lambda - \delta$
16:   Assign $\forall k$, $\mathcal{I}_k \leftarrow \mathcal{I}_k^c$ and $k^* \leftarrow k^c$ {$s_{k^*} \leftarrow s_{k^c}$}
17: end while
18: if $\left|\mathrm{BIC}(\xi_\mu^{k^*}) - \mathrm{BIC}(\xi_\mu^{k^*-1})\right| / \left|\mathrm{BIC}(\xi_\mu^{k^*-1})\right| < \tau_0$ then
19:    Consider an increased or decreased $\tau_0$ value to prevent overfitted or underfitted models, respectively
20: end if
21: return $s_{k^*}, \lambda_{\mathrm{U}} = 10^\lambda$ and $\forall k$, $\mathcal{I}_k$ {used in plotting}
```

the same correct selection as the default value. Such an effective heuristic is $\tau_0 = P_{75}(S)$; $S = \{\tau_{k^1}^{k^2} \mid k^1, k^2 = \mathrm{argmin}(r)$ s.t. $r = s_{k^2} - s_{k^1} > 0$, and $\forall s_{k^0} < s_{k^1}$, $\mathrm{BIC}(\xi_\mu^{k^2}) < \mathrm{BIC}(\xi_\mu^{k^1}) < \mathrm{BIC}(\xi_\mu^{k^0})\}$, the 75th percentile of successive improvement factors respecting just BIC-decreasing models. If an overfitted model is detected by line 18, we retry with a stricter percentile of S, e.g., $P_{80}(S)$. On the contrary, lessening τ_0 helps discern selecting a supposedly underfitted 1-support-size PDE. The cost of computing UBIC scores is controlled by limiting the maximum support size s_{N_s} because the best subsets for all the support sizes have to be prepared in advance.

4.3 UBIC for data-driven identification of parametric PDEs with varying coeffcients

Here, we shift our focus to a scenario where the coefficients of the governing equations are not necessarily constant. These coefficients may vary with respect to either space or time. The question arises: *How can we adapt the original formulation of the UBIC to make it applicable to this parametric PDE discovery?* To precisely pinpoint necessary changes, we will systematically revisit and refine the data-driven discovery steps to address the challenges of parametric PDEs.

4.3.1 Problem Formulation

We consider the following parametric form of governing PDEs:

$$u_t = \mathcal{F}(u, u_x, u_{xx}, \ldots; \psi(x,t)) = \sum_{j=1} \mathcal{F}_j(u, u_x, u_{xx}, \ldots)\psi_j(x,t). \tag{4.10}$$

We aim to identify the nonlinear operator $\mathcal{F}$, which involves spatial derivatives of the state variable u, whose discretization $\mathbf{U} \in \mathbb{R}^{N_x \times N_t}$ on a spatio-temporal grid is given. $\mathcal{F}$ is parameterized by $\psi(x,t)$, which we assume reducible to either $\psi(x)$ or $\psi(t)$—spatially or temporally varying functions.

4.3.2 Best-subset Regression

Suppose, without loss of generality to spatially varying cases, Equation (4.10) is formulated as systems of linear equations, with temporal dependency. Given there are N_t time steps and N_x spatial points, the linear system evaluated at a time $t = t_i$ is expressed by

$$\mathbf{U}^{\boldsymbol{i}}_{\boldsymbol{t}} = \mathbf{Q}^{\boldsymbol{i}}\boldsymbol{\xi}^{\boldsymbol{i}} = \sum_{j=1}^{N_q} \xi^i_j \boldsymbol{q}^{\boldsymbol{i}}_{\boldsymbol{j}};\ \mathbf{Q}^{\boldsymbol{i}} = \begin{pmatrix} | & | & | & | \\ \boldsymbol{q}^{\boldsymbol{i}}_{\boldsymbol{1}} & \cdots & \boldsymbol{q}^{\boldsymbol{i}}_{\boldsymbol{j}} & \cdots \\ | & | & | & | \end{pmatrix} \in \mathbb{R}^{N_x \times N_q}. \tag{4.11}$$

$\mathbf{U}_{\boldsymbol{t}}$ is the first-order time derivative numerically computed with Kalman smoothing. Every $\mathbf{Q}^{\boldsymbol{i}}$ comprises overcomplete N_q candidate terms, each term potentially serving as a true $\mathcal{F}_j$. We define the candidate library $\mathbf{Q}$ as a block-diagonal matrix constructed by all $\mathbf{Q}^{\boldsymbol{i}}$ matrices, building a single system for the parametric PDE discovery problem: $\mathbf{U}_{\boldsymbol{t}} = \mathbf{Q}\boldsymbol{\Xi}$. We solve for the solution with s_k support size (the number of nonzero terms), satisfying

$$\hat{\boldsymbol{\Xi}} = \underset{\boldsymbol{\Xi}}{\operatorname{argmin}} \sum_{i=1}^{N_t} \left\| \underline{\mathbf{U}}^{\boldsymbol{i}}_{\boldsymbol{t}} - \underline{\mathbf{Q}}^{\boldsymbol{i}}\boldsymbol{\xi}^{\boldsymbol{i}} \right\|_2^2 + \lambda \left\| \boldsymbol{\xi}^{\boldsymbol{i}} \right\|_2^2 \text{ such that } \left\| \boldsymbol{\xi}^{\boldsymbol{i}} \right\|_0 = s_k, \forall k \leq N_s; \tag{4.12}$$

where $\underline{\mathbf{U}}$ and $\underline{\mathbf{Q}}$ are the validation data on which $\boldsymbol{\Xi} \in \mathbb{R}^{N_q N_t}$ (a tall column vector collecting every $\boldsymbol{\xi}^{\boldsymbol{i}}$) is evaluated. We set $\lambda = \frac{1}{N_t}\sum_{i=1}^{N_t} \left\| \underline{\mathbf{U}}^{\boldsymbol{i}}_{\boldsymbol{t}} - \underline{\mathbf{Q}}^{\boldsymbol{i}}\boldsymbol{\xi}^{\boldsymbol{i}}_{\mathbf{LS}} \right\|_2^2 / \left\| \boldsymbol{\xi}^{\boldsymbol{i}}_{\mathbf{LS}} \right\|_2^2$; where $\boldsymbol{\xi}^{\boldsymbol{i}}_{\mathbf{LS}}$ is the least-squares solution—leveraging all of the candidate terms, to balance between the residual sum of squares (RSS) loss and the L2-norm penalty. For each time step, the best-subset solver based on mixed-integer optimization (MIOSR) [43] is used to impose sparsity, gathering $\boldsymbol{\xi^i}$ of consecutive support sizes with zero L2-norm penalty (not a sensitive hyperparameter). We prefer MIOSR over SGTR to ensure that potential PDEs with some support sizes are not overlooked. We achieve the group sparsity by controlling that the support set $\{j \mid \left|\xi^i_j\right| > 0\}$, is the same, say s_k, for every time step. Since we cannot infer the optimal number of nonzero terms solely from Equation (4.11), the model selection step is performed next.

4.3.3 Model Selection

We minimize an information criterion to select the optimal support size s^* in the strictly increasing sequence of all available support sizes, $(s_k)_{k=1}^{N_s}$. An information criterion is expressed by $-2\log L(\hat{\boldsymbol{\Xi}}) + \mathcal{C}(a_N, \hat{\boldsymbol{\Xi}}, \mathcal{P})$; where L is the likelihood function, and $\mathcal{C}(a_N, \hat{\boldsymbol{\Xi}}, \mathcal{P})$ is the total complexity penalty defined with a_N, a sequence of positive numbers. For example, $2s_k$ and $\log(N)s_k$; where $N = N_xN_t$, is the complexity penalty for AIC and BIC, respectively. $\mathcal{P}$ is any other necessary information, e.g., the complexity measures of ICOMP (informational complexity criterion) [44] or the UBIC's quantified PDE uncertainty. Considering a particular support size of s_k, we propose an extension of the original UBIC (incorporating a fixed threshold $\zeta = 10^{-5}$ to prevent underflowing) for the parametric PDE discovery as follows:

$$\begin{gathered}\text{UBIC} = N\log\left(\frac{2\pi}{N}\left\|\mathbf{U}_{\boldsymbol{t}} - \mathbf{Q}\hat{\boldsymbol{\Xi}}_{\boldsymbol{\mu}}\right\|_2^2 + \zeta\right) + \log(N)(\mathfrak{U} + s_k);\\ \mathfrak{U} = 10^{\lambda^*}\mathcal{V},\ \mathcal{V} = \frac{\text{V}}{\text{V}_{\max}},\ \text{V} = \Sigma_{i=1}^{N_t}R_i,\ \text{and}\ R_i = \frac{\Sigma_{j=1}^{s_k}\sigma_j^i}{\left\|\hat{\boldsymbol{\xi}}^{\boldsymbol{i}}_{\boldsymbol{\mu}}\right\|_1}.\end{gathered} \tag{4.13}$$

Following **Algorithm 2**, we compute the uncertainty $\mathfrak{U}$ of the s_k-support-size PDE using the tuned data-dependent λ^* and the scaled coefficient of variation $\mathcal{V}$. At each time step, V accumulates an instability ratio R_i, defined as the total posterior standard deviation divided by the L1-norm of the posterior mean coefficient vector. Both the posterior covariance matrix ($\in \mathbb{R}^{s_k\times s_k}$) and mean coefficient vector ($\in \mathbb{R}^{s_k}$) are obtained using Bayesian automatic relevance determination (ARD) regression [45]. $\text{V}_{\max}$ is the maximum value of V over all available support sizes. With the temporal (or spatial) accumulation, we essentially derive the extended UBIC for the parametric PDE discovery. After the best PDE has been decided, a PINN may be employed to simulate the state variable, on which UBIC is calculated to additionally verify the validity of the equation.

Spectral density based transformation. The validation data $\underline{\mathbf{Q}}$ in frequency space is obtained by applying discrete Fourier transformation over the temporal axis to every $\mathbf{Q}_{\boldsymbol{j}} = \begin{pmatrix} | & | & | & | \\ \boldsymbol{q^1_j} & \cdots & \boldsymbol{q^i_j} & \cdots \\ | & | & | & | \end{pmatrix} \in \mathbb{R}^{N_x\times N_t}$, and removing

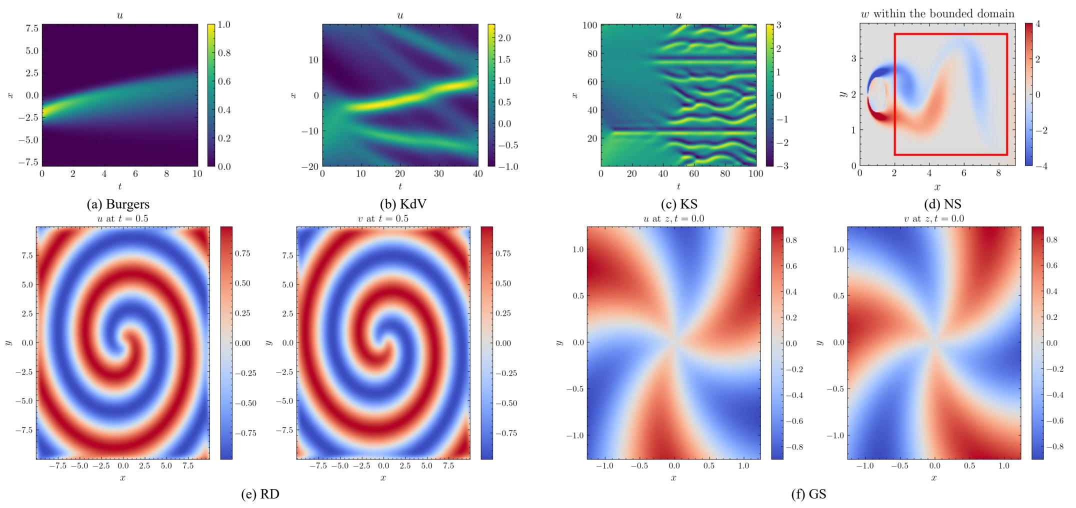


Figure 4.2: 2D visualization of the noiseless datasets

entries corresponding to low-power frequencies—less than the ninety percentile. The transformation is beneficial not only when deciding the optimal coefficient vector with s_k support size, but also when selecting the optimal s^*. We generalize the RSS to $\left\| T(\mathbf{U}_{\boldsymbol{t}}) - T(\mathbf{Q}\hat{\boldsymbol{\Xi}}_{\boldsymbol{\mu}}) \right\|_2^2$; where T transforms $\mathbf{U}_{\boldsymbol{t}}$ and $\mathbf{Q}\hat{\boldsymbol{\Xi}}_{\boldsymbol{\mu}}$ to new representations, i.e., mapping $T(\mathbf{U}_{\boldsymbol{t}}) = \tilde{\mathbf{U}}_{\boldsymbol{t}}$. Every $\tilde{\mathbf{U}}_{\boldsymbol{t}}^{\boldsymbol{i}}$ is a numerical result from a trapezoidal integration applied along the spatial axis (the frequency/temporal axis for a spatially-dependent PDE) of estimated power spectral density (PSD) using a periodogram. The integration limits the sample number and therefore facilitates the model selection step, as [11, 13] have shown that conventional information criteria tend to select overfitted PDEs when the number of samples is large. The PSD representation is noise-tolerant with its clear characteristics, exhibiting larger values for true data-generating frequencies. The integration is ablated if the estimated PSD is a one-dimensional vector.

4.4 Experiments and discussion on different canonical PDEs and ODEs

The PDE dataset description is given in Table 4.1. We plot 2D visualization of the datasets we experimented with in Figure. Experiments were run on a 2.6 GHz 6-Core Intel i7 CPU with 32 GB RAM. For reproducibility, the data and code are available at `https://github.com/Pongpisit-Thanasutives/UBIC`.

4.4.1 Burgers' PDE

We tested the PDE solution with the initial condition: $u(x, 0) = e^{-(x+2)^2}$. To denoise $\tilde{u}$, we ran regularized KSVD with $\rho = 0.05$ on the stack $S_p(\tilde{u})$ created with square patches of size 8×8. For sparse encoding during the training, OMP was configured with one target sparsity.

Table 4.1: PDE dataset descriptions. The number of discretized spatial and temporal points are specified by the (N_x, N_y, N_z) and N_t. The intensity of ϵ%-sd Gaussian noise is listed in the rightmost column.

Dataset	PDE	N_x, N_y, N_z	N_t	ϵ
Burgers	$\partial_t u = 0.1\partial_x^2 u - u\partial_x u$	256 on $[-8, 8]$	101 on $[0, 10]$	30
KdV	$\partial_t u = -\partial_x^3 u - u\partial_x u$	512 on $[-20, 20]$	501 on $[0, 40]$	30
KS	$\partial_t u = -\partial_x^2 u - \partial_x^4 u - u\partial_x u$	1024 on $[0, 32\pi]$	251 on $[0, 100]$	30
NS	$w_t = 0.01(w_{xx} + w_{yy}) - uw_x - vw_y$	325×170 on $[2, 8.48] \times [0.3, 3.68]$	151 on $[0, 30]$	1
RD	$u_t = u - u^3 + v^3 - uv^2 + u^2v + 0.1(u_{xx} + u_{yy})$ $v_t = v - u^3 - v^3 - uv^2 - u^2v + 0.1(v_{xx} + v_{yy})$	256×256 on $[-10, 10] \times [-10, 10]$	201 on $[0, 10]$	10
GS	$u_t = 0.014 - 0.014u - uv^2 + 0.02(u_{xx} + u_{yy} + u_{zz})$ $v_t = -0.067v + uv^2 + 0.01(v_{xx} + v_{yy} + v_{zz})$	$128 \times 128 \times 128$ on $[-1.25, 1.25]^3$	100 on $[0, 10]$	0.1

We gathered an overcomplete set of candidate functions, $\phi_j(\cdot) \in \{(\mathcal{G}_{\hat{u}}^{d_1} \partial_x^{d_2} \mathcal{G}_{\hat{u}})(\cdot) \mid d_1 + d_2 \geq 1; d_1, d_2 = 0, 1, 2\}$. For transforming to the integral weak forms $(\Phi(\hat{u}), q_0)$, we stick with 10000 domain centers throughout this paper. Exhaustive all subsets selection solved Equation (4.4), attaining $\hat{\xi}^k$ for every $k \leq N_q = 8$ (no constant term) in 0.29 secs (seconds).

Figure 4.3(a) shows the BIC scores of the found PDEs, where the support sizes are arranged in increasing order. After the 2-support-size PDE, the improvement in BIC becomes stagnant. However, the model selection based on BIC does not choose the 2-support-size PDE as the optimal choice. This is because the BIC scores continue decreasing beyond the plateau, and it is the 5-support-size PDE that yields the lowest BIC score. We inspect that the log-likelihood dominates the BIC score, when the number of samples in the library N_Ω is large, causing the penalization by only the support size s_k (model complexity) not strong enough for identifying the true governing PDE.

To use the proposed UBIC, we quantify the uncertainty U^k for all the best subsets, as plotted in Figure 4.3(a). These uncertainty values are incorporated to further penalize the obtained BIC scores, preferring the parsimonious PDE with the stable coefficient estimates. The PDE with a support size of 2 (2 effective candidates) exhibits the highest stability (inversely proportional to the PDE uncertainty). **Algorithm 2** suggests the UBIC scores with $\lambda_{\mathrm{U}} = 10^0 = 1$. The UBIC-selected PDE aligns with the true Burgers' PDE form.

4.4.2 Korteweg-De Vries (KdV) PDE

We generated the two-soliton u with the initial condition $u(x, 0) = -\sin\left(\frac{\pi x}{20}\right)$. We denoise using regularized KSVD with $\rho = 0.01$ on the stack $S_p(\tilde{u})$ created with 25×25 patches. We set the OMP algorithm with one target sparsity during the training.

Next, the candidate functions $\phi_j(\cdot)$ were chosen from $\{(\mathcal{G}_{\hat{u}}^{d_1} \partial_x^{d_2} \mathcal{G}_{\hat{u}})(\cdot) \mid d_1 + d_2 \geq 1; d_1 = 0, 1, 2 \text{ and } d_2 = 0, 1, 2, 3, 4\}$ before building their weak forms and q_0. We exhaustively searched for all the optimal subsets respecting every support size, achieving $\forall k \leq N_q = 14$, $\hat{\xi}^k$ in 19.04 secs.

In Figure 4.3(b), we observe that the true equation favored by the UBIC with tuned $\lambda_{\mathrm{U}} = 10^{2.28}$ stands out, in accordance with the minimal uncertainty, from the other potential PDEs.

4.4.3 Kuramoto-Sivashinsky (KS) PDE

Following the PDE-FIND paper [7], we experimented with the identical chaotic PDE generated with the initial condition: $u(x, 0) = \cos\left(\frac{x}{16}\right)\left(1 + \sin\left(\frac{x}{16}\right)\right)$. We used the same regularized KSVD settings as detailed in the KdV example.

Given that the set of candidate functions adopted in the KdV example was considered to build a weak-form library for recovering the KS PDE, we completed the same best-subset regression strategy in 32.22 secs.

As seen in Figure 4.3(c), the lowest BIC and UBIC score with $\lambda_{\mathrm{U}} = 1$ determine the true equation form.

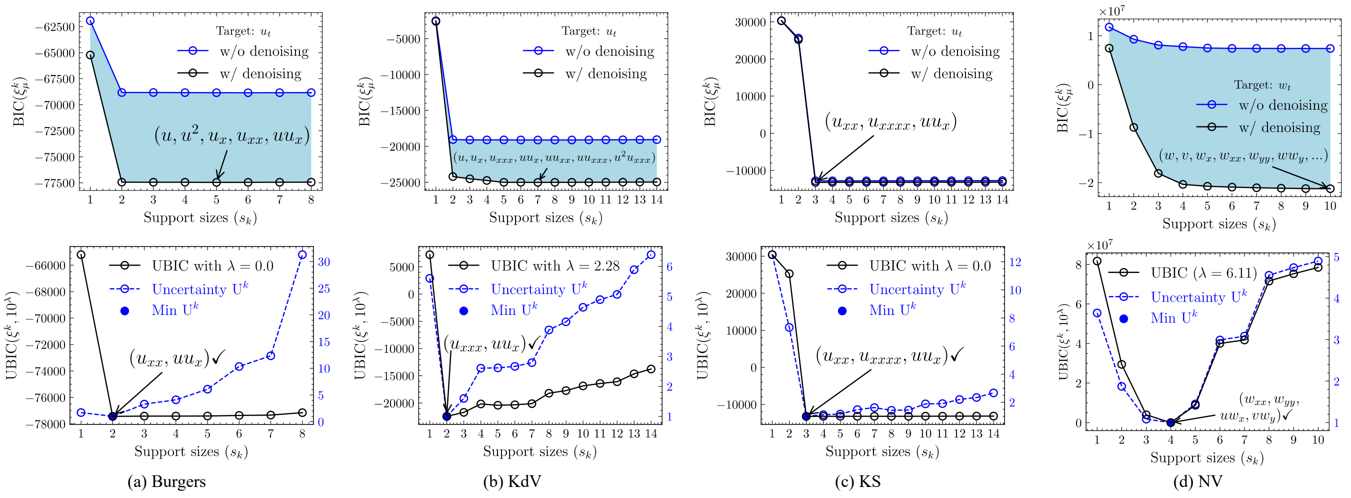


Figure 4.3: We plot the BIC, uncertainty U^k and UBIC with tuned $\lambda_\mathrm{U} = 10^\lambda$ for the model selection in the Burgers, KdV, KS, and NS examples arranged from left to right. We use an arrow ($\rightarrow$) to locate where the IC is minimized. "✓" indicates that the UBIC selects the true PDE form.

4.4.4 Navier-Stokes (NS) PDE

We consider the explicit form of the NS equation given in the 3D spatio-temporal grid. As seen in Table 4.1, w denotes the vorticity. The components of the velocity field are denoted by u (x-component) and v (y-component), which are both treated as known terms to construct an overcomplete library. We generated the dataset according to the instructions given in the PDE-FIND paper and focused on the bounded spatial domain $(x, y) \in [2, 8.48] \times [0.3, 3.68]$ after the cylinder. We were left with $N_\Omega = 8342750$ data points for each variable—w, u, and v—to which we add 1%-sd noise after the subsampling.

To obtain each noise-reduced variable, we applied 2D Savitzky-Golay filters for spatial denoising at every time step, then employed the denoising SVD. As experimented in the PDE-FIND, we specifically retained the top singular values, which were 26, 20, and 20 for w, u, and v (reshaped as metrics with 325×170 rows and 151 columns), respectively. The process enhanced the variables' quality, thereby preserving the correct equation form to be captured at the discovered 4-support-size PDE. The 19 non-weak terms included the following variables: $\psi_1 \in \{w, u, v\}$, the spatial derivatives of the vorticity w: $\psi_2 \in \{w_x, w_{xx}, w_y, w_{yy}\}$, and $\psi_1\psi_2$ (all possible polynomial interactions included). Every best subset, whose cardinality ranges from 1 to 10 was initially approximated by the MIQP (mixed-integer quadratic programming) with the l_0-norm based budget constraint [43]. The computational runtime taken was 99.48 secs. We then performed an all-subsets exhaustive search over the top candidates, each at least existing in one of the 10 best subsets.

Despite the underlying PDE form being dependent on the 4 candidate terms, it is the most stable one, as illustrated in Figure 4.3(d) (bottom). Undoubtedly, the quantified uncertainty associated with each discovered PDE is a beneficial indicator for finding the correct model by the UBIC with $\lambda_\mathrm{U} = 10^{6.11}$.

Table 4.2: The better is <u>underlined</u> (%CE) or on **bold** (R_{BIC}).

Dataset		w/o Denoising		w/ Denoising	
		%CE	R_{BIC}	%CE	R_{BIC}
Burgers		<u>0.7900</u>[1]	-6919	0.9108	$\mathbf{-12236}$
KdV		17.34	-16614	<u>9.2987</u>	$\mathbf{-22419}$
KS		0.4508	-43121	<u>0.3813</u>	$\mathbf{-43533}$
NS		False Eq.	-4379570	<u>11.43</u>	$\mathbf{-28710947}$
RD:	u_t	3.1177	-14790	<u>2.1639</u>	$\mathbf{-14963}$
	v_t	3.3251	-15729	<u>2.2967</u>	$\mathbf{-15916}$
GS:	u_t	<u>0.02621</u>	-106720	0.05542	$\mathbf{-113852}$
	v_t	0.01108	-114432	<u>0.01096</u>	$\mathbf{-120588}$

[1]In this case, we testify that the PDEs discovered by the STRidge algorithm are indecisive: $\partial_t u = -0.9482 u\partial_x u$ and $\partial_t u = 0.09918\partial_x^2 u - 1.0089 u\partial_x u$ when the l_0-penalty hyperparameter (see PDE-FIND [7]) is set to 10^{-1} and 10^{-3}, respectively. Similar results were found in the nPIML paper [11].

4.4.5 Denoising Effect

The positive effect of denoising is evident from the overall drop (shaded area) in BIC, observed throughout the previous examples. We evaluate the trade-off by the maximum reduction in the BIC: $R_{\text{BIC}} = \min_k \text{BIC}(\hat{\xi}^k_\mu) - \max_k \text{BIC}(\hat{\xi}^k_\mu)$ in Table 4.2. In Figure 4.3(c), the reduction in BIC scores is not as pronounced as what is demonstrated in the Burgers and KdV examples, implying the challenge of restoring the chaotic solution of the KS PDE. In the NS example, the omission of the true PDE when no denoising processes were performed causes a noticeable gap in the BIC values between the two cases, as illustrated in Figure 4.3(d), confirming the usefulness of the denoising step to the model selection. R_{BIC} or the area between trade-offs is a prospective metric for tuning denoising hyperparameters, possibly facilitating a clearer identification of the governing PDE.

4.4.6 Discovery Accuracy

We evaluate the proximity of each discovered $\hat{\xi}^k_j$ to the ground truth ξ_j by the percentage relative coefficient error: $100 \times \left|\hat{\xi}^k_j - \xi_j\right| / \left|\xi_j\right|$. The %CE reported in Table 4.2 denotes the average over every effective coefficient. In most of the cases, $\hat{\xi}^k$ obtained on the denoised data delivers a lower %CE than when we omit the denoising step.

4.4.7 Robust Adaptive Model Selection

We fully harness the capability of our proposed method in such extremely noisy scenarios that, without the denoising step, the true governing PDE would be omitted or poorly recovered from the best subsets. Figure 4.4 shows that we correctly identify the governing PDEs selected using the adaptive UBIC despite the severe noise interference. Particularly in the Burgers example, neither the BIC nor the uncertainty alone can recover the true equation form.

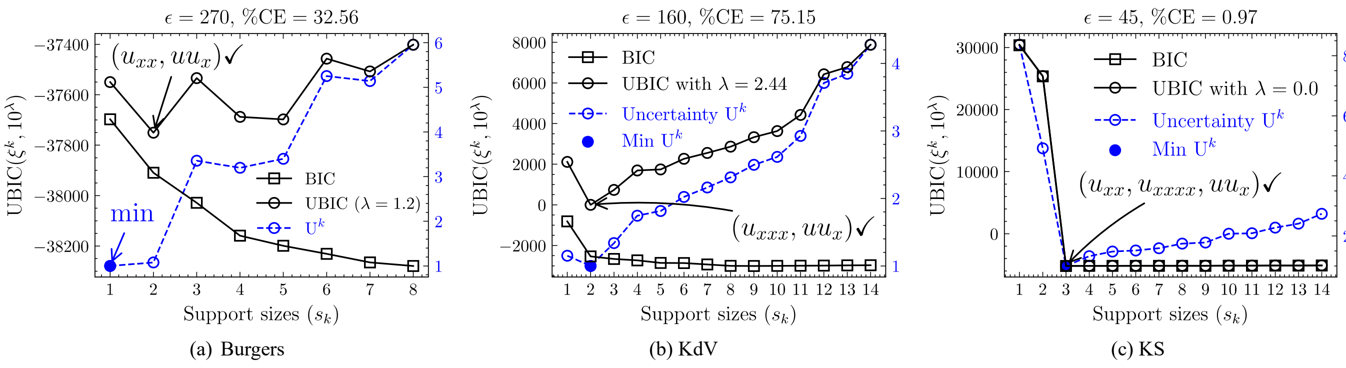


Figure 4.4: Robust adaptive model selection by the UBIC with the preceding denoising step under the extremely noisy scenarios.

We also tried ablating the denoising step to solely justify the usability of the UBIC with just the weak formulation and the adaptive model selection. As illustrated in Figure 4.5, the correct identification of the governing PDEs emphasizes the advantage of incorporating penalizing uncertainty information for the model selection.

4.4.8 Reaction-Diffusion (RD) PDE

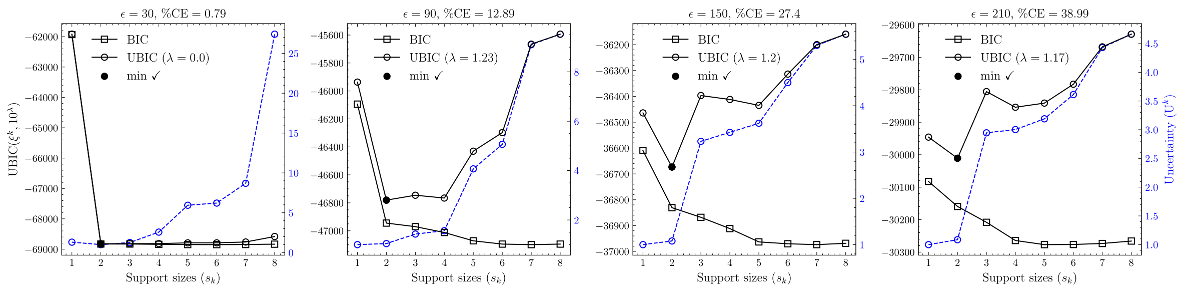
The PDE governs a system that simulates double spiral waves on a periodic domain, consisting of 7 actual terms. To countermeasure 10%-sd noise that perturbed a stack of the u and v variables, 2D Savitzky-Golay filters were employed for spatial denoising at each time step, the results were then collected to construct the noise-reduced data.

The candidate library encompassed the following variables and their transformations: $\psi_1 \in \{u, v, u^3, v^3, u^2v, uv^2\}$, the spatial derivatives (up to second-order) of either u or v: $\psi_2 \in \{u_x, u_y, u_{xx}, u_{yy}, u_{xy}, v_x, v_y, v_{xx}, v_{yy}, v_{xy}\}$, or $\psi_1\psi_2$ (polynomial interaction). To identify the best subset for each cardinality from 1 to 10, an initial approximation was obtained (in $2.56 + 2.62$ secs for u_t and v_t) using the MIQP with the budget constraint based on the l_0-norm. We then ensured the optimality of the subsets with support sizes not greater than 10, respecting the set of unique effective candidates.

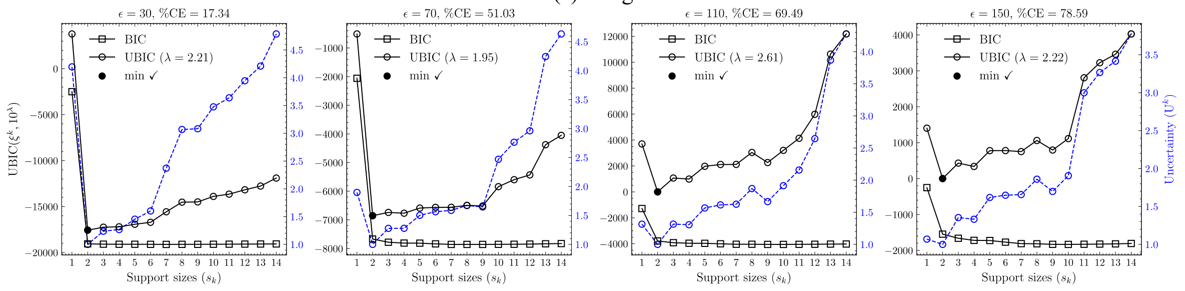
As evidenced by Figure 4.6(a), the uncertainty positively correlated with the support size, as implied by Equation (4.8). The uncertainty alone without the base BIC in Equation (4.6) is thus not enough for model selection. The 1-support-size PDE exhibits the least uncertainty. Nevertheless, an intriguing observation emerges at the 7-support-size PDE, where the uncertainty drops relatively to the surrounding PDEs plotted alongside. This local minimum is exploited by the tuned UBIC with $\lambda_{\mathrm{U}} = 10^{1.32}$ for u_t and $\lambda_{\mathrm{U}} = 10^0$ for v_t to successfully identify the 7 true candidates.

4.4.9 Gray-Scott (GS) PDE

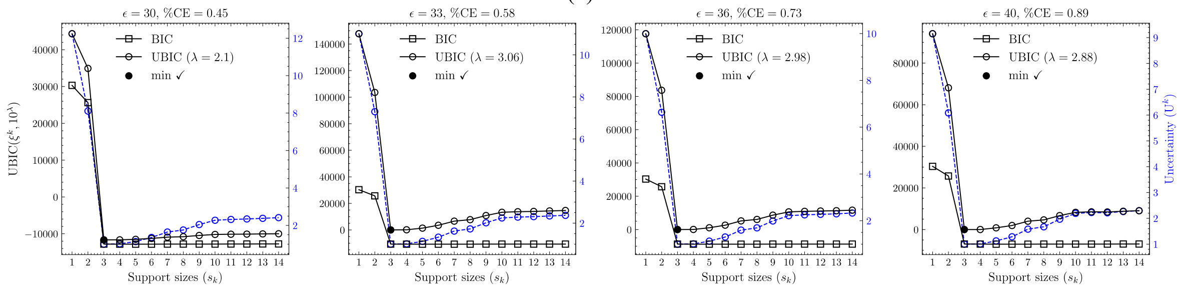
The GS PDE governs the reaction-diffusion system in the 4D spatio-temporal grid. For each variable in the noisy stack, we looped through the t-temporal



(a) Burgers



(b) KdV



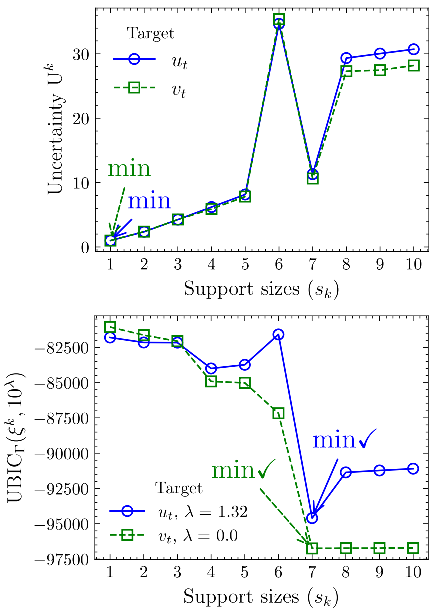
(c) KS

Figure 4.5: Robust model selection by the UBIC without the preceding denoising step under the highly noisy scenarios. For the Burgers and KS cases, we assign $\tau_0 = P_{75}(S)$. For the KdV cases, we assign $\tau_0 = P_{85}(S)$.

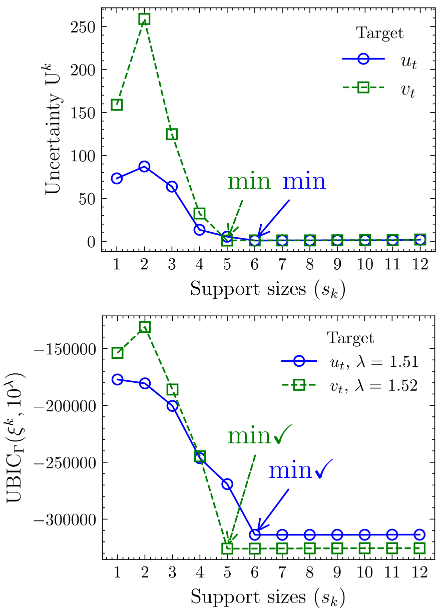
and then z-spatial axes to perform spatial denoising using 2D Savitzky-Golay filters. Hereafter, similarly to the NS example, each variable was reconstructed via the denoising SVD, retaining the 10 most significant singular values.

We comprised an overcomplete candidate library with the variables (including a constant term) and their transformations: $\{1, u, v, u^3, v^3, u^2v, uv^2\}$, the spatial derivatives of u: $\{u_x, u_y, u_z, u_{xx}, u_{yy}, u_{zz}, u_{xy}, u_{xz}, u_{yz}\}$, and the spatial derivatives of v: $\{v_x, v_y, v_z, v_{xx}, v_{yy}, v_{zz}, v_{xy}, v_{xz}, v_{yz}\}$. The best subsets were approximated (in $0.18 + 0.18$ secs for u_t and v_t) using the FROLS solver with a maximum support size of 12. These subsets were guaranteed to be at their optimum within all the effective candidates, each once delivered by the solver.

According to Figure 4.6(b), it is evident that the best subsets, which align with the true complexity of the PDE system with support sizes of 6 and 5 for u_t and v_t, exhibit the minimal uncertainty values. The tuned UBIC (with $\lambda \approx 1.51$) leverages the uncertainty pattern to penalize the BIC values, hence



(a) RD



(b) GS

Figure 4.6: Model selection results for the RD and GS PDEs.

the successful identification of the true PDE system.

4.5 Application of physics-informed neural networks for model selection

We simulated the PDE selected by the tuned UBIC and another potential PDE with an additional candidate, using the PINN learning. The PINN architecture comprised 4 hidden layers, each with 5 neurons. The learning rate parameter of the L-BFGS optimization algorithm was initialized equal to 0.1. The number of training epochs was set to 500 (taking approximately less than 1.5 hours when training on a Quadro RTX graphics processing unit with 49152 MiB memory to converge to the local optimum).

Comparing the two PDEs, we measured the proximity of their simulated solutions to the denoised observed data using BIC. Table 4.4 warrants that the s_{k^*}-support-size PDE has indeed the sufficient complexity in yielding the lower-BIC simulated state variable than its competitor, the s_{k^*+1}-support-size PDE with the dispensable candidate. Ensuring our findings, we also solve the PDEs on their entire spatio-temporal domain using the Chebfun and Dedalus software, which is unlike the PINN approach that necessitates a train-validation data split. The symbolic representation of the initial condition required by the software is recovered using the PySR package [46].

In Table 4.5, we evaluate the UBIC-selected PDE by the Frobenius and infinity matrix norms (l_F and l_∞) of the difference between its simulated solution to the noiseless PDE solution. For the Burgers and KdV PDEs, the results confirm that the simulated solution of the UBIC-selected PDE captures the

Table 4.3: Nonoverlapping train and validation domains bounded for performing the PINN-based model selection.

Dataset	$\mathcal{D}_{\text{Train}}$		$\mathcal{D}_{\text{Val}}$	
	x	t	x	t
Burgers	$[-8, 0]$	$[0, 5]$	$[0.0625, 7.9375]$	$[5.1, 10]$
KdV	$[-20, 19.84]$[1]	$[20.08, 40]$	$[-19.92, 19.92]$[2]	$[0, 20]$
KS	$[0.0982, 50.36]$	$[0, 50]$	$[50.46, 100.53]$	$[50.4, 100]$

[1]From even indices of the dataset's discretized x.
[2]From odd indices of the dataset's discretized x.

Table 4.4: Simulation-based model comparison between the PDEs with (optimal) s_{k^*} and (suboptimal) s_{k^*+1} support sizes.

Dataset		PINN[1]	Dedalus	Chebfun
Burgers	$s_{k^*} = 2$	**−16439**	**−134490**	**−134490**
	$s_{k^*+1} = 3$	−2020	−134150	−134160
KdV	$s_{k^*} = 2$	**247070**	**−732057**[2]	**−708493**[2]
	$s_{k^*+1} = 3$	280799	−729386[2]	Divergence[3]
KS	$s_{k^*} = 3$	**339036**	**783459**	**783461**
	$s_{k^*+1} = 4$	501440	811910	811906

[1]The simulated solution by PINN is evaluated on the validation set $\mathcal{D}_{\text{Val}}$ detailed in Table 4.3. [2]Before the simulation, the PDE coefficients are refitted using CWS [38] then added with a small bias value of -10^{-4}, which minimizes the resultant BIC scores. [3]The solution obtained by the spin function explodes (diverges) with a time-step of 10^{-5}.

Table 4.5: Frobenius and infinity matrix norm-based errors (l_F and l_∞) between the simulated solution of the UBIC-selected PDE and the noiseless PDE solution.

Dataset		PINN[1]	Dedalus	Chebfun	Denoised $\hat{u}$
Burgers	l_F	**0.335944**	0.481315	0.481312	2.85836
	l_∞	<u>0.293584</u>	0.485626	0.485622	2.27481
KdV	l_F	29.1799[2]	**7.21427**	11.7038	28.3743
	l_∞	36.4070[2]	<u>8.03854</u>	14.5979	41.0023
KS	l_F	27.4813	562.835	562.838	**27.1081**
	l_∞	29.4790	327.662	327.490	<u>17.8813</u>

[1]The PINN learning without optimizing $\hat{\xi}^{\kappa}$ ($\kappa = k^*$) is conducted on the entire domain. Only for the KS case, the number of neurons per layer is 50, and pretraining the network to fit the denoised observed data without minimizing any physical constraint is undergone. [2]Before the PINN-based simulation, the PDE coefficients are just refitted using CWS [38].

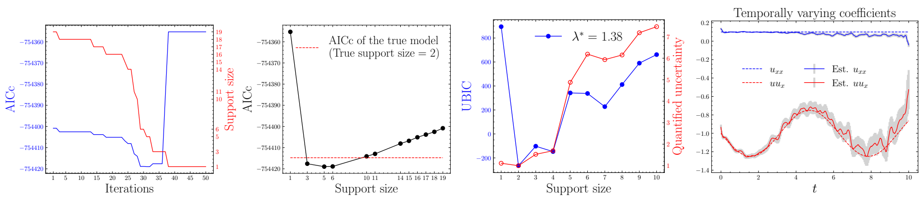
Table 4.6: Parametric PDE datasets from [47].

Dataset	PDE	Varying coefficient	N_x, N_t	ϵ
Burgers	$u_t = a(t)uu_x + 0.1u_{xx}$ $x \in [-8, 8]$ and $t \in [0, 1]$	$a(t) = -(1 + \frac{\sin(t)}{4})$	$256, 256$	4
AD	$u_t = a'(x)u + a(x)u_x + 0.1u_{xx}$ $x \in [-5, 5]$ and $t \in [0, 5]$	$a(x) = -1.5 + \cos\left(\frac{2\pi x}{5}\right)$	$256, 256$	4
KS	$u_t = a(x)uu_x + b(x)u_{xx}$ $+c(x)u_{xxxx}$ $x \in [-20, 20]$ and $t \in [0, 100]$	$a(x) = 1 + 0.25\sin\left(\frac{2\pi x}{20}\right)$ $b(x) = -1 + 0.25e^{-\frac{(x-2)^2}{5}}$ $c(x) = -1 - 0.25e^{-\frac{(x+2)^2}{5}}$	$512, 512$	10^{-2}

noiseless solution. For the KS PDE, it is more difficult to simulate the chaotic solution better than the denoised $\hat{u}$ even though the slightly inaccurate PDE coefficients (with $\%\text{CE} = 0.38$) are used. Regarding the simulation methods, the PINN-based solver, learning from both the physical constraint and the denoised observed data $\hat{u}$, performs competitively to the Dedalus or Chebfun software.

4.6 Experiments and discussion on parametric PDEs

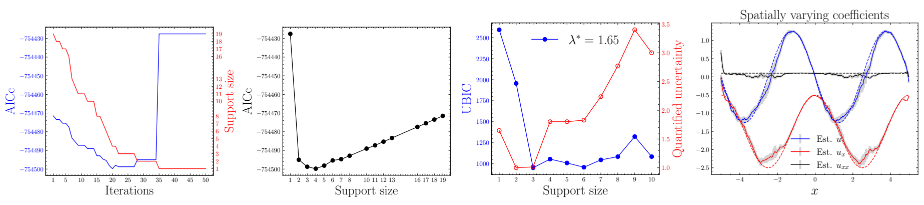
As detailed in Table 4.6, we experiment with three canonical parametric PDEs: the time-dependent Burgers' equation, the spatially-dependent advection-diffusion (AD) PDE, and the spatially-dependent chaotic Kuramoto-Sivashinsky (KS) PDE. Each entry of the noise-free simulated solution $\mathbf{U}$ is perturbed with $\epsilon\%$-sd (standard deviation) Gaussian noise sampled from $\frac{\epsilon}{100} \times \text{sd}(\mathbf{U}) \times \mathcal{N}(0, 1)$. The noise levels are listed in Table 4.6. We apply a Savitzky-Golay filter to smooth the resulting distorted data before computing partial derivatives. We refer readers to [13] for a discussion on the positive effects of the data denoising. The candidate terms include powers of u up to the cubic degree, which are multiplied by spatial derivatives of u up to the fourth order. All experiments were run on an Intel i7 CPU with 32 GB of RAM. The code is publicly available at `https://github.com/Pongpisit-Thanasutives/parametric-discovery`.



(a) SGTR (Baseline) (b) Ours (c) Identified coefficients

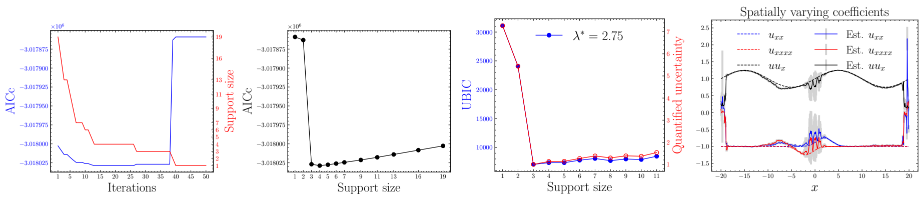
Figure 4.7: Temporal dependent Burgers' PDE. In (b), $\lambda = 1.38$.

We compare our method against the widely adopted SGTR baseline, which evaluates models using corrected AIC (AICc) for finite sample size, under noisy situations. In Figures 4.7(a), 4.8(a), and 4.9(a), although SGTR converges, it fails to explore certain support sizes, including the true one of the Burgers' PDE,



(a) SGTR (Baseline) (b) Ours (c) Identified coefficients

Figure 4.8: Spatially-denpendent Advection-diffusion PDE. Dashed lines in (c) denote the true spatially varying coefficients. In (b), $\lambda = 1.65$.



(a) SGTR (Baseline) (b) Ours (c) Identified coefficients

Figure 4.9: Spatially-dependent chaotic Kuramoto-Sivashinsky. In (b), $\lambda = 2.75$.

raising concerns about how SGTR imposes sparsity through hard thresholding. The AICc losses has led to the selection of too complex or overfitted models. In contrast, our UBIC, calculated with the PSD-based transformation, utilizes the quantified uncertainty to penalize overfitted models and identifies the correct governing equations despite the high noise levels, consistently outperforming the SGTR baseline, as shown in Figures 4.7(b), 4.8(b), and 4.9(b). Following the model selection results by our UBIC, Figures 4.7(c), 4.8(c), and 4.9(C) present the posterior coefficients with twice their standard deviation, representing about the 95% confidence intervals. These intervals demonstrate regions (in space or time) where the instability in estimating the posterior coefficients is relatively high, offering insights that can further improve the PDE discovery method by circumventing these unstable regions.

4.7 Summary

We extend the BIC to the parameter-adaptive UBIC, which accounts for model uncertainty in selecting the governing PDE from noisy data. The quantified uncertainty of the posterior model-coefficient vector enhances reliability in model selection, preventing the overfitting issues that the BIC may overlook. Leveraging the derived analytical posterior, the proposed UBIC rapidly identifies the true underlying equation, efficiently completing the task. To validate consistency in model selection, we compare the PDE chosen by the UBIC with that of a competitor PDE, including an additional candidate, to select the PDE with the lower BIC value calculated between the PINN-simulated state and the denoised observed data. Finally, we demonstrate that PDE discovery from denoised data effectively improves the BIC trade-off.

The extension of the vanilla UBIC is also proposed for identifying governing parametric PDEs. The extended UBIC, computed using the PSD-based transformation, leverages accumulated PDE uncertainty to address the overfitting problem in the model selection step, disambiguating the true governing parametric PDE from overfitted PDEs containing unnecessary candidate terms. The ability to compute confidence intervals for varying coefficients enhances the interpretability of potential models, providing comprehensive insights into their stability.

Chapter 5

Improving PINNs by adversarial multi-task training

5.1 Overview

A natural extension of the vanilla PINN is the incorporation of multi-task learning. While the traditional PINN is designed to solve a single instance of a PDE, the coefficients of the PDE can often be parameterized and varied. In my earlier work [48], I proposed a method that enables PINNs to handle multiple realizations of a PDE simultaneously. The multi-task approach captures how changes in coefficients influence the solution, enhancing the network's generalization capabilities, as later shown by Figure 5.4. By solving multiple realizations concurrently, the method has been shown to reduce errors, particularly in dynamically changing regions.

5.2 Adversarial multi-task enhanced physics-informed loss learning

5.2.1 Solving single forward partial differential equation

We consider the general form of PDEs as follows

$$\boldsymbol{\mathcal{N}}[u(t,x);\lambda] = 0, x \in \mathbb{R}^D, t \in \mathbb{R} \tag{5.1}$$

where $u(t,x)$ denotes the latent solution and $\boldsymbol{\mathcal{N}}[u(t,x);\lambda]$ is the underlying partial differential equation which is parameterized by λ. The typical physics-informed loss function [12] for solving PDE includes both the underlying equation and a boundary condition, given by

$$\mathcal{L}_{eq} = \frac{1}{N_f}\sum_{i=1}^{N_f}\Big|\boldsymbol{\mathcal{N}}[\hat{u}(t_f^i, x_f^i;\theta);\lambda]\Big|^2 + \frac{1}{N_b}\sum_{j=1}^{N_b}\Big|\hat{u}(t_b^j, x_b^j;\theta) - u(t_b^j, x_b^j)\Big|^2 \tag{5.2}$$

where $\hat{u}(t_f, x_f; \theta)$ and $\hat{u}(t_b, x_b; \theta)$ together denote the predictions of a PDE solver network, parameterized by θ, for the entire input space. N_f denotes the number of interior collocation points. N_b denotes the data points for learning the initial and boundary condition.

5.2.2 Auxiliary task generation

By setting $\lambda^{aux} = \alpha\lambda$, we are able to acquire an auxiliary task to be learned jointly. In practice, a good choice of α is achievable via either arbitrary assignment by human or Bayesian optimization to assuredly enforce the similar solution behavior by minimizing the solver network loss, which we shall discuss shortly. After acquiring λ^{aux} for each generated auxiliary task, we turn our focus to how the solver network can learn from multiple physics-informed objectives based on two effective multi-task learning strategies, uncertainty-weighted loss [16] and gradient surgery [17], namely PCGrad update rule. In order to apply a supervised weighting scheme, we proceed by the assumption that there exists a close neural network approximation $\hat{u}(t_f, x_f; \theta)$ at the desired accuracy or equivalently $\left\|\boldsymbol{\mathcal{N}}[\hat{u}(t_f, x_f; \theta); \lambda]\right\| < \epsilon$.

5.2.3 Uncertainty-weighted physics-informed loss function

We proceed by replacing the squared losses in [16] with the typical losses for solving PDEs. We define our uncertainty-weighted physics-informed loss function (Uncert) as follows

$$\mathcal{L}_{uncert} = \sum_{i=1}^{N_T} \left(\frac{1}{2\sigma_i^2}\mathcal{L}_{eq}^i + \log \sigma_i\right) \tag{5.3}$$

where N_T refers to the number of tasks. Here, σ_i is a gradient-based trainable (e.g. by utilizing $\nabla_{\sigma_i}\mathcal{L}_{uncert}$) parameter which indicates each PDE solution uncertainty. The uncertainty originates under the assumption that, for each PDE $p(u^i(t,x)|\hat{u}^i(t,x;\theta)) = \mathcal{N}(\hat{u}^i(t,x;\theta), \sigma_i^2)$ which fairly encapsulates both the noisy and noise-free cases. Then σ_i, the scaling factor, is derived to maximize the multi-task Gaussian likelihood.

5.2.4 PCGrad: Project conflicting gradients

Nevertheless, there are alternative approach for learning from multiple objectives. We also apply the backpropagation algorithm with PCGrad updates (Algorithm 1 in [17]) to calculate the modified gradients for the unweighted equation losses. Consequently, we update the model parameters with

$$\begin{gathered} \delta_{PC}\theta = \sum_{i=1}^{N_T} \mathbf{g}_{PC}^i \qquad [\mathit{Option\ II}] \\ \{\mathbf{g}_{PC}^i\} = PCGrad(\{\nabla_\theta \mathcal{L}_{eq}^i\}) \end{gathered} \tag{5.4}$$

where $\mathbf{g}_{PC}^i$ refers to the resulting modified gradients by the PCGrad update for the task i. Then, we use the $\delta_{PC}\theta$ to update the solver network parameters, θ. For a couple of tasks, PCGrad projects task i's gradient onto the normal vector

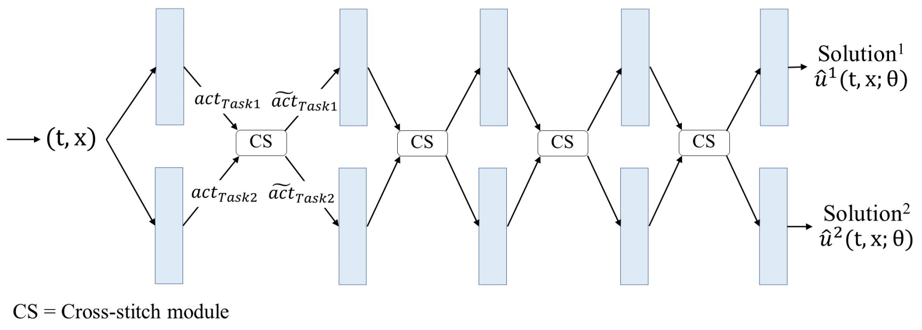

(t, x)
act_{Task1} $\widetilde{act}_{Task1}$
act_{Task2} $\widetilde{act}_{Task2}$
CS
Solution¹
$\hat{u}^1(t, x; \theta)$
Solution²
$\hat{u}^2(t, x; \theta)$
CS = Cross-stitch module

Figure 5.1: Overview of the cross-stitch network architecture for the adversarial multi-task training.

of task j's gradient, and vice versa, to deconflict the gradient directions during training.

5.2.5 Multi-task learning architecture

From the multi-task learning architecture point of view, we leverage two parallel baseline networks with cross-stitch modules to share the activations among all single-task networks. The overall architecture is depicted in Figure 5.1. Assume that we are considering two activation maps $(act_{Task1}, act_{Task2})$ at a particular layer, which belong to Task1 and Task2 respectively. We note that Task1 is the target PDE and Task2 is the generated auxiliary PDE, A trainable linear transformation of these activation maps is applied, before feeding into the successive layer for each PDE. The transformation can be formalized as

$$\begin{bmatrix} \tilde{act}_{Task1} \\ \tilde{act}_{Task2} \end{bmatrix} = \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix} \times \begin{bmatrix} act_{Task1} \\ act_{Task2} \end{bmatrix} \tag{5.5}$$

where every γ is gradient-based learnable to linearly control how much information to share from Task1 to Task2 and vice versa. With γ, cross-stitch modules adaptively retain the low-layer information, for example, γ_{11} determines how much act_{Task1} influence the higher-layer activations of Task1.

5.2.6 Adversarial multi-task training

Additionally including the high-loss samples lets the solver network focus more on the domain regions that are more challenging to regress for the solutions. The generator periodically (every F iterations, See **Algorithm 3**.) produces the additional high-loss samples by minimizing the loss function defined as follows

$$\begin{gathered} \mathcal{L}_{gen} = \frac{1}{N_f} \left\| scale(h(t_f, x_f; \theta_g), lb, ub) - (t_f, x_f) \right\|_2^2 - \sum_{j=1}^{N_T} \mathcal{L}_{eq}^j \\ scale(a, lb, ub) = (a - lb) \oslash (ub - lb) \\ \{(t^k, x^k)\} \overset{\text{U}}{\sim} \{scale(h(t_f^i, x_f^i; \theta_g), lb, ub)\}, |\{(t^k, x^k)\}| = pN_f \\ \{(t^i, x^i)\} := concat(\{(t^i, x^i)\}, \{(t^k, x^k)\}) \end{gathered} \tag{5.6}$$

Algorithm 3 Adversarial multi-task training

```
Require: Adversarial training frequency $F$, limit $L$ and iterations $E$, Sampling proportion $p$ and Generator parameters $\theta_g$
for $iter = 0$ to $epochs - 1$ do
  if $(iter\ mod\ F) = 0$ and $iter \leq \lfloor \frac{epochs}{L} \rfloor$ then
    for $e = 0$ to $E - 1$ do
      Freeze the solver network parameters $\theta$
      Generator's forward pass to obtain $scale(h(t_f, x_f; \theta_g), lb, ub)$
      Reconstruct $\forall i\ (t^i, x^i)$ as the solver's input
      Solver's forward pass to obtain $-\sum_{j=1}^{N_T} \mathcal{L}_{eq}^j$
      Calculate the adversarial loss $\mathcal{L}_{gen}$
      Backpropagate $\nabla_{\theta_g} \mathcal{L}_{gen}$ and update $\theta_g$
    end for
  else
    Freeze the generator parameters $\theta_g$
    Set the latest $\forall i\ (t^i, x^i)$ as the solver's input
    Solver's forward pass to obtain $\forall j\ \mathcal{L}_{eq}^j$
    Backpropagate using either $\nabla_\theta \mathcal{L}_{uncert}$ or $\delta_{PC}\theta$ and update $\theta$
  end if
end for
Return: $\theta^*$ and $\theta_g^*$
```

where $h(t_f, x_f; \theta_g)$ denotes the transformed samples from the N_f interior collocation points, (t_f, x_f). The averaged squared l_2-norm of $\mathcal{L}_{gen}$ helps the transformed samples to maintain the characteristics of the training distribution. The second term, $-\sum_{i=1}^{N_T} \mathcal{L}_{eq}^i$, is not averaged, and therefore dominate the overall loss magnitude for inducing the transformed samples to be from the regions that are difficult to regress. We scale the generator's outputs to the PDE domain (bounded from below and above by the vector lb and ub) and then uniformly pick a portion, p, of the scaled values to reconstruct $\{(t^i, x^i)\}$ as the new training set with total $(1+p)N_f + N_b$ samples. The $\oslash$ and $\overset{\text{U}}{\sim}$ notation refer to Hadamard division and uniformly sampling elements from a set. The pseudocode for our adversarial multi-task training is described in **Algorithm 3**.

5.3 Experiments on canonical PDEs

5.3.1 Burgers' equation

We consider the following Burgers' equation with Dirichlet boundary conditions. The equation is known to be notoriously difficult to solve using traditional numerical methods.

$$\begin{gathered} u_t + uu_x - \left(\frac{0.01}{\pi}\right) u_{xx} = 0,\ x \in [-1, 1],\ t \in [0, 1], \\ u(0, x) = -\sin(\pi x),\ u(t, -1) = u(t, 1) = 0. \end{gathered} \tag{5.7}$$

A Latin Hypercube Strategy (LHS) is chosen for sampling 10,000 interior data points (N_f) and 100 points of the the initial and boundary data (N_u).

Table 5.1: Burgers'equation: Performance comparison

Method	MAE	MSE	Rel. l_2 error
PINN	2.3×10^{-3}	2.1×10^{-4}	2.4×10^{-2}
	6.7×10^{-3}	2.1×10^{-3}	7.4×10^{-2}
ResNet	1.8×10^{-3}	5.0×10^{-5}	1.1×10^{-2}
	2.8×10^{-3}	1.7×10^{-4}	2.1×10^{-2}
PCGrad w/ CS*	2.3×10^{-3}	2.7×10^{-5}	8.4×10^{-3}
	7.1×10^{-3}	9.3×10^{-5}	5.7×10^{-2}
Uncert w/o CS*	6.4×10^{-3}	2.2×10^{-4}	2.5×10^{-2}
	3.0×10^{-3}	2.1×10^{-4}	2.3×10^{-2}
Uncert w/ CS*	1.1×10^{-3}	2.1×10^{-5}	7.4×10^{-3}
	1.8×10^{-3}	1.5×10^{-5}	6.3×10^{-3}
Uncert w/ CS* + Adv.**	$\mathbf{4.1 \times 10^{-4}}$	$\mathbf{1.0 \times 10^{-6}}$	$\mathbf{1.6 \times 10^{-3}}$
	$\mathbf{1.5 \times 10^{-3}}$	$\mathbf{1.0 \times 10^{-5}}$	$\mathbf{5.2 \times 10^{-3}}$

Note: CS* refers to the cross-stitch module and Adv.** refers to the adversarial multi-task training. These abbreviations are also used in the other tables. Indicated by the blue colour, $\mathcal{N}(0, 0.01)$ (Gaussian noise) is added to the referenced solutions on the initial and boundary condition, $u(t_b, x_b)$, for testing the method robustness. The best performance is on boldface.

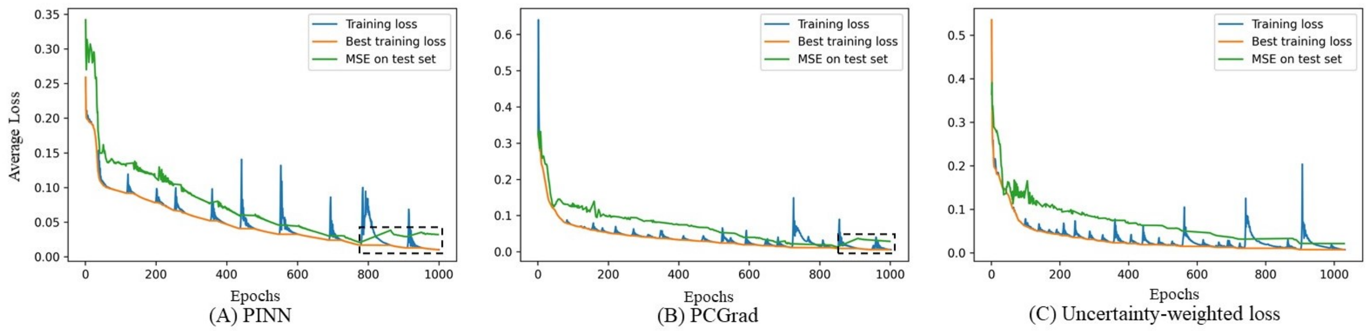


Figure 5.2: The average training loss ($\frac{1}{N_T}\sum_{i=1}^{N_T}\mathcal{L}_{eq}^i$) plotted with mean squared error (MSE) evaluated on Burgers' equation test set in the case of (A) original PINN (unweighted losses) (B) PCGrad and (C) Uncertainty-weighted loss. The dotted black blocks indicate the overfitting areas. In (B) and (C), where the multi-task learning is employed, the overfitting areas are either narrower (B) or unnoticeable (c). During training the network does not have an access to any validation or test sets.

Table 5.2: Poisson equation: Performance comparison

Method	MAE	MSE	Rel. l_2 error
PINN	7.9×10^{-4}	8.8×10^{-7}	2.9×10^{-2}
ResNet	4.7×10^{-4}	2.7×10^{-7}	1.7×10^{-2}
PCGrad w/ CS	1.1×10^{-4}	2.0×10^{-8}	4.8×10^{-3}
Uncert w/o CS	3.9×10^{-4}	2.1×10^{-7}	1.5×10^{-2}
Uncert w/ CS	1.2×10^{-4}	2.0×10^{-8}	4.7×10^{-3}
PCGrad w/ CS + Adv.	1.4×10^{-4}	2.6×10^{-8}	4.7×10^{-3}
Uncert w/ CS + Adv.	$\mathbf{9.5 \times 10^{-5}}$	$\mathbf{1.6 \times 10^{-8}}$	$\mathbf{4.1 \times 10^{-3}}$

The exact solutions for testing are made available by [36]. We set $\lambda = 0.01/\pi$ and consider a nearby coefficient of the form $\lambda^{aux} = \alpha\lambda$. Then, the Bayesian optimization, with $\mathcal{L}_{uncert}$ in Eq. (5.3) as the objective function, is applied for searching the best-tuned α^* under the $(0, 1)$ range, obtaining $\alpha^* \approx 0.6$. In the other experiments, we have found that random assignments with minimal tuning are sufficient for training the multi-task networks to outperform the existing approaches. Our neural networks are optimized by full-batch Adam [49] with 0.005 learning rate for 50,000 epochs. For the adversarial setting, we assign $F = 100$, $L = 2$, $E = 10$ and $p = 0.1$.

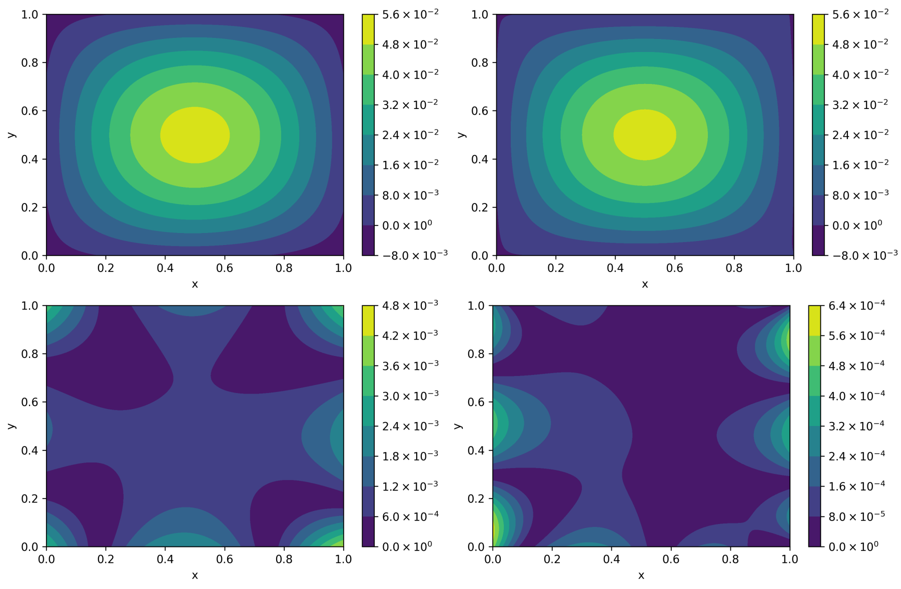
The performance comparison amongst variations of our method, PINN [12] and ResNet [50] is listed in Table 5.1. Both the multi-task modifications improve the performance by reducing the MAE, MSE and relative l_2 error. We select the best performing strategy, which in this PDE, is the uncertainty-weighting scheme, to undergo our adversarial training, and consequently, the MSE reaches the order of 10^{-6}. The adversarial training is found to successfully diminish the losses around the domain regions with high nonlinearity. With the aleatoric uncertainty quantification, Uncert carries out the similar performance against the noisy initial and boundary solutions, signifying our method robustness. Our ablation study of the cross-stitch units, comparing the fourth and fifth row in Table 5.1 on both the noise-free and noisy cases, shows that the modules are keys for enabling knowledge share between the neural networks, which reduces the overfitting as shown in Figure 5.2.

5.3.2 Poisson equation

The equation is commonly encountered in fluid dynamics. The considering equation is not temporally varying, but still, hard to solve analytically and therefore a numerical approach is usually required.

$$\begin{gathered}
u_{xx} + u_{yy} = f(x, y),\ x \in [0, 1],\ y \in [0, 1] \\
f(x, y) = -\sin(\pi x)\sin(\pi y) \\
u(x, 0) = 0,\ u(x, 1) = -\sin(\pi x)\sin(\pi) \\
u(0, y) = 0,\ u(1, y) = -\sin(\pi)\sin(\pi y)
\end{gathered} \tag{5.8}$$

We sample $N_f = 8,000$ and $N_b = 200$ using LHS for training. To build the test set with the shifted distribution, the input space of (x, y) is thoroughly discretized, having $\Delta x = \Delta y = 0.005$. To generate an auxiliary task, we scale



(A) PINN (B) Uncertainty-weighted loss

Figure 5.3: 2D visualization of the Poisson equation's estimated solutions by (A) PINN and (B) our uncertainty weighting based multi-task network enhanced with the adversarial training (the best variant of Table 5.2). At the second row, the color bar indicates the absolute prediction error from the ground truth.

up the $f(x, y)$, setting $f^{aux}(x, y) = -2\pi^2 \sin(\pi x) \sin(\pi y)$. We train our neural networks using full-batch Adam with 0.005 learning rate for 50,000 epochs. For the adversarial setting, we set $F = 100$, $L = 5$, $E = 10$ and $p = 0.1$.

The reported results in Table 5.2 shows that both the MTL modifications show the greater performance compared to PINN and ResNet, still, there is no significant difference between the strategies; thus the adversarial training is conducted on both strategies. The similar results are found in the PCGrad case whilst the performance boosts are seen in the uncertainty weighting case, contributing to the highest prediction accuracy as shown in Figure 5.3. We also found that the cross-stitch units are recommended to make the most out of the designed MTL losses.

The reported results in Table 5.2 shows that both the MTL modifications show the greater performance compared to PINN and ResNet, still, there is no significant difference between the strategies; thus the adversarial training is conducted on both strategies. The similar results are found in the PCGrad case whilst the performance boosts are seen in the uncertainty weighting case, contributing to the highest prediction accuracy as shown in Figure 5.3. We also found that the cross-stitch units are recommended to make the most out of the designed MTL losses.

Table 5.3: Fokker-Planck equation: Performance comparison

Method	MAE	MSE	Rel. l_2 error
PINN	2.2×10^{-3}	8.4×10^{-6}	9.9×10^{-3}
ResNet	3.9×10^{-3}	2.6×10^{-5}	1.7×10^{-2}
PCGrad w/ CS	1.7×10^{-3}	4.9×10^{-6}	7.6×10^{-3}
Uncert w/o CS	1.8×10^{-3}	5.2×10^{-6}	7.8×10^{-3}
Uncert w/ CS	9.0×10^{-4}	1.4×10^{-6}	4.0×10^{-3}
Uncert w/ CS + Adv.	$\mathbf{3.2 \times 10^{-4}}$	$\mathbf{1.9 \times 10^{-7}}$	$\mathbf{1.5 \times 10^{-3}}$

5.3.3 1D Fokker-Planck equation

The 1D PDE describes a snapshot of the probability density functions evolution of stochastic systems.

$$\begin{gathered} -[(ax - bx^3)u(x)]_x + \frac{\sigma^2}{2}u(x)_{xx} = 0,\ \Delta x \sum_{i=1}^{N_f} u(x^i) = 1 \\ u(-2.2) = u(2.2) = 0,\ (a, b, \sigma, \Delta x) = (0.3, 0.5, 0.5, 0.01) \end{gathered} \tag{5.9}$$

We can get the training set and the boundary set with the step length, $\Delta x = 0.01$, and evaluate the solver network performance on the more fine-grained points constructed with the smaller step length, $\Delta x = 0.005$. We consider a to be the PDE parameter λ and set $\lambda^{aux} = 0.5$ (a close value) to simply generate an auxiliary equation. Our neural networks are optimized using full-batch Adam with 0.01 learning rate for 30,000 epochs. For the adversarial setting, we set $F = 100$, $L = 3$, $E = 10$ and $p = 0.1$.

Based on the performance comparison, which is provided in Table 5.3, we inspect that the neural network, trained with PCGrad, does not outperform PINN. This might be because, in practice, we could fortuitously break the assumptions of PCGrad [17], which essentially need to be held for the loss reduction guarantee. We choose the uncertainty-weighting scheme to further experiment with our adversarial training and, in consequence, the performance is enhanced in terms of the reduced MSE from 10^{-6} to 10^{-7}.

5.4 Summary

We propose a novel approach that applies multi-task learning to the physics-informed learning frameworks to produce better-generalized solutions. Compared to previous work, our method significantly reduces test errors across various PDE examples, spanning both low-dimensional and high-dimensional settings. The combined use of multi-task learning and adversarial training enhances the network’ s performance by enabling joint representation learning across multiple target PDE instances and incorporating challenging samples. This approach allows the network to focus more effectively on high-loss regions, which are typically harder to learn.

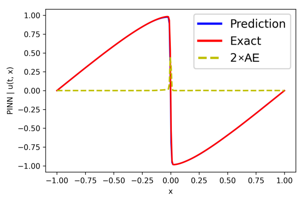
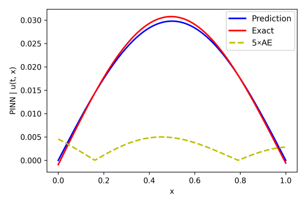
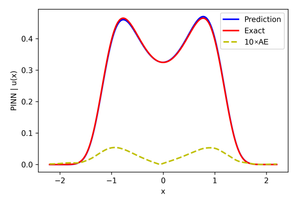
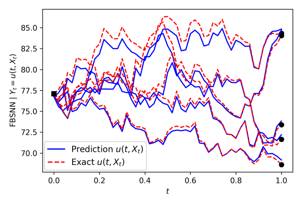
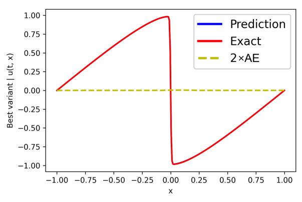
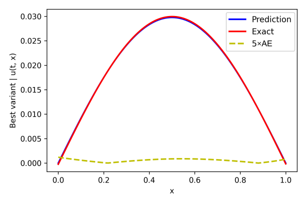
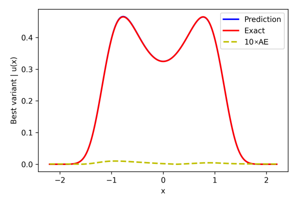
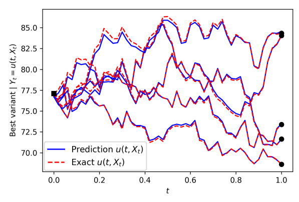
	Burgers (t=0.5)	**Poisson (y=0.8)**	**Fokker-Planck**	**100-dimensional Black-Scholes-Barenblatt**
Previous work				
Best variant (Ours)				

Figure 5.4: Comparison of the predicted and exact solutions. The predictions at the first row are reproduced from the previous works (PINN or FBSNN [51]). The predictions at the second row are from the best variant of our proposed method for each PDE. The absolute errors (AEs or dotted yellow lines) around the high-loss or dynamically changing regions are lower when our method is employed.

Chapter 6

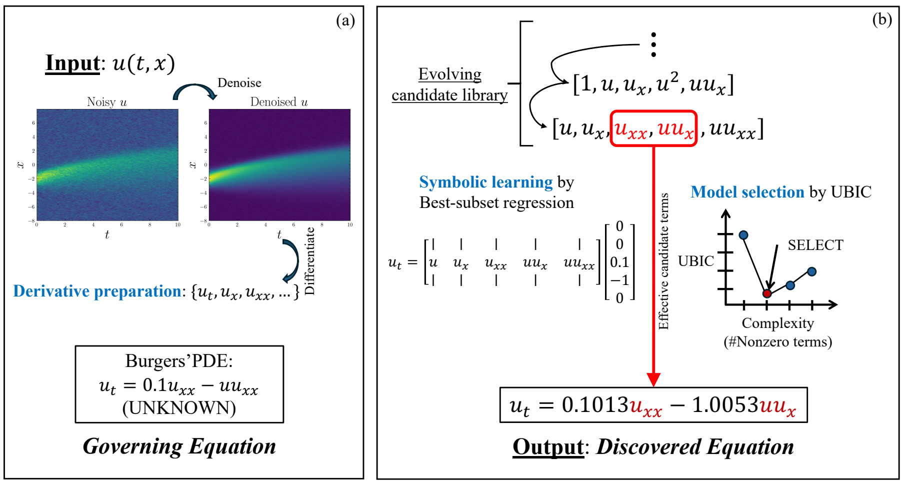
Conclusion and future work

6.1 Summary and limitation

In this thesis, we introduced an interpretable and robust framework called nPIML for nonlinear PDE discovery. The nPIML framework effectively addresses challenges in derivative computation, sensitivity to regularization, and noisy data through a combination of the weakly physics-informed solver network, preselector mechanisms, and sparse regression techniques. By leveraging dPINNs and projection networks, the framework fine-tunes PDE coefficients and enhances denoising, demonstrating both accuracy and robustness even in noisy and data-limited scenarios. However, the proposed framework is not without limitations. For example, it lacks an explicit denoising mechanism during the early stages of derivative preparation and sparse regression. As a result, noise with an unknown distribution could adversely affect these initial processes, potentially causing the entire framework to fail.

Complementing the automatic model selection component of the nPIML framework, the UBIC method extends the BIC by incorporating model uncertainty, thereby enhancing the reliability and efficiency of model selection, even in noisy data environments. The analytical posterior not only expedites the identification of the true governing equations but also mitigates the overfitting risks associated with traditional BIC. Through comparative validation, UBIC demonstrates consistent and effective model selection across various canonical PDEs. There is still room for improving the UBIC. For instance, exploring innovative applications of UBIC to guide users toward the true governing equation, even when an overcomplete set of candidate terms is not initially available, presents a interesting direction. One possible approach could involve using UBIC as the fitness function in an evolutionary-based PDE-discovery framework. Additionally, it would be highly beneficial for the research community to gain deeper insights into which quantities are most effective for identifying governing equations.

We employ PINNs as a tool for performing simulation-based model selection in Chapter 4. Since PINNs offer a novel and flexible approach to solving forward and inverse problems regarding PDEs, especially when the data is irregularly sampled, improving their generalization capability remains a valuable and ongoing research. In this thesis, we demonstrated that generalization can be enhanced through adversarial multi-task learning. However, a notable limi-



(a)
Input: $u(t,x)$
Denoise
Noisy u
Denoised u
Differentiate
Derivative preparation: $\{u_t, u_x, u_{xx}, \dots\}$
Burgers' PDE: $u_t = 0.1u_{xx} - uu_{xx}$ (UNKNOWN)
Governing Equation
(b)
Evolving candidate library
$[1, u, u_x, u^2, uu_x]$
$[u, u_x, u_{xx}, uu_x, uu_{xx}]$
Symbolic learning by Best-subset regression
Effective candidate terms
Model selection by UBIC
UBIC
SELECT
Complexity (#Nonzero terms)
$u_t = 0.1013u_{xx} - 1.0053uu_x$
Output: Discovered Equation

Figure 6.1: The conceptual framework of an evolutionary-based PDE discovery method, as future work, aims to overcome the potential violation of the overcompleteness assumption. (a) Data preprocessing. (b) Symbolic learning and model selection.

tation of PINNs, in general, is their reliance on prior knowledge of the governing PDE, which must be explicitly specified before training the network.

Overall, this thesis marks a significant advancement in tackling forward and inverse problems related to differential equations, establishing a foundation for more interpretable and effective frameworks in the modeling and data-driven discovery of complex physical systems.

6.2 Future work

I plan to develop a compositional machine learning framework for the sparse identification of governing equations. The primary objectives of this framework are to achieve noise tolerance, computational efficiency, and, most critically, the accurate recovery of governing equations. My approach will involve integrating UBIC with an evolutionary-based or reinforcement-learning-based PDE discovery framework to ensure that the identified PDEs are both parsimonious and can be effectively optimized, even in cases where an overcomplete set of candidate terms is not initially available. To address the current limitation that the overcompleteness assumption may be violated, I plan to implement an evolving candidate library inspired by genetic algorithms. As illustrated in Figure 6.1, this conceptual approach involves iteratively refining the library' s terms: older terms with minimal contribution will be removed, while new terms with varying mathematical forms (e.g., derivative orders and polynomial degrees) will be introduced to ensure distinctive functional behaviors. Throughout the learning iterations, we will track the top-performing candidate terms using a reliable information criterion, such as the proposed UBIC.

I am also interested in creating a noise-tolerant differentiation method that requires minimal or, ideally, no hyperparameter tuning. Such a method would enhance the robustness of discovery approaches, reducing sensitivity to model configurations and ensuring higher-quality results across diverse scenarios.

Ultimately, my goal is to identify the most effective information criterion for the practical identification of governing physics. To this end, I aim to determine which quantities or features are most impactful in tackling this challenging task.

Bibliography

[1] Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2, Jan 2024. ISSN 1573-7462. doi: 10.1007/s10462-023-10622-0. URL `https://doi.org/10.1007/s10462-023-10622-0`.

[2] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K. Jha. Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare. *IEEE Journal of Biomedical and Health Informatics*, 19(6):1893–1905, 2015. doi: 10.1109/JBHI.2014.2344095.

[3] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631. URL `https://www.science.org/doi/abs/10.1126/sciadv.aay2631`.

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1517384113`.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[6] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[7] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017. doi: 10.1126/sciadv.1602614. URL `https://www.science.org/doi/abs/10.1126/sciadv.1602614`.

[8] Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970. doi: 10.1080/00401706.1970.10488634. URL `https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634`.

[9] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. doi: 10.1109/TAC.1974.1100705.

[10] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[11] Pongpisit Thanasutives, Takashi Morita, Masayuki Numao, and Ken ichi Fukui. Noise-aware physics-informed machine learning for robust PDE discovery. *Machine Learning: Science and Technology*, 4(1):015009, feb 2023. doi: 10.1088/2632-2153/acb1f0. URL `https://dx.doi.org/10.1088/2632-2153/acb1f0`.

[12] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

[13] Pongpisit Thanasutives, Takashi Morita, Masayuki Numao, and Ken-ichi Fukui. Adaptive Uncertainty-Penalized Model Selection for Data-Driven PDE Discovery. *IEEE Access*, 12:13165–13182, 2024. doi: 10.1109/ACCESS.2024.3354819.

[14] Pongpisit Thanasutives and Ken ichi Fukui. On uncertainty-penalized Bayesian information criterion, 2024. URL `https://arxiv.org/abs/2404.16881`.

[15] Pongpisit Thanasutives and Ken-ichi Fukui. Uncertainty-Penalized Bayesian Information Criterion for Parametric Partial Differential Equation Discovery. In *Machine Learning and the Physical Sciences Workshop @ NeurIPS 2024*, 2024. URL `https://ml4physicalsciences.github.io/2024/files/NeurIPS_ML4PS_2024_5.pdf`.

[16] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[17] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html`.

[18] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[19] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3), Jun 2011. ISSN 0004-5411. doi: 10.1145/1970392.1970395. URL `https://doi.org/10.1145/1970392.1970395`.

[20] Hao Xu, Haibin Chang, and Dongxiao Zhang. DLGA-PDE: Discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm. *Journal of Computational Physics*, 418: 109584, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109584. URL `https://www.sciencedirect.com/science/article/pii/S0021999120303582`.

[21] Luning Sun, Daniel Huang, Hao Sun, and Jian-Xun Wang. Bayesian Spline Learning for Equation Discovery of Nonlinear Dynamics with Quantified Uncertainty. *Advances in Neural Information Processing Systems*, 35:6927–6940, 2022.

[22] Jun Li, Gan Sun, Guoshuai Zhao, and H Lehman Li-wei. Robust low-rank discovery of data-driven partial differential equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 767–774, 2020.

[23] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press, 2022.

[24] Patrick A. K. Reinbold, Daniel R. Gurevich, and Roman O. Grigoriev. Using noisy or incomplete data to discover models of spatiotemporal dynamics. *Phys. Rev. E*, 101:010203, Jan 2020. doi: 10.1103/PhysRevE.101.010203. URL `https://link.aps.org/doi/10.1103/PhysRevE.101.010203`.

[25] N. M. Mangan, J. N. Kutz, S. L. Brunton, and J. L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204):20170009, 2017. doi: 10.1098/rspa.2017.0009. URL `https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2017.0009`.

[26] Xin Dong, Yu-Long Bai, Yani Lu, and Manhong Fan. An improved sparse identification of nonlinear dynamics with Akaike information criterion and group sparsity. *Nonlinear Dynamics*, 111(2):1485–1510, 2023. doi: 10.1007/s11071-022-07875-9. URL `https://doi.org/10.1007/s11071-022-07875-9`.

[27] Maziar Raissi, Paris Perdikaris, Nazanin Ahmadi, and George Em Karniadakis. Physics-informed neural networks and extensions. *arXiv preprint arXiv:2408.16806*, 2024.

[28] Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Computer Methods in Applied Mechanics and Engineering*, 370: 113250, 2020.

[29] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3 (3):218–229, Mar 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL `https://doi.org/10.1038/s42256-021-00302-5`.

[30] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021. doi: 10.1126/sciadv.abi8605. URL `https://www.science.org/doi/abs/10.1126/sciadv.abi8605`.

[31] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[32] Peter Ranacher, Richard Brunauer, Wolfgang Trutschnig, Stefan Van der Spek, and Siegfried Reich. Why GPS makes distances bigger than they are. *International Journal of Geographical Information Science*, 30(2):316–333, 2016.

[33] David A Faux and Janet Godolphin. Manual timing in physics experiments: error and uncertainty. *American Journal of Physics*, 87(2):110–115, 2019.

[34] Kadierdan Kaheman, Steven L Brunton, and J Nathan Kutz. Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data. *Machine Learning: Science and Technology*, 3(1):015031, 2022.

[35] G Hosein Mohimani, Massoud Babaie-Zadeh, and Christian Jutten. Fast sparse representation based on smoothed L0 norm. In *International Conference on Independent Component Analysis and Signal Separation*, pages 389–396. Springer, 2007.

[36] Cea Basdevant, M Deville, P Haldenwang, JM Lacroix, J Ouazzani, R Peyret, Paolo Orlandi, and AT Patera. Spectral and finite difference solutions of the Burgers equation. *Computers & fluids*, 14(1):23–41, 1986.

[37] Diederik Johannes Korteweg and Gustav De Vries. On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 39(240):422–443, 1895.

[38] Daniel A Messenger and David M Bortz. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021.

[39] D. Bertsimas and R. Weismantel. *Optimization Over Integers*. Dynamic Ideas, 2005.

[40] Sheng Chen, Stephen A Billings, and Wan Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control*, 50(5):1873–1896, 1989.

[41] Stephen A Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[42] Hussein Hazimeh, Rahul Mazumder, and Ali Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Mathematical Programming*, pages 1–42, 2021.

[43] Dimitris Bertsimas and Wes Gurnee. Learning sparse nonlinear dynamics via mixed-integer optimization. *Nonlinear Dynamics*, 111(7):6585–6604, 2023.

[44] Hamparsum Bozdogan and Dominique M.A. Haughton. Informational complexity criteria for regression models. *Computational Statistics & Data Analysis*, 28(1):51–76, 1998. ISSN 0167-9473. doi: https://doi.org/10.1016/S0167-9473(98)00025-5. URL https://www.sciencedirect.com/science/article/pii/S0167947398000255.

[45] David JC MacKay et al. Bayesian nonlinear modeling for the prediction competition. *ASHRAE transactions*, 100(2):1053–1062, 1994.

[46] Miles Cranmer. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl, 2023.

[47] Samuel Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz. Data-Driven Identification of Parametric Partial Differential Equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, 2019. doi: 10.1137/18M1191944. URL https://doi.org/10.1137/18M1191944.

[48] Pongpisit Thanasutives, Masayuki Numao, and Kenichi Fukui. Adversarial Multi-task Learning Enhanced Physics-informed Neural Networks for Solving Partial Differential Equations. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2021.

[49] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: international conference on learning representations*, pages 1–15, 2015.

[50] Batuhan Güler, Alexis Laignelet, and Panos Parpas. Towards Robust and Stable Deep Learning Algorithms for Forward Backward Stochastic Differential Equations. *arXiv preprint arXiv:1910.11623*, 2019.

[51] Maziar Raissi. *Forward–Backward Stochastic Neural Networks: Deep Learning of High-Dimensional Partial Differential Equations*, chapter Chapter 18, pages 637–655. doi: 10.1142/9789811280306_0018. URL https://www.worldscientific.com/doi/abs/10.1142/9789811280306_0018.