Data-Driven Discovery of PDEs via the Adjoint Method

Mohsen Sadr* *msadr@mit.edu*
Massachusetts Institute of Technology, USA
Paul Scherrer Institute, Switzerland

Tony Tohme* *tohme@mit.edu*
Massachusetts Institute of Technology, USA

Kamal Youcef-Toumi *youcef@mit.edu*
Massachusetts Institute of Technology, USA

Reviewed on OpenReview: *https://openreview.net/forum?id=Az3mJ4d1eT*

Abstract

In this work, we present an adjoint-based method for discovering the underlying governing partial differential equations (PDEs) given data. The idea is to consider a parameterized PDE in a general form and formulate a PDE-constrained optimization problem aimed at minimizing the error of the PDE solution from data. Using variational calculus, we obtain an evolution equation for the Lagrange multipliers (adjoint equations), allowing us to compute the gradient of the objective function with respect to the parameters of PDEs given data in a straightforward manner. In particular, we consider a family of temporal parameterized PDEs that encompass linear, nonlinear, and spatial derivative candidate terms, and elegantly derive the corresponding adjoint equations. We show the efficacy of the proposed approach in identifying the form of the PDE up to machine accuracy, enabling the accurate discovery of PDEs from data. We also compare its performance with the famous PDE Functional Identification of Nonlinear Dynamics method known as PDE-FIND Rudy et al. (2017) among others, on both smooth and noisy data sets. Even though the proposed adjoint method relies on forward/backward solvers, it outperforms PDE-FIND in the limit of large data sets thanks to the analytic expressions for gradients of the cost function with respect to each PDE parameter. Our implementation is publicly available on GitHub.[1]

1 Introduction

A significant body of literature on data-driven modeling of physical processes focuses on using neural networks to achieve rapid predictions from training datasets. The data-driven estimation methods include Physics-Informed Neural Networks Raissi et al. (2019), Pseudo-Hamiltonian neural networks Eidnes and Lye (2024), structure preserving Matsubara et al. (2020); Sawant et al. (2023), and reduced order modelling Duan and Hesthaven (2024). These methods often provide efficient and somewhat "accurate" predictions when tested as an interpolation method in the space of input or boundary parameters. Such fast estimators are beneficial when many predictions of a dynamic system is needed, for example in the shape optimization task in fluid dynamics.

However, the data-driven estimators often fail to provide accurate solution to the dynamical system when tested outside the training space, i.e. for extrapolation. Furthermore, given the regression-based nature of these predictors, often they do not offer any error estimator in prediction. Since a wide range of numerical methods already exist for solving classical governing equations, a compelling alternative is to

*Equal contribution.
[1] `https://github.com/mohsensadr/DiscoverPDEAdjoint.git`

infer the underlying equations directly from data. Once the governing equation is found, one can either use the standard numerical methods for prediction, or train a PINN-like surrogate model for fast evaluation. This way we guarantee the consistency with observed data, estimator for the numerical approximation, and interoperability. Hence, learning the underlying physics given data has motivated a new branch in the scientific machine learning for discovering the mathematical expression as the governing equation given data.

The wide literature of data-driven discovery of dynamical systems includes equation-free modelling Kevrekidis et al. (2003), artificial neural networks González-García et al. (1998), nonlinear regression Voss et al. (1999), empirical dynamic modeling Sugihara et al. (2012); Ye et al. (2015), modeling emergent behavior Roberts (2014), automated inference of dynamics Schmidt et al. (2011); Daniels and Nemenman (2015a;b), normal form identification in climate Majda et al. (2009), nonlinear Laplacian spectral analysis Giannakis and Majda (2012), modeling plasma physics Alves and Fiuza (2022), and Koopman analysis Mezić (2013) among others. There has been a significant advancement in this field by combining symbolic regression with the evolutionary algorithms Bongard and Lipson (2007); Schmidt and Lipson (2009); Tohme et al. (2022), which enable the direct extraction of nonlinear dynamical system information from data. Furthermore, the concept of sparsity Tibshirani (1996) has recently been employed to efficiently and robustly deduce the underlying principles of dynamical systems Brunton et al. (2016); Mangan et al. (2016).

Related work. Next, we review several relevant works that have shaped the current landscape of discovering PDEs from data:

PDE-FIND Rudy et al. (2017). This method has been developed to discover underlying partial differential equation by minimizing the L_2^2-norm point-wise error of the parameterized forward model from the data using sparse regression. Estimating all the possible derivatives using Finite Difference, PDE-FIND constructs a dictionary of possible terms and finds the underlying PDE by performing a sparse search using ridge regression problem with hard thresholding, also known as STRidge optimization method. Several further developments in the literature has been carried out based on this idea Champion et al. (2019); Kaheman et al. (2020) including Weak-SINDy Messenger and Bortz (2021). In these methods, as the size (or dimension) of the data set increases, the PDE discovery optimization problem based on point-wise error becomes extremely expensive, forcing the user to arbitrarily reduce the size of data by resampling, or compressing the data using proper orthogonal decomposition. Needles to say, in case of non-linear dynamics, such truncation of data can introduce bias in prediction leading to finding a wrong PDE.

PDE-Net, PINN-SR, and PDE-LEARN. One of the issues with the PDE-FIND is the use of Finite Difference in estimating the derivatives. This has motivated the idea of combining the PDE discovery task with the PDE estimator that avoids the use of Finite Difference. In methods such as PDE-Net Long et al. (2018; 2019), PDE-LEARN Stephany and Earls (2024) or PINN-SR Chen et al. (2021), the search for weights/biases of a complicated neural network as a differentiable data-driven PDE estimator is combined with the sparse search in the space of possible terms to find the coefficients of the underlying PDE. Combining PDE discovery with data-driven estimation of PDEs makes these methods more expensive than PDE-FIND in practice.

Hidden Physics Models Raissi and Karniadakis (2018). This method assumes that the relevant terms of the governing PDE are already identified and finds its unknown parameters using Gaussian process regression (GPR). While GPR is an accurate interpolator which offers an estimate for the uncertainty in prediction, its training scales poorly with the size of the training data set as it requires inversion of the covariance matrix.

Contributions. In this paper, we introduce a novel approach for discovering PDEs from data based on the well-known adjoint method, i.e. PDE-constrained optimization method. The idea is to formulate the objective (or cost) functional such that the estimate function f minimizes the L_2^2-norm error from the data points f^* with the constraint that f is the solution to a parameterized PDE using the method of Lagrange multipliers. Here, we consider a parameterized PDE in a general form and the task is to find all the parameters including irrelevant ones. By finding the variational extremum of the cost functional with respect to the function f, we obtain a backward-in-time evolution equation for the Lagrange multipliers (adjoint equations). Next, we solve the forward parameterized PDE as well as the adjoint equations numerically. Having found estimates of the Lagrange multipliers and solution to the forward model f, we can numerically compute the gradient

of the objective function with respect to the parameters of PDEs given data in a straightforward manner. In particular, for a family of parameterized and nonlinear PDEs, we show how the corresponding adjoint equations can be elegantly derived. We note that the adjoint method has been successfully used before as an efficient method for uncertainty quantification Flath et al. (2011), shape optimization and sensitivity analysis method in fluid mechanics Hughes et al. (1998); Jameson (2003); Caflisch et al. (2021) and plasma physics Antonsen et al. (2019); Geraldini et al. (2021). Unlike the usual use of PDE-constrained adjoint optimization where the governing equation is known, in this paper we are interested in finding the form along with the coefficients of the PDE given data.

The remainder of the paper is organized as follows. First in Section 2, we introduce and derive the proposed adjoint-based method of finding the underlying system of PDEs given data. Next in Section 3, we present our results on a wide variety of PDEs and compare the solution with the celebrated PDE-FIND in terms of error and computational/training time. In Section 4, we discuss the limitations for the current version of our approach and provide concluding remarks in Section 5.

2 Adjoint method for finding PDEs

In this section, we introduce the problem and derive the proposed adjoint method for finding governing equations given data.

Problem setup. Assume we are given a data set on a spatial/temporal grid $\mathcal{G} = \bigcup_{j=0}^{N_t} \mathcal{G}^{(j)}$ with $\mathcal{G}^{(j)} = \{(\boldsymbol{x}^{(k)}, t^{(j)}) \mid k = 1, ..., N_{\boldsymbol{x}}\}$ for the vector of functions $\boldsymbol{f}^*$ where k is the spatial index and j the time index with $t^{(N_t)} = T$ being the final time. Here, $\boldsymbol{x}^{(k)} \in \Omega \subset \mathbb{R}^n$ is spatial position inside the solution domain Ω, $t^{(j)}$ denotes the j-th time that data is available, and output is a discrete map $\boldsymbol{f}^* : \mathcal{G} \to \mathbb{R}^N$. The goal is to find the governing equations that accurately estimates $\boldsymbol{f}^*$ at all points on $\mathcal{G}$. In order to achieve this goal, we formulate the problem using the method of Lagrange multipliers.

Adjoint method. For simplicity, let us first consider only the time interval $t \in [t^{(j)}, t^{(j+1)}]$. Consider a general a forward model $\mathcal{L}[\cdot]$ that evolves an N-dimensional vector of sufficiently differentiable functions $\boldsymbol{f}(\boldsymbol{x}, t = t^{(j)})$ in $t \in (t^{(j)}, t^{(j+1)}]$ and $\boldsymbol{x} \in \Omega$ where the i-th PDE is given by

$$\mathcal{L}_i[\boldsymbol{f}] := \partial_t f_i + \sum_{\boldsymbol{d},\boldsymbol{p}} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[f^{\boldsymbol{p}}] = 0 \tag{1}$$

for $i = 1, \ldots, N$, resulting in a system of N-PDEs, i.e. the i-th PDE $\mathcal{L}_i$ predicts f_i. Here, $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ is an n-dimensional (spatial) input vector, and $\boldsymbol{f} = [f_1, f_2, \ldots, f_N]$ is an N-dimensional vector of functions. We use the shorthand $f_i = f_i(\boldsymbol{x}, t)$ and $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}, t)$. Furthermore, $\boldsymbol{p} = [p_1, \ldots, p_N]$ and $\boldsymbol{d} = [d_1, d_2, \ldots, d_n]$ are non-negative index vectors such that $f^{\boldsymbol{p}} = f_1^{p_1} f_2^{p_2} \cdots f_N^{p_N}$ where p_i for $i = 1, ..., N$ denotes the power of f_i and

$$\nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[f^{\boldsymbol{p}}] := \nabla_{x_1}^{(d_1)} \nabla_{x_2}^{(d_2)} \cdots \nabla_{x_n}^{(d_n)} [f_1^{p_1} f_2^{p_2} ... f_N^{p_N}] \ , \tag{2}$$

is an iterated differential operator acting on $f^{\boldsymbol{p}}$ where $\nabla_{x_j}^{(d_j)}$ for $j = 1, ..., n$ indicates d_j-th derivative in x_j dimension, and $\partial_t f_i$ denotes the time derivative of the i-th function. We denote the vector of unknown parameters by $\boldsymbol{\alpha} = [\alpha_{i,\boldsymbol{d},\boldsymbol{p}}]_{(i,\boldsymbol{d},\boldsymbol{p}) \in \mathcal{D}}$, where $\mathcal{D}$ represents the domain of all valid combinations of i, $\boldsymbol{d}$, and $\boldsymbol{p}$.

Having written the forward model 1 as general as possible, the goal is to find the parameters $\boldsymbol{\alpha}$ such that $\boldsymbol{f}$ approximates the data points of $\boldsymbol{f}^*$ at $t = t^{(j+1)}$ given the solution $\boldsymbol{f} = \boldsymbol{f}^*$ at $t = t^{(j)}$. To this end, we formulate a semi-discrete objective (or cost) functional that minimizes the L_2^2-norm error between what the model predicts and the data $\boldsymbol{f}^*$ on $\mathcal{G}^{(j+1)}$, with the constraint that $\boldsymbol{f}$ solves the forward model in Eq. (1), i.e.

$$\mathcal{C}[\boldsymbol{f}] = \sum_{i=1}^{N} \left(\sum_k \left(f_i^*\big(\boldsymbol{x}^{(k)}, t^{(j+1)}\big) - f_i\big(\boldsymbol{x}^{(k)}, t^{(j+1)}\big) \right)^2 + \int \lambda_i(\boldsymbol{x}, t) \mathcal{L}_i[\boldsymbol{f}(\boldsymbol{x}, t)] d\boldsymbol{x} dt \right) + \epsilon_0 ||\boldsymbol{\alpha}||_2^2 \ , \tag{3}$$

where $||.||_2$ denotes L_2-norm, and ϵ_0 is the regularization factor. We note that PDE discovery task is ill-posed since the underlying PDE is not unique and the regularization term helps us find the PDE with the least possible coefficients.

Clearly, given estimates of $\boldsymbol{f}$ and Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)$, the gradient of the cost function with respect to model parameters can be simply computed via

$$\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} = (-1)^{|\boldsymbol{d}|} \int f^{\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] d\boldsymbol{x}dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \tag{4}$$

where $i = 1, ..., N$ and $|\boldsymbol{d}| = d_1 + ... + d_n$, where $|.|$ denotes L_1-norm. Here, we used integration by parts and imposed the condition that $\boldsymbol{\lambda} \to \boldsymbol{0}$ on the boundaries of Ω at all time $t \in [0, T]$. The compact support of λ is motivated by the fact that we consider boundary conditions as known. In case boundary conditions are parameterized, we need to add another constraint to find its parameters.

The analytical expression 4 can be used for finding the parameters of PDE using in the gradient descent method with update rule

$$\alpha_{i,\boldsymbol{d},\boldsymbol{p}} \leftarrow \alpha_{i,\boldsymbol{d},\boldsymbol{p}} - \eta \frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} \tag{5}$$

for $i = 1, \ldots, N$, where $\eta = \beta \min(\Delta \boldsymbol{x})^{|\boldsymbol{d}| - d_{\max}}$ is the learning rate which includes a free parameter β and scaling coefficient for each term of the PDE, and $d_{\max} = \max(|\boldsymbol{d}|)$ for all considered $\boldsymbol{d}$. Let us also define $p_{\max} = \max(|\boldsymbol{p}|)$ as the highest order in the forward PDE model. We note that since the terms of the PDEs may have different scaling, the step size for the corresponding coefficient must be adjusted accordingly. Based on our simulation studies, the gradient of the cost function is most sensitive to the highest order terms of the PDE. In Appendix D, we give a justification for our choice of the learning rate η.

However, before we can use Eq. (4) and (5), we need to find $\boldsymbol{\lambda}$, hence the adjoint equation. This can be achieved by finding the functional extremum of the cost functional $\mathcal{C}$ with respect to $\boldsymbol{f}$. First, we note that the semi-discrete total variation of $\mathcal{C}$ can be derived as

$$\begin{aligned}\delta \mathcal{C} = \sum_{i=1}^{N} \Bigg(& \sum_k \lambda_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) \delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}} - \sum_k 2(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})) \delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}} \\ & + \int (-\frac{\partial \lambda_i}{\partial t} + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i]) \delta f_i d\boldsymbol{x}dt \Bigg)\end{aligned}$$

where δf_i denotes variation with respect to $f_i(\boldsymbol{x}, t)$ in $t \in (t^{(j)}, t^{(j+1)})$, and $\delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}}$ variation with respect to $f_i(\boldsymbol{x} = \boldsymbol{x}^{(k)}, t = t^{(j+1)})$. In this derivation, we discretized the last integral resulting from integration by parts in time using the same mesh as the one of data $\mathcal{G}^{(j+1)}$. Here again, we used integration by parts and imposed the condition that $\boldsymbol{\lambda} \to \boldsymbol{0}$ on the boundaries of Ω at all time $t \in [t^{(j)}, t^{(j+1)}]$ for $i = 1, ..., N$. Note that $f_i(\boldsymbol{x}, t)$ is the output of i-th PDE.

Next, we find the optimums of $\mathcal{C}$ (and hence the adjoint equations) by taking the variational derivatives with respect to f_i and $f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}}$, i.e.

$$\frac{\delta \mathcal{C}}{\delta f_i} = 0 \implies \frac{\partial \lambda_i}{\partial t} = \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] \tag{6}$$

and

$$\frac{\delta \mathcal{C}}{\delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}}} = 0 \implies \lambda_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) = 2(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})) \tag{7}$$

for $i = 1, ..., N$ and $j = 0, ..., N_t - 1$. We note that the adjoint equation 6 for the system of PDEs is backward in time with the final condition at the time $t = t^{(j+1)}$ given by Eq. (7). In Appendices A and B, we provide a detailed derivation of adjoint equation and its gradient. In order to make the notation clear, we also present examples for adjoint equations in Appendix G. The adjoint equation is in the continuous form, while the final condition is on the discrete points, i.e. on the grid $\mathcal{G}^{(j+1)}$. In order to obtain the Lagrange multipliers in $t \in [t^{(j+1)}, t^{(j)})$, a numerical method appropriate for the forward 1 and adjoint equation 6 should be deployed. Furthermore, the adjoint equation should have the same or coarser spatial discretization as $\mathcal{G}^{(j+1)}$ to enforce the final condition 7.

Training with smooth data set. The training procedure follows the standard gradient descent method. We start by taking an initial guess for parameters $\boldsymbol{\alpha}$, e.g. here we take $\boldsymbol{\alpha} = 0$ initially. For each time interval $t \in [t^{(j)}, t^{(j+1)}]$, first we solve the forward model 1 numerically to estimate $\boldsymbol{f}(\boldsymbol{x}^{(k)}, t^{(j+1)})$ given the initial condition

$$\boldsymbol{f}(\boldsymbol{x}^{(k)}, t^{(j)}) = \boldsymbol{f}^*(\boldsymbol{x}^{(k)}, t^{(j)}) \ . \tag{8}$$

Then, the adjoint Eq. 6 is solved backwards in time with the final time condition 7. Finally, the estimate for parameters of the model is updated using Eq. 5. We repeat this for all time intervals $j = 0, ..., N_t - 1$ until convergence.

In order to improve the search for coefficients and enforce the PDE identification, we also deploy thresholding Blumensath and Davies (2009), i.e. set $\alpha_{i,\boldsymbol{d},\boldsymbol{p}} = 0$ if $|\alpha_{i,\boldsymbol{d},\boldsymbol{p}}| < \sigma$ where σ is a user-defined threshold, during and at the end of training, respectively. Thresholding improves the regression by reducing PDE space into smaller intermediate PDEs by dropping out irrelevant terms during training. At the end of the process, the terms whose coefficients survive thresholding form the identified PDE.

In Algorithm 1, we present a pseudocode for finding the parameters of the system of PDEs using the Adjoint method (a flowchart is also shown in Fig. 18 of Appendix E). For the introduced hyperparameters, we note that β in the learning rate needs to be small enough to avoid unstable intermediate guessed PDEs, ϵ_0 must be large enough to ensure uniqueness in cases where more than one solution may exist, and the thresholding should be applied only when solution to the optimization is not improving anymore up to a user-defined tolerance γ_{thr}. For suggested default values, please see the description of the algorithm. For experimental investigation on impact of these parameters for a few examples, see Appendix F.

We note that the type of guessed PDE may change during the training, which adds numerical complexity to the optimization and motivates the use of an appropriate solver for each type of guessed PDE, e.g. Finite Volume method for hyperbolic and Finite Element method for Elliptic PDEs. For simplicity, in this work we use the second-order Finite Difference method across the board to estimate the spatial and Euler for the time derivative with small enough time step sizes in solving the forward/backward equations to avoid blow-ups due to possible instabilities. See appendix C for the analysis on the numerical error for the estimated adjoint gradient. We note that the adjoint method is most effective when there is some prior knowledge of the underlying PDE type, and a suitable numerical method is deployed.

Algorithm 1 Finding system of PDEs using Adjoint method. Default threshold $\sigma = 10^{-3}$ applied after $N_{\text{thr}} = 100$ iterations, with tolerances $\gamma = 10^{-9}$ and $\gamma_{\text{thr}} = 10^{-6}$, and regularization factor $\epsilon_0 = 10^{-12}$.

Input: data $\boldsymbol{f}^*$, learning rate η, tolerance γ, threshold σ applied after N_{thr}, and ϵ_0.
Initialize the parameters $\boldsymbol{\alpha} = \boldsymbol{0}$
repeat
  for $j = 0, \ldots, N_t - 1$ **do**
    Estimate $\boldsymbol{f}$ in $t \in (t^{(j)}, t^{(j+1)}]$ by solving forward model (1) given initial condition (8)
    Solve for $\boldsymbol{\lambda}$ in $t \in [t^{(j)}, t^{(j+1)})$ given the backward adjoint Eq. (6) and final condition Eq. 7 from data.
    Compute the gradient using Eq. (4)
    Update parameters $\boldsymbol{\alpha}$ using Eq. (5)
  end for
  if Epochs $> N_{\text{thr}}$ or convergence in $\boldsymbol{\alpha}$ with γ_{thr} **then**
    Thresholding: set $\alpha_i = 0$ for all i such that $|\alpha_i| < \sigma$
  end if
until Convergence in $\boldsymbol{\alpha}$ with tolerance γ
Output: $\boldsymbol{\alpha}$

Training with noisy data set. Often the data set comes with some noise. There are several pre-processing steps that can be done to reduce the noise at the expense of introducing bias, for example removing high frequencies using Fast Fourier Transform or removing small singular values from data set using Singular Value Decomposition.

However, we can also reduce the sensitivity of the training algorithm to the noise by averaging the gradients before updating the parameters. Assuming that the noise is martingale, the Monte Carlo averaging gives us the unbiased estimator for the expected value of the gradient over all the data set. We adapt the training procedure by averaging gradients over all available data points and then updating the parameters (see Algorithm 2 and the flowchart in Fig. 18 of Appendix E for more details). Clearly, this will make the algorithm more robust at higher cost since the update happens only after seeing all the data.

Algorithm 2 Finding a system of PDEs using the Adjoint method with averaging for the computation of gradients over the data set. Default threshold $\sigma = 10^{-3}$ applied after $N_{\text{thr}} = 100$ iterations, with tolerances $\gamma = 10^{-9}$ and $\gamma_{\text{thr}} = 10^{-6}$, and regularization factor $\epsilon_0 = 10^{-12}$.

Input: data $\boldsymbol{f}^*$, learning rate η, tolerance γ, threshold σ applied after N_{thr}, and ϵ_0.
Initialize the parameters $\boldsymbol{\alpha} = \mathbf{0}$
repeat
  for $j = 0, \ldots, N_t - 1$ **do**
    Estimate $\boldsymbol{f}$ in $t \in (t^{(j)}, t^{(j+1)}]$ by solving forward model (1) given initial condition (8)
    Solve for $\boldsymbol{\lambda}$ in $t \in [t^{(j)}, t^{(j+1)})$ given the backward adjoint Eq. (6) and final condition Eq. 7 from data.
    Compute the gradient $\boldsymbol{g}^{(j)} = \partial\mathcal{C}^{(j)}/\partial\boldsymbol{\alpha}$ using Eq. (4)
  end for
  Average the gradient $\mathbb{E}[\partial\mathcal{C}/\partial\boldsymbol{\alpha}] = \sum_j \boldsymbol{g}^{(j)}/N_t$
  Update parameters $\boldsymbol{\alpha}$ using Eq. (5) and $\mathbb{E}[\partial\mathcal{C}/\partial\boldsymbol{\alpha}]$
  if Epochs $> N_{\text{thr}}$ or convergence in $\boldsymbol{\alpha}$ with γ_{thr} **then**
    Thresholding: set $\alpha_i = 0$ for all i such that $|\alpha_i| < \sigma$
  end if
until Convergence in $\boldsymbol{\alpha}$ with tolerance γ
Output: $\boldsymbol{\alpha}$

3 Results

We demonstrate the validity of our proposed adjoint-based method in discovering PDEs given measurements on a spatial-temporal grid. We have compared our approach to PDE-FIND in terms of error and time to convergence. All the results are obtained using a single core-thread of a 2.3 GHz Quad-Core Intel Core i7 CPU. In this paper, we report the execution time τ obtained with averaging over 10 independent runs and we use error bars to show the standard deviation of the expected time, i.e. $d_{\text{error-bar}} = \sqrt{\mathbb{E}[(\tau - \mathbb{E}[\tau])^2]}$. Furthermore, we also report the *true positivity ratio* Lagergren et al. (2020) defined by

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}} \tag{9}$$

where TP denotes the number of correctly identified non-zero coefficients, FN the number of coefficients falsely identified as zero, and FP is the number of coefficients falsely identified as non-zero. Identification of the true PDE form results in TPR = 1.

3.1 Considered PDEs

In this section, we consider the PDE discovery task given data from numerical solutions to a variety of problems, including the Heat, Burgers', Kuramoto Sivashinsky, Random Walk, and Reaction Diffusion equations, summarized in Table 1.

Table 1: A summary of recovered PDEs from dataset using Adjoint and PDE-FIND method.

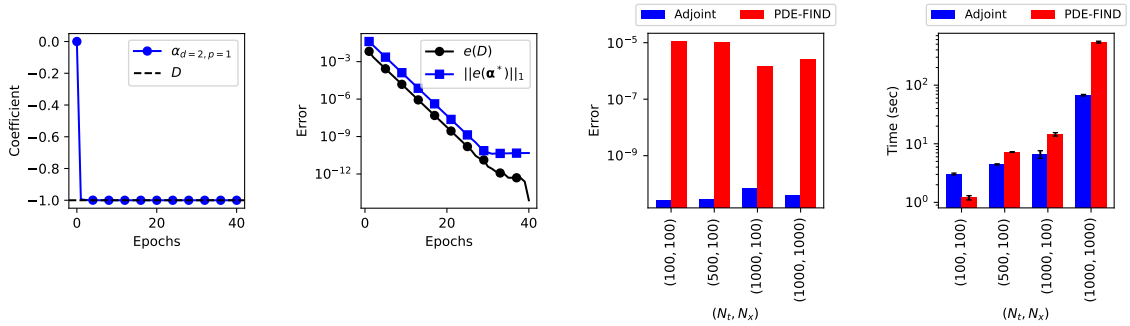
Problem $(N_t, N_{x_1}, N_{x_2}, ...)$	Method	Ex. Time [s]	Recovered PDE
Heat eq. (1D) (128,128)	Adjoint	2.14 ± 0.01	$f_t = f_{xx} + \mathcal{O}(10^{-12})$
	PDE-FIND	2.66 ± 0.02	$f_t = f_{xx} + \mathcal{O}(10^{-5})$
Heat eq. (2D) (100,100,100)	Adjoint	44.87 ± 1.63	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-6})$
	PDE-FIND	796.01 ± 11.94	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-6})$
Burgers' eq. (1D) (128,128)	Adjoint	3.37 ± 0.24	$f_t = (f^2)_x + \mathcal{O}(10^{-12})$
	PDE-FIND	2.56 ± 0.01	$f_t = -0.036f + 0.094f^3 + 2ff_x + 0.002f^3f_x + \mathcal{O}(10^{-4})$
Burgers' eq. (2D) (100,100,100)	Adjoint	250.4 ± 6.6	$f_t = (f^2)_{x_1x_1}(f^2)_{x_2x_2} + \mathcal{O}(10^{-4})$
	PDE-FIND	914.93 ± 8.32	$f_t = 1.998ff_{x_1x_1} + 1.998ff_{x_2x_2} + \mathcal{O}(10^{-4})$
KS eq. (1D) (64,256)	Adjoint	9.34 ± 1.2	$f_t = -0.5f_{xx} + 0.5f_{xxxx} + (f^2)_x + O(10^{-5})$
	PDE-FIND	1.12 ± 0.11	$f_t = -0.5f_{xx} + 0.5f_{xxxx} + 1.972ff_x + 0.042f + \mathcal{O}(10^{-3})$
Random Walk (1D) (50,100)	Adjoint	1.253 ± 0.38	$f_t + 1.025f_x - 0.465f_{xx} + \mathcal{O}(10^{-2}) = 0$
	PDE-FIND	0.17 ± 0.09	$f_t + 0.798f_x - 0.454f_{xx} + \mathcal{O}(10^{-2}) = 0$
Reaction Diffusion eqs. (2D) (200,70,70)	Adjoint	998.32 ± 14.66	$u_t = 0.1u_{xx} + 0.2u_{yy} + 0.3u - 0.3v^3 - 0.1uv^2 - 0.2u^2v + 0.4u^3 + \mathcal{O}(10^{-11})$, $v_t = 0.4v_{xx} + 0.3v_{yy} + 0.2v + 0.1v^3 - 0.2uv^2 - 0.3u^2v - 0.1u^3 + \mathcal{O}(10^{-10})$
	PDE-FIND	2234.54 ± 31.52	$u_t = 0.1u_{xx} + 0.2u_{yy} + 0.3u - 0.3v^3 - 0.1uv^2 - 0.2u^2v + 0.4u^3 + \mathcal{O}(10^{-9})$, $v_t = 0.4v_{xx} + 0.3v_{yy} + 0.2v + 0.1v^3 - 0.2uv^2 - 0.3u^2v - 0.1u^3 + \mathcal{O}(10^{-8})$

3.1.1 Heat equation

As a first example, let us consider measured data collected from the solution to the heat equation, i.e.

$$\frac{\partial f}{\partial t} + D\frac{\partial^2 f}{\partial x^2} = 0, \tag{10}$$

with $D = -1$. The data is constructed using the Finite Difference method with initial condition $f(x, 0) = 5\sin(2\pi x)x(x - L)$ and a mesh with $N_x = 100$ nodes in x covering the domain $\Omega = [0, L]$ with $L = 1$ and $N_t = 100$ steps in t with final time $T = N_t\Delta t$ where $\Delta t = 0.05\Delta x^2/|D|$ is the step size and $\Delta x = L/N_x$ is the mesh size in x.



(a) (b) (c) (d)

Figure 1: The estimated coefficient corresponding to D in heat equation (a) and the L_1-norm error of all considered coefficients (b) using the proposed Adjoint method with $N_t = 100$ and $N_x = 100$. Also, we show L_1-norm error of the estimated coefficients (c) and the execution time (d) using the proposed Adjoint method (blue) and PDE-FIND method (red), given data on a grid with $N_t \in \{100, 500, 1000\}$ steps in t, and $N_x \in \{100, 1000\}$ nodes in x.

We consider a system consisting of a single PDE (i.e. $N = 1$, $\boldsymbol{f} = f$, and $\boldsymbol{p} = p$) with one-dimensional input, i.e. $n = 1$ and $\boldsymbol{x} = x \subset \mathbb{R}$, and $\boldsymbol{d} = d \in \mathbb{N}$. In order to construct a general forward model, here we consider derivatives and polynomials with indices $d, p \in \{1, 2, 3\}$ as the initial guess for the forward model. This leads to 9 terms with unknown coefficients $\boldsymbol{\alpha}$ that we find using the proposed adjoint method (an illustrative

derivation of the candidate terms can be found in Appendix G.1). While we expect to recover the coefficient that corresponds to D, we expect all the other coefficients (denoted by $\boldsymbol{\alpha}^*$) to become negligible. That is what we indeed observe in Fig. 1 where the error of the coefficient for each term is plotted against the number of epochs.

Table 2: Comparison between Adjoint, PDE-FIND, WSINDy, and PDE-LEARN in recovering 1D Heat equation from noise-free data.

(N_t, N_x)	Method	Ex. Time [s]	Recovered PDE	TPR
(128,128)	Adjoint	2.14 ± 0.01	$f_t = f_{xx} + \mathcal{O}(10^{-12})$	1
	PDE-FIND	2.66 ± 0.02	$f_t = f_{xx} + \mathcal{O}(10^{-5})$	1
	WSINDy	0.20 ± 0.01	$f_t = 10.777 f_x - 20.701 f_x^3$	0.25
	PDE-LEARN	941.01 ± 2.13	$f_t = -0.004 f_x + 0.002 f_{xx} + 0.009 f_{xxx} - 0.006 f f_x - 0.007 f f_{xx}$ $-0.003 (f_x)^2 - 0.004 f^2 f_x - 0.004 f^2 f_{xx}$	0.22

Next, we compare the solution obtained via the adjoint method against PDE-FIND with STRidge optimization method. Here, we test both methods in recovering the heat equation given data on the grid with discretization (N_t, N_x). As shown in Fig. 1, the proposed adjoint method provides more accurate results across all data sizes.

We also point out that as the size of the data set increases, PDE-FIND with STRidge regression method becomes more expensive, e.g. one order of magnitude more expensive than the adjoint method for the data on a grid size $(N_t, N_x) = (1000, 1000)$. In Table 2, we also compare the adjoint method with more recent methods such as WeakSINDy and PDE-LEARN, where, to our surprise, PDE-FIND remains the strongest alternative that justifies its use as the baseline method here.

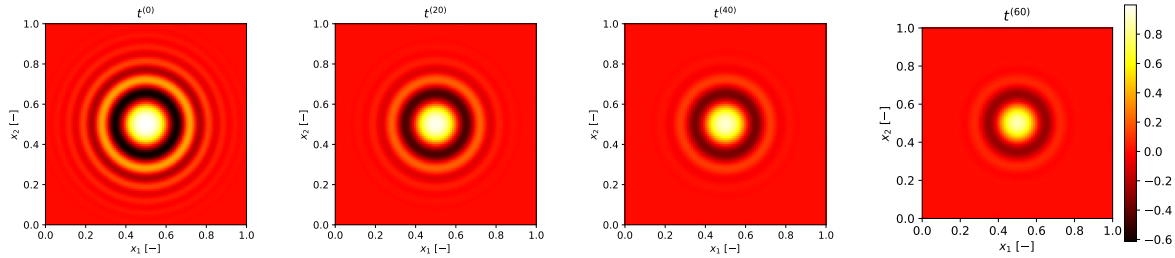
Next, we consider the Heat equation in 2D, i.e.

$$\frac{\partial f}{\partial t} + D\left(\frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2}\right) = 0 \tag{11}$$

with the initial condition

$$f(\boldsymbol{x}, 0) = \exp(-b(\boldsymbol{x} - \boldsymbol{x}_c)^2)\cos(2c\pi(\boldsymbol{x} - \boldsymbol{x}_c)^2) \tag{12}$$

where $\boldsymbol{x}_c = (0.5, 0.5)^T$ and coefficients $b = c = 20$, inside the domain $\boldsymbol{x} \in [0, 1]^2$, as depicted in Fig. 2. Again, we have tested the adjoint method against other methods for a variety of mesh sizes in Table 3. Overall, the adjoint method seems to provide a more efficient solution at larger data sets and higher dimensions.



(a) (b) (c) (d)

Figure 2: The initial condition described in Eq. 12 on 100×100 grid (a) and the evolution of the solution to the 2D heat equation after 20, 40, and 60 time steps (b)-(d).

Table 3: Recovering 2D Heat equation using the Adjoint and PDE-FIND method given noise-free dataset for a range of discretizations. *The method failed in providing any solutions.

(N_t, N_{x_1}, N_{x_2})	Method	Ex. Time [s]	Recovered PDE	TPR
(20,50,50)	Adjoint	11.4	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-5})$	1
	PDE-FIND	15.31	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-5})$	1
	WSINDy	$-^*$	$-^*$	$-^*$
(20,100,100)	Adjoint	21.96	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-5})$	1
	PDE-FIND	75.27	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-5})$	1
	WSINDy	0.139	$f_t = -0.0009 f_{x_1x_1x_1} + \mathcal{O}(10^{-6})$	0.25
(100,100,100)	Adjoint	44.87	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-6})$	1
	PDE-FIND	796.83	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-6})$	1
	WSINDy	0.152	$f_t = -0.02116 f_{x_2x_2x_2} + \mathcal{O}(10^{-6})$	0.25

3.1.2 Burgers' equation

As a nonlinear test case, let us consider the data from Burgers' equation given by

$$\frac{\partial f}{\partial t} + \frac{\partial (Af^2)}{\partial x} = 0 \tag{13}$$

where $A = -1$. The data is obtained with similar simulation setup as for heat equation (Section 3.1.1) except for the time step, i.e. $\Delta t = 0.05\Delta x/|A|$.

Similar to Section 3.1.1, we adopt a system of one PDE with one-dimensional input. We also consider derivatives and polynomials with indices $d, p \in \{1, 2, 3\}$ in the construction of the forward model. This leads to 9 terms whose coefficients we find using the proposed adjoint method. As shown in Fig. 3, the proposed adjoint method finds the correct coefficients, i.e. $\alpha_{d=1,p=2}$ that corresponds to D as well as all the irrelevant ones denoted by $\boldsymbol{\alpha}^*$, up to machine accuracy in $\mathcal{O}(10)$ epochs.

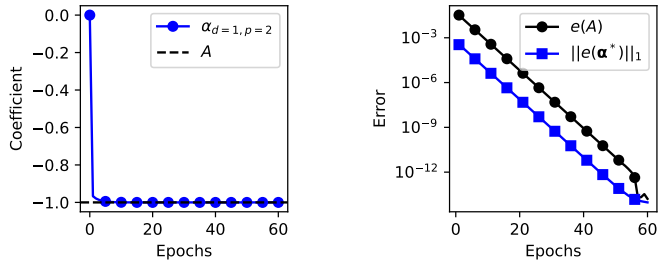


Figure 3: The estimated coefficient corresponding to A (left) and the L_1-norm error of all considered coefficients (right) given the discretized data of 1D Burgers' equation during training.

Next, we compare the solution obtained from the adjoint method to the one from PDE-FIND using STRidge optimization method. Here, we compare the error on coefficients and computational time between the adjoint and PDE-FIND by repeating the task for the data set with increasing size, i.e. $(N_t, N_x) \in \{(100, 100),\ (1000, 100),\ (1000,\ 1000)\}$. As depicted in Fig. 4, the adjoint method provides us with more accurate solution across the different mesh sizes. Regarding the computational cost, while PDE-FIND seems faster on smaller data sets, as the size of the data grows, it becomes increasingly more expensive than the adjoint method. Similar to the heat equation, for the mesh size $(N_t, N_x) = (1000,\ 1000)$ we obtain one order of magnitude speed-up compared to PDE-FIND. Again, we compare the adjoint method to WeakSINDy and PDE-LEARN as more recent alternative methods. As shown in Table 4, to our surprise, PDE-FIND remains the strongest alternative that justifies its use as the baseline method here.

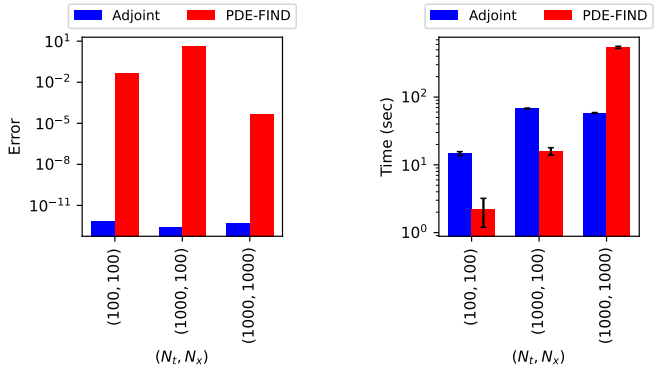


Figure 4: L_1-norm error of the estimated coefficients (left) and the execution time (right) for discovering the Burgers' equation equation using the Adjoint method (blue) and PDE-FIND method (red), given data on a grid with $N_t \in \{100, 1000\}$ steps in t, and $N_x \in \{100, 1000\}$ nodes in x.

Table 4: Comparison between the Adjoint, PDE-FIND, WSINDy, and PDE-LEARN in recovering the 1D Burgers' equation from noise-free data.

(N_t, N_x)	Method	Ex. Time [s]	Recovered PDE	TPR
(128,128)	Adjoint	3.37 ± 0.15	$f_t = (f^2)_x + \mathcal{O}(10^{-12})$	1
	PDE-FIND	2.56 ± 0.02	$f_t = 2ff_x - 0.03f + 0.09f^3 + \mathcal{O}(10^{-4})$	0.5
	WSINDy	0.21 ± 0.05	$f_t = -0.003f_{xxx}^2 + 0.016f_{xx}^3 + \mathcal{O}(10^{-4})$	0.25
	PDE-LEARN	889.15 ± 6.23	$f_t = 0.08f_x + 0.10f_{xx} + 0.01f_{xxx} + 0.03ff_x + 0.06f^2f_x + 0.034ff_{xx}$ $+0.06f^2f_{xx} - 0.05ff_{xxx} - 0.04f^2f_{xxx}$	0.2

Next, we consider the Burgers' equation in 2D, i.e.

$$\frac{\partial f}{\partial t} + A\left(\frac{\partial f^2}{\partial x_1} + \frac{\partial f^2}{\partial x_2}\right) = 0 \tag{14}$$

with $A = -1$ and the initial condition

$$f(\boldsymbol{x}, 0) = \exp(-b(\boldsymbol{x} - \boldsymbol{x}_c)^2) \tag{15}$$

with $\boldsymbol{x}_c = (0.5, 0.5)^T$ and coefficients $b = 30$, inside the domain $\boldsymbol{x} \in [0, 1]^2$, as depicted in Fig. 5. Again, we have tested the adjoint method against the PDE-FIND method in variety of mesh sizes as shown in Table 5. We observe that the adjoint method remains computationally advantageous at larger data sets.

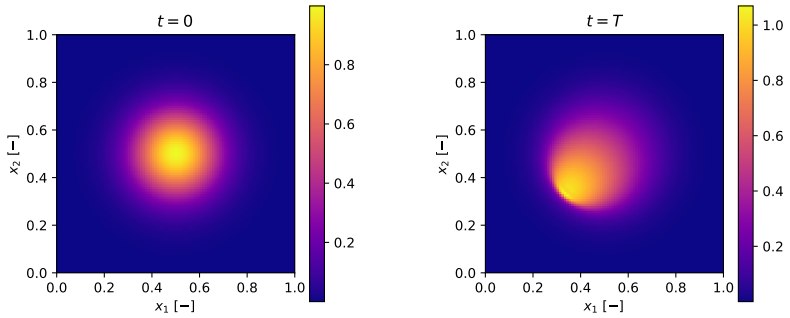


Figure 5: Initial condition (left) described in Eq. 15 and the solution of 2D Burgers' equation at final time (right).

Table 5: Recovering 2D Burgers' equation using the Adjoint method without averaging Algorithm 1 and PDE-FIND method given noise-free dataset for a range of discretizations. *The WSINDy method failed in providing any solutions.

(N_t, N_{x_1}, N_{x_2})	Method	Ex. Time [s]	Recovered PDE	TPR
(20,50,50)	Adjoint	148.4	$f_t = 0.979(f^2)_{x_1} + 0.988(f^2)_{x_2} + \mathcal{O}(10^{-4})$	1
	PDE-FIND	15.23	$f_t = 2.03ff_{x_1} - 2.02ff_{x_2} - 0.027f^3f_{x_1} - 0.029f^2f_{x_2}$ $-0.035f^3f_{x_2} - 0.002f_{x_1} - 0.002f_{x_2} + \mathcal{O}(10^{-4})$	0.37
	WSINDy	$-^*$	$-^*$	
(20,100,100)	Adjoint	145.6	$f_t = 0.989(f^2)_{x_1} + 0.989(f^2)_{x_2} + 0.004f_{x_1} + 0.007(f^3)_{x_1}$ $+0.004f_{x_2} - 0.007(f^3)_{x_2} + \mathcal{O}(10^{-4})$	0.42
	PDE-FIND	85.34	$f_t = 2.01ff_{x_1} + 2.01ff_{x_2} - 0.027f^3f_{x_1} - 0.012f^2f_{x_1} - 0.012f^2f_{x_2}$ $-0.004f^3f_{x_2} + 0.008f^3 - 0.002f_{x_1} - 0.002f_{x_2} + \mathcal{O}(10^{-4})$	0.3
	WSINDy	0.1346	$f_t = -0.001722(f^3)_{x_2x_2x_2} + \mathcal{O}(10^{-6})$	0.25
(100,100,100)	Adjoint	250.4	$f_t = (f^2)_{x_1} + (f^2)_{x_2} + \mathcal{O}(10^{-4})$	1
	PDE-FIND	914.93	$f_t = 1.998ff_{x_1} + 1.998ff_{x_2} + \mathcal{O}(10^{-4})$	1
	WSINDy	0.156	$f_t = -0.0106277(f^3)_{x_1x_1x_1} + \mathcal{O}(10^{-6})$	0.25

3.1.3 Kuramoto Sivashinsky equation

As a more challenging test case, let us consider the recovery of the Kuramoto-Sivashinsky (KS) equation given by

$$\frac{\partial f}{\partial t} + A\frac{\partial f^2}{\partial x} + B\frac{\partial^2 f}{\partial x^2} + C\frac{\partial^4 f}{\partial x^4} = 0 \tag{16}$$

where $A = -1$, $B = 0.5$ and $C = -0.5$. The data is generated in a similar way to previous sections except for the grid $(N_t, N_x) = (64, 256)$ and the time step size $\Delta t = 0.01\Delta x^4/|C|$.

Here again, we adopt a system of one PDE with one-dimensional input. As a guess for the forward model, we consider terms consisting of derivatives with indices $d \in \{1, 2, 3, 4\}$ and polynomials with indices $b \in \{1, 2\}$, leading to 8 terms whose coefficients we find using the proposed adjoint method. As shown in Fig. 6, the adjoint method finds the coefficient with error of $\mathcal{O}(10^{-5})$, yet achieving machine accuracy seems not possible.

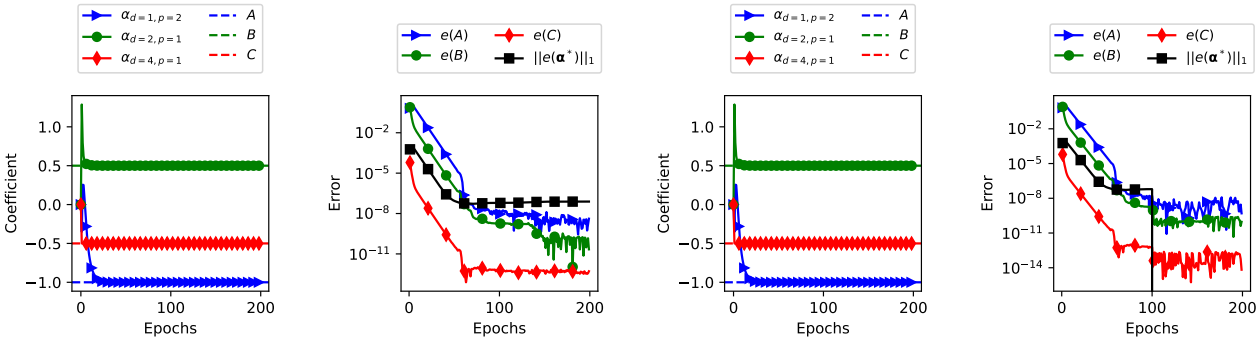


Figure 6: The estimated coefficients corresponding to A, B, C and the L_1-norm error of all considered coefficients given the discretized data of the KS equation during training without (a-b) and with (c-d) active thresholding for Epochs $> N_{\text{thr}} = 100$.

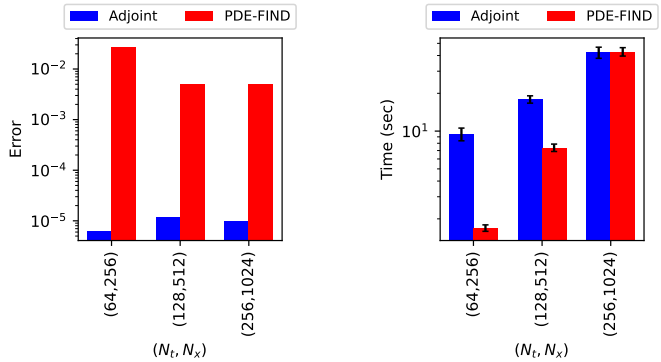


Figure 7: L_1-norm error of the estimated coefficients (left) and the execution time (right) for discovering the KS equation using the Adjoint method (blue) and PDE-FIND method (red), given data on a grid with $N_t \in \{64, 128, 256\}$ steps in t, $N_x \in \{256, 512, 1024\}$ nodes in x.

Table 6: Comparison between Adjoint, PDE-FIND, WSINDy, and PDE-LEARN method in recovering the Kuramoto Sivashinsky equation from data.

(N_t, N_x)	Method	Ex. Time [s]	Recovered PDE	TPR
(64,256)	Adjoint	79.14 ± 2.31	$f_t = -0.5f_{xx} + 0.5f_{xxxx} + (f^2)_x + \mathcal{O}(10^{-8})$	1
	PDE-FIND	26.20 ± 3.11	$f_t = -0.5f_{xx} + 0.5f_{xxxx} + 1.972ff_x + 0.042f + O(10^{-4})$	0.8
	WSINDy	0.20 ± 0.01	$f_t = -0.255(f^2)_{xxx}$	0.2
	PDE-LEARN	892.08 ± 7.72	$f_t = 0.04f - 0.03f^2 - 0.05f_x + 0.02f_{xx} + 0.05f_{xxx} + 0.04f_{xxxx}$ $-0.07ff_x + 0.03f^2f_x - 0.04ff_{xx} - 0.05f^2f_{xx} + 0.08ff_{xxx}$ $-0.04f^2f_{xxx} + 0.07ff_{xxxx} + 0.01f^2f_{xxxx}$	0.26

Again, in Fig. 7 we make a comparison between the predicted PDE using the adjoint method against PDE-FIND. In particular, we consider a data set on a temporal/spatial mesh of size $(N_t, N_x) = \{(64, 256), (128, 512), (256, 1024)\}$ and compare how the error and computational cost vary.

Similar to previous sections, the error is reported by comparing the obtained coefficients against the coefficients of the exact PDE in L_1-norm. Interestingly, the PDE-FIND method has 3 to 4 orders of magnitude larger error compared to the adjoint method. Also, in terms of cost, the training time for PDE-FIND seems to grow at a higher rate than the adjoint method as the (data) mesh size increases. Again, we made further comparison with more recent methods such as WeakSINDy and PDE-LEARN. As shown in Table 6, PDE-FIND remains the strongest alternative which justifies its use as the baseline method here.

3.1.4 Random Walk

Next, let us consider the recovery of the governing equation on probability density function (PDF) given samples of its underlying stochastic process. As an example, we consider the Itô process

$$dX = Adt + \sqrt{2D}dW \tag{17}$$

where $A = 1$ is drift and $D = 0.5$ is the diffusion coefficient, and W denotes the standard Wiener process with $\mathrm{Var}(dW) = \Delta t$. We generate the data set by simulating the random walk using Euler-Maruyama scheme starting from $X(t = 0) = 0$ for $N_t = 50$ steps with a time step size of $\Delta t = 0.01$. We estimate the PDF using histogram with $N_x = 100$ bins and $N_\mathrm{s} = 1000$ samples.

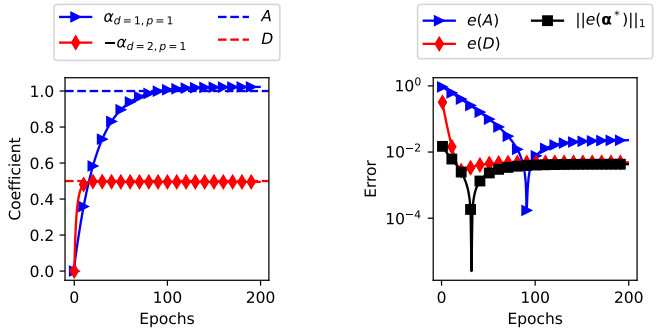


Figure 8: The estimated coefficients corresponding to A and D (left) and the L_1-norm error of all considered coefficients (right) of the Fokker-Planck equation as the governing law for the PDF associated with the random walk during training.

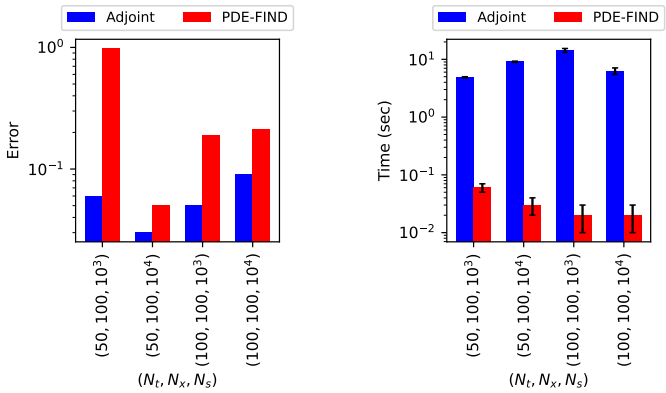


Figure 9: L_1-norm error of the estimated coefficients (left) and the execution time (right) for discovering the Fokker-Planck equation using the proposed Adjoint method (blue) and PDE-FIND method (red), given samples of its underlying stochastic process with $N_t \in \{50, 100\}$ steps in t, $N_x = 100$ histogram bins, and $N_s \in \{10^3, 10^4\}$ samples.

Let us denote the distribution of X by f. Itô's lemma gives us the Fokker-Planck equation

$$\frac{\partial f}{\partial t} + A\frac{\partial f}{\partial x} - D\frac{\partial^2 f}{\partial x^2} = 0 \ . \tag{18}$$

Given the data set for f on a mesh of size (N_t, N_x), we can use Finite Difference to compute the contributions from derivatives of f in the governing law. Since this is one of the challenging test cases due to noise, here we only consider three possible terms in the forward model, consisting of derivatives with indices $d \in \{1, 2, 3\}$ and polynomial power $p = 1$. In Fig. 8, we show how the error of finding the correct coefficients evolves during training for the adjoint method. Clearly, the adjoint method seems to recover the true PDE with L_1 error of $\mathcal{O}(10^{-2})$ in its coefficients.

In Fig. 9, we make a comparison with PDE-FIND for the same number and order of terms as the initial guess for the PDE. We compare the two methods for a range of grid and sample sizes, i.e. $N_t \in \{50, 100\}$, $N_x = 100$, and $N_s \in \{10^3,\ 10^4\}$. It turns out that the proposed adjoint method overall provides more accurate estimate of the coefficients than PDE-FIND, though at a higher cost. In Table 7, we show the discovered PDEs for both methods across the different discretizations, and Table 8 shows further comparison wtih WSINDy and PDE-LEARN.

Table 7: Recovery of the Fokker-Planck equation, i.e. $f_t + f_x - 0.5 f_{xx} = 0$, using the proposed adjoint method against PDE-FIND method given samples of the underlying stochastic process for various discretization parameters.

N_t	N_x	N_s	Method	Recovered PDE	TPR
50	100	1000	Adjoint	$f_t + 1.025 f_x - 0.465 f_{xx} = 0$	1
			PDE-FIND	$f_t + 0.798 f_x - 0.454 f_{xx} = 0$	1
		10000	Adjoint	$f_t + 1.022 f_x - 0.495 f_{xx} = 0$	1
			PDE-FIND	$f_t + 0.818 f_x - 0.496 f_{xx} = 0$	1
100	100	1000	Adjoint	$f_t + 1.010 f_x - 0.543 f_{xx} = 0$	1
			PDE-FIND	$f_t + 0.863 f_x - 0.560 f_{xx} = 0$	1
		10000	Adjoint	$f_t + 1.015 f_x - 0.589 f_{xx} = 0$	1
			PDE-FIND	$f_t + 0.894 f_x - 0.612 f_{xx} = 0$	1

Table 8: Comparison between Adjoint, PDE-FIND, WSINDy, and PDE-LEARN in recovering the Fokker-Planck equation corresponding to the Random Walk from data.

(N_t, N_x)	Method	Ex. Time [s]	Recovered PDE	TPR
(50,100)	Adjoint	1.25 ± 0.01	$f_t + 1.025 f_x - 0.465 f_{xx} = 0$	1
	PDE-FIND	0.17 ± 0.02	$f_t + 0.798 f_x - 0.454 f_{xx} = 0$	1
	WSINDy	0.19 ± 0.01	$f_t = 7.8766 f$	0.25
	PDE-LEARN	69.10 ± 2.13	$f_t = 0.0479 f_x + 0.0228 f_{xx} - 0.0014 f_{xxx}$	0.75

3.1.5 Reaction Diffusion System of Equations

In order to show scalability and accuracy of the adjoint method for a system of PDEs in a higher dimensional space, let us consider a system of PDEs given by

$$\frac{\partial u}{\partial t} + c_0^u \nabla^2_{x_1}[u] + c_1^u \nabla^2_{x_2}[u] + R^u(u,v) = 0, \tag{19}$$

$$\frac{\partial v}{\partial t} + c_0^v \nabla^2_{x_1}[v] + c_1^v \nabla^2_{x_2}[v] + R^v(u,v) = 0 \tag{20}$$

where

$$R^u(u,v) = c_2^u u + c_3^u u^3 + c_4^u u v^2 + c_5^u u^2 v + c_6^u v^3 \tag{21}$$

$$R^v(u,v) = c_2^v v + c_3^v v^3 + c_4^v v u^2 + c_5^v v^2 u + c_6^v u^3 \tag{22}$$

We construct the data set by solving the system of PDEs Eqs. 19-20 using a 2nd order Finite Difference scheme with initial values

$$u_0 = a \sin\left(\frac{4\pi x_1}{L_1}\right) \cos\left(\frac{3\pi x_2}{L_2}\right) \left(L_1 x_1 - x_1^2\right) \left(L_2 x_2 - x_2^2\right)$$

$$v_0 = a \cos\left(\frac{4\pi x_1}{L_1}\right) \sin\left(\frac{3\pi x_2}{L_2}\right) \left(L_1 x_1 - x_1^2\right) \left(L_2 x_2 - x_2^2\right)$$

where $a = 100$, and the coefficients

$$\boldsymbol{c}^u = [c_i^u]_{i=0}^6 = [-0.1, -0.2, -0.3, -0.4, 0.1, 0.2, 0.3]$$

$$\boldsymbol{c}^v = [c_i^v]_{i=0}^6 = [-0.4, -0.3, -0.2, -0.1, 0.3, 0.2, 0.1].$$

We generate data by solving the system of PDEs Eqs. (19)-(20) using the Finite Difference method and forward Euler scheme for $N_t = 25$ steps with a time step size of $\Delta t = 10^{-6}$, and in the domain $\Omega = [0, L_1] \times [0, L_2]$

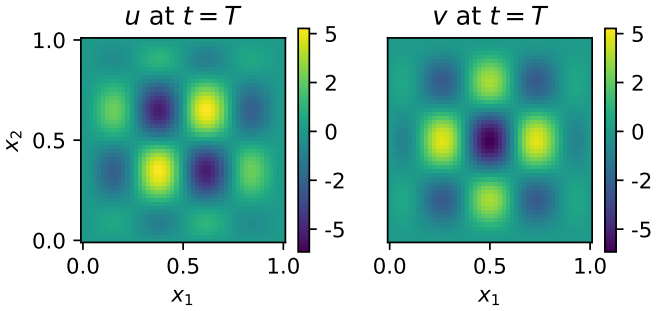


Figure 10: Solution to the reaction diffusion system of PDEs at time $t = T$ for u (left) and v (right).

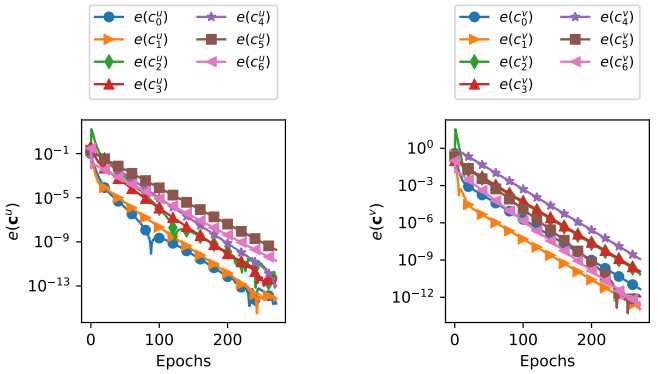


Figure 11: L_1-norm error in the estimated coefficients of the reaction diffusion system of PDEs during training.

where $L_1 = L_2 = 1$ which is discretized using a uniform grid with $N_{x_1} \times N_{x_2} = 50^2$ nodes leading to mesh size $\Delta x_1 = \Delta x_2 = 0.02$. In Fig. 10 we show the solution to the system at time $T = N_t \Delta t$ for u and v.

We consider a system consisting of two PDEs, i.e. $N = \dim(\boldsymbol{f}) = \dim(\boldsymbol{p}) = 2$, with two-dimensional input, i.e. $n = \dim(\boldsymbol{x}) = \dim(\boldsymbol{d}) = 2$. Here, $\dim(\boldsymbol{f}) = \dim(\text{Ima}(\boldsymbol{f}))$, where $\text{Ima}(\cdot)$ denotes the image (or output) of a function.

In order to use the developed adjoint method, we construct a guess forward system of PDEs (or forward model) using derivatives up to 2nd order and polynomials of up to 3rd order. That is, $d_{\max} = 2$ and $p_{\max} = 3$. This leads to 90 terms whose coefficients we find using the proposed adjoint method (an illustrative derivation of the candidate terms can be found in Appendix G.2). The solution to the constructed model $\boldsymbol{f} \approx [u, v]$ as well as the adjoint equation for $\boldsymbol{\lambda}$ is found using the same discretization as the data set.

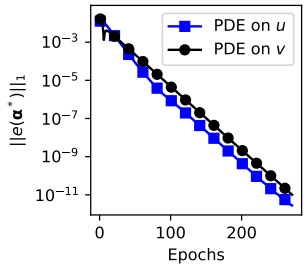


Figure 12: L_1-norm error in the estimated coefficients of the irrelevant terms compared to the true reaction diffusion system of PDEs during training, i.e. $||e(\boldsymbol{\alpha}^*)||_1 = ||\boldsymbol{\alpha}^*||_1$.

As shown in Fig. 11, the adjoint method finds the correct equations with error up to $\mathcal{O}(10^{-12})$. Furthermore, the coefficients corresponding to the irrelevant terms $\boldsymbol{\alpha}^*$ tend to zero with error of $\mathcal{O}(10^{-11})$, see Fig. 12. Furthermore, we have compared the adjoint method against PDE-FIND for a range of grid sizes in Fig. 13. We observe that the cost of PDE-FIND grows with higher rate than adjoint method as the size of the data set increases.

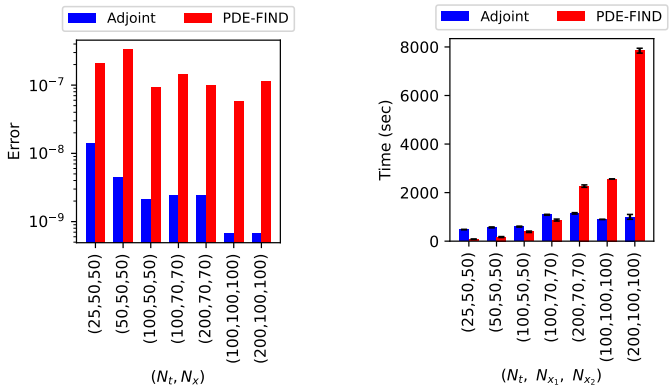


Figure 13: Comparing the error and execution time of the adjoint method (blue) to PDE-FIND method (red) against the size of the data set for the tolerance of 10^{-7} in the discovered coefficients.

3.1.6 Wave equation

Consider wave equation

$$f(x,t) = \sin(x-t) \tag{23}$$

which is a solution to infinite PDEs. For example, one class of PDEs with solution f is

$$f_t + kf_x + (k-1)f_{xxx} + c(f_{xx} + f_{xxxx}) = 0 \quad \forall k \in \mathbb{N} \ \text{ and } \ c \in \mathbb{R}\ , \tag{24}$$

defined in a domain $x \in [0, 2\pi]$ and $T = 1$. We create a data set using the exact f on a grid with $N_t = 10$ time intervals and $N_x = 100$ spatial discretization points.

Let us consider a similar setup as the heat equation example 3.1.1 with derivatives and polynomials indices $d \in \{1, ..., 6\}$ and $p = 1$ as the initial guess for the forward model. This leads to 6 terms with unknown coefficients $\boldsymbol{\alpha}$. Here, we enable averaging and use a finer discretization in time (100 steps for forward and backward solvers in each time interval) to cope with the instabilities of the Finite Difference solver due to the inclusion of the high-order derivatives. We also disable thresholding except at the end of the algorithm.

As shown in Fig. 14, the proposed Adjoint method returns the solution

$$f_t + 0.996 f_x = 0 \tag{25}$$

which is the PDE with the least number of terms compared to all possible PDEs. We note that for the same problem setting, PDE-FIND identifies the same form of the PDE, i.e.

$$f_t + 0.9897 f_x = 0\ . \tag{26}$$

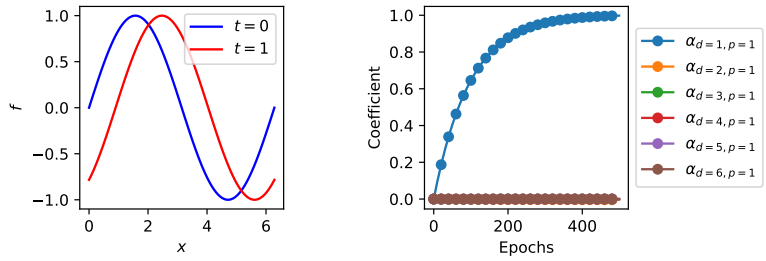


Figure 14: Profile of f at $t = 0$ and $t = 1$ (left) and the evolution of considered coefficients during adjoint optimization (right)

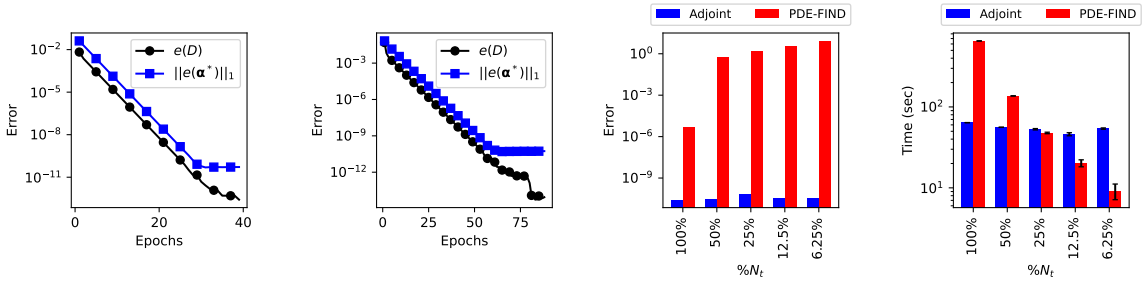
3.2 Partial observations in time

Here, we investigate how the error of the discovery task increases when only a subset of the fine data set is available. Consider the heat equation presented in section 3.1.1 and consider a data set created by solving the exact PDE using the Finite Difference method with $\Delta t = T/N_t$ where $N_t = 1000$ and $\Delta x = L/N_x$ and $N_x = 1000$.

Let us assume that we are only provided with a subset of this data set. As a test, let us take every ν time step as the input for the PDE discovery task, where $\nu \in \{1, 2, 4, 8, 16\}$. This corresponds to using $\{100, 50, 25, 12.5, 6.25\}\%$ of the total data set. By doing so, the accuracy of the Finite Difference method in estimating the time derivatives using the available data deteriorates, leading to large error in PDE discover task for PDE-FIND method.

However, the adjoint method can use a finer mesh in time compared to the data set in computing the forward and backward equations and only compare the solution to the data on the coarse mesh where data is available. We use $N_t = 1000$ for the forward and backward solvers in the adjoint method, and impose the final time condition where data is available. As shown in Table 9 and Fig. 15 the proposed adjoint method is able to recover the exact PDE regardless of how sparse the data set is in time.

We emphasize that while adjoint method can use a finer discretization in time than the one for data on $\mathcal{G}$ in solving forward and backward equations, it is bound to use similar or coarser spatial discretization as $\mathcal{G}$. This is due to the fact that the data points $\boldsymbol{f}^*$ are used for the initial condition of the forward model eq. 8, and the final condition of the backward adjoint equation eq. 7.

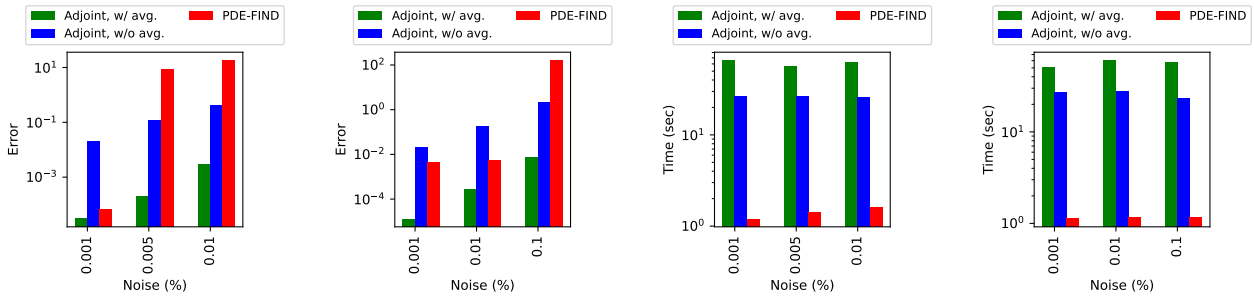


(a) (b) (c) (d)

Figure 15: Evolution of the L_1-norm error in coefficients of all considered terms using adjoint method when only (a) 50% and (b) 6.25% (b) of the data set is available. Error and execution time of the adjoint method (blue) and PDE-FIND method (red) in finding the coefficients of true heat equation given sparse data set in time in (c) and (d).

Table 9: Recovering the heat equation given sparse dataset in time. Here we rounded the coefficients up to three decimals.

$\%N_t$	Method	Recovered PDE	TPR
100	Adjoint	$f_t - f_{xx} = 0$	1
	PDE-FIND	$f_t - f_{xx} = 0$	1
50	Adjoint	$f_t - f_{xx} = 0$	1
	PDE-FIND	$f_t - 0.999f_{xx} + 0.177f - 0.261f^3 - 0.089ff_x$ $-0.011f^3f_x - 0.003f^2f_{xx} - 0.001ff_{xxx} = 0$	0.25
25	Adjoint	$f_t - f_{xx} = 0$	1
	PDE-FIND	$f_t - 0.999f_{xx} + 0.532f - 0.778f^3 - 0.268ff_x$ $-0.035f^3f_x - 0.010f^2f_{xx} - 0.003ff_{xxx} = 0$	0.25
12.5	Adjoint	$f_t - f_{xx} = 0$	1
	PDE-FIND	$f_t - 0.999f_{xx} + 1.264f - 1.863f^3 - 0.638ff_x - 0.081f^3f_x$ $-0.025f^2f_{xx} - 0.007ff_{xxx} - 0.001f^3f_{xxx} = 0$	0.22
6.25	Adjoint	$f_t - f_{xx} = 0$	1
	PDE-FIND	$f_t - 0.999f_{xx} + 2.769f - 4.051f^3 - 1.398ff_x - 0.185f^3f_x$ $-0.055f^2f_{xx} - 0.016ff_{xxx} - 0.002f^3f_{xxx} = 0.$	0.22



(a) (b) (c) (d)

Figure 16: Error and execution time of the adjoint method with (green) and without averaging the gradients (blue), along with the PDE-FIND method (red) in finding the coefficients of the true PDE, i.e. heat equation (a)-(c) and Burgers' equation (b)-(d), given noisy data.

3.3 Sensitivity to noise

Let us investigate how the error increases once noise is added to the data set. In particular, we add noise to each point of the data set for $\boldsymbol{f}^*$ via $\boldsymbol{f}^*(1+\epsilon)$, where the noise to signal ratio is $(\epsilon\boldsymbol{f}^*)/\boldsymbol{f}^* = \epsilon \sim \mathcal{N}(0, \sigma^2)$ with $\mathcal{N}(0, \sigma^2)$ denoting a normal distribution with zero mean and standard deviation of σ. As test cases, we revisit the heat (section 3.1.1) and Burgers' equations (section 3.1.2) with added noise of ϵ with $\sigma \in \{0.001,\ 0.005,\ 0.01,\ 0.1\}$ %. Before searching for the PDE, we first denoise the data set using Singular Value Decomposition and drop out terms with singular value below a threshold of $\mathcal{O}(10^{-4})$. Here, we report the expected coefficients of discovered PDE and execution time by repeating the process 10 times.

As shown in Fig. 16, adding noise to the dataset deteriorates the accuracy in finding the correct coefficients of the underlying PDE for both the adjoint and PDE-FIND method. We observe that the adjoint method, both with and without gradient averaging, is less susceptible to noise compared to PDE-FIND, albeit at a higher computational cost. Additionally, averaging the gradients in the adjoint method improves the accuracy around two orders of magnitude at higher computational cost.

Let us again revisit the Heat and Burgers' equation test cases in 2D, and compare the adjoint method with averaging gradient algorithm 2 against PDE-FIND method without applying any noise reduction in Table 10- 11. Although the accuracy of both methods deteriorates with noise, the adjoint method seems to provide more reliable solution.

Table 10: Recovering 2D Heat equation given noisy data on the grid $(N_t, N_{x_1}, N_{x_2}) = (50, 100, 100)$ using the adjoint method with averaging and learning rate parameter $\beta = 0.02$ in Algorithm 2 and PDE-FIND. $-^*$: Given the raw noisy data, PDE-FIND was not able to find any PDE, as it lead to run-time errors. Here, we also report the discovered PDE for denoised data with SVD.

σ	Denoised	Method	Time [s]	Recovered PDE	TPR
0.1 %	no	Adjoint	108.5	$f_t = f_{x_1x_1} + f_{x_2x_2} + 0.002f^3_{x_1x_1} + 0.003f^3_{x_2x_2} + \mathcal{O}(10^{-4})$	0.6
	yes		110.5	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-4})$	1
	no	PDE-FIND	235.40	$f_t = 1.003f_{x_1x_1} + 1.02f_{x_2x_2} + 14.44f + 37.70f^2 + 0.05f^2f_{x_1x_1}$ $+0.14f^3f_{x_1x_1} + 0.06f^2f_{x_2x_2} + 0.15f^3f_{x_2x_2} + \mathcal{O}(10^{-3})$	0.33
	yes		227.1	$f_t = f_{x_1x_1} + f_{x_2x_2} + \mathcal{O}(10^{-4})$	1
1 %	no	Adjoint	110.2	$f_t = 0.963f_{x_1x_1} + 1.008f_{x_2x_2} + 0.147f^3_{x_2x_2}$ $+0.010f^3_{x_1x_1} + 0.002f^2_{x_1x_1} + \mathcal{O}(10^{-4})$	0.5
	yes		109.19	$f_t = f_{x_1x_1} + f_{x_2x_2} + 0.024f^2_{x_2} + 0.027f^3_{x_2} + 0.031f^3_{x_2x_2} + \mathcal{O}(10^{-3})$	0.5
	no	PDE-FIND	$-^*$	$-^*$	$-^*$
	yes		245.3	$f_t = -0.36 + 1.01f_{x_1x_1} + 1.01f_{x_2x_2} + 57.08f + 110.54f^2 + 32.68f^3$ $+1.78f^2f_{x_1} + 3.20f^3f_{x_2} - 1.06ff_{x_2} - 0.77ff_{x_1} + 0.20f^2f_{x_1x_1}$ $+0.44f^3f_{x_1x_1} - 0.10f^2f_{x_1x_2} + 0.24f^2f_{x_2x_2} + 0.64f^3f_{x_2x_2} + \mathcal{O}(10^{-2})$	0.18
5 %	no	Adjoint	105.43	$f_t = 1.30f_{x_1x_1} + 1.32f_{x_2x_2} - 0.13f_{x_1} + 0.50f^2_{x_1} + 0.18f^3_{x_1} - 0.23f^2_{x_2}$ $-0.19f^3_{x_1} - 0.22f^2_{x_1x_1} - 0.20f^2_{x_2x_2} + \mathcal{O}(10^{-2})$	0.3
	yes		106.21	$f_t = 1.06f_{x_1x_1} + 1.07f_{x_2x_2} + 0.91f^3_{x_1x_1} + 0.87f^3_{x_2x_2} + 0.12f^2_{x_2} - 0.11f^3_{x_1}$ $-0.15f^3_{x_3} + 0.11f^2_{x_1x_1} - 0.10f^2_{x_2x_2} + \mathcal{O}(10^{-2})$	0.3
	no	PDE-FIND	$-^*$	$-^*$	$-^*$
	yes		248.53	$f_t = 1771.33f - 2369.49f^2 + 3086.75f^3 + 4.82ff_{x_1} + 11.41f^2f_{x_1}$ $-26.46f^3f_{x_1} - 4.40ff_{x_2} + 7.77f^3f_{x_2} - 1.47ff_{x_1x_1} + 7.40f^2f_{x_1x_1}$ $-0.81f_{x_1} + 0.34f_{x_2} + 1.45f_{x_1x_1} + 1.46f_{x_2x_2} + -1.99ff_{x_2x_2}$ $+7.37f^2f_{x_2x_2} + 2.81f^3f_{x_2x_2} + \mathcal{O}(10^{-1})$	0.16

Table 11: Recovering 2D Burgers' equation given raw noisy data on the grid $(N_t, N_{x_1}, N_{x_2}) = (50, 100, 100)$ using the adjoint method with averaging and learning rate parameter $\beta = 0.01$ Algorithm 2 and PDE-FIND. $-^*$: Given the raw noisy data, PDE-FIND was not able to find any PDE, as it lead to run-time errors. Here, we also report the discovered PDE given noisy dataset that is denoised using SVD.

σ	Denoised	Method	Time [s]	Recovered PDE	TPR
0.1 %	no	Adjoint	210.2	$f_t = 0.971(f^2)_{x_1} + 0.968(f^2)_{x_2} + 0.024f^3_{x_1} + 0.027f^3_{x_2} + \mathcal{O}(10^{-3})$	0.6
	yes		214.09	$f_t = 0.995(f^2)_{x_1} + 0.992(f^2)_{x_2} + \mathcal{O}(10^{-3})$	1
	no	PDE-FIND	280.71	$f_t = 0.11f_{x_2x_2} - 42.29f + 140.06f^2 - 65.64f^3 + 0.21ff_{x_1} + 4.51f^2f_{x_1} - 2.01f^3f_{x_1}$ $+0.17ff_{x_2} + 4.72f^2f_{x_2} - 2.14f^3f_{x_2} + 0.77ff_{x_1x_1} - 0.94f^2f_{x_1x_1} + 0.31f^3f_{x_1x_1}$ $+0.79ff_{x_2x_2} - 1.01f^2f_{x_2x_2} + 0.38f^3f_{x_2x_2} + \mathcal{O}(10^{-2})$	0.17
	yes		99.32	$f_t = 1.990ff_{x_1} + 1.990ff_{x_2} + \mathcal{O}(10^{-3})$	1
1 %	no	Adjoint	220.2	$f_t = 0.52(f^2)_{x_1} + 0.51(f^2)_{x_2} + 0.09f_{x_1} + 0.41(f^3)_{x_1} - 0.09f_{x_1} + 0.51(f^3)_{x_2}$ $+0.13f_{x_2x_2} - 0.01(f^3)_{x_1x_1} - 0.13f_{x_2x_2} - 0.02(f^3)_{x_2x_2} + \mathcal{O}(10^{-3})$	0.27
	yes		231.0	$f_t = 0.965(f^2)_{x_1} + 0.948(f^2)_{x_2} + 0.028(f^3)_{x_1} + 0.0118f_{x_1}$ $-0.013f_{x_1} - 0.032(f^3)_{x_2} + \mathcal{O}(10^{-3})$	0.42
	no	PDE-FIND	279.41	$f_t = 0.18f_{x_1x_1} + 0.19f_{x_2x_2} - 53.15f + 171.85f^2 - 83.15f^3 + 3.22f^2f_{x_1}$ $+3.24f^2f_{x_2} + 0.78ff_{x_1x_1} - 1.22f^2f_{x_1x_1} + 0.57f^3f_{x_1x_1} + 0.75ff_{x_2x_2}$ $-1.23f^2f_{x_2x_1} + 0.62f^3f_{x_2x_2} + \mathcal{O}(10^{-2})$	0.06
	yes		290.01	$f_t = -14.53f + 32.98f^2 + 17.51f^3 + 6.78f^2f_{x_2} + 6.51f^3f_{x_1} - 1.51ff_{x_2} - 1.04f^2f_{x_1x_1}$ $+0.70ff_{x_1x_1} + 0.66f^3f_{x_1x_1} + 0.69ff_{x_2x_2} - 0.92f^2f_{x_2x_2} + 0.52f^3f_{x_2x_2} + \mathcal{O}(10^{-2})$	0.14
5 %	no	Adjoint	240.9	$f_t = 0.515(f^2)_{x_1} + 0.393(f^2)_{x_2} + 1.193(f^3)_{x_1} - 0.788(f^3)_{x_2} - 0.142f_{x_1}$ $+0.788(f^3)_{x_1} + 0.051(f^3)_{x_1} - 0.0569(f^3)_{x_2} + \mathcal{O}(10^{-3})$	0.33
	yes		242.90	$f_t = 0.678(f^2)_{x_1} + 0.509(f^2)_{x_2} + 0.504(f^3)_{x_1} + 0.445(f^3)_{x_2} + \mathcal{O}(10^{-2})$	0.6
	no	PDE-FIND	$-^*$	$-^*$	$-^*$
	yes		288.91	$f_t = 171.85f^2 - 83.15f^3 - 53.15f + 3.22f^2f_{x_1} + 3.24f^2f_{x_2} - 1.23f^2f_{x_2x_2}$ $-1.22f^2f_{x_1x_1} + 0.19f_{x_2x_2} + 0.18f_{x_1x_1} + 0.78ff_{x_1x_1} + 0.57f^3f_{x_1x_1}$ $+0.75ff_{x_2x_2} + 0.62f^3f_{x_2x_2} + \mathcal{O}(10^{-2})$	0.07

3.4 Addressing ill-posedness

There may exist more than one PDE that replicates the data set. Therefore, the PDE discovery task is ill-posed due to the lack of uniqueness in the solution. This is an indication that further physically motivated constraints are needed to narrow the search space to find the desired PDE. However, among all possible PDEs, which PDE is found by the Adjoint method with the loss function defined as Eq. 3?

To answer this question, let us consider a simple example of the wave equation

$$f(x,t) = \sin(x-t) \tag{27}$$

which is a solution to infinite PDEs. For example, one class of PDEs with solution f is

$$f_t + kf_x + (k-1)f_{xxx} + c(f_{xx} + f_{xxxx}) = 0 \quad \forall k \in \mathbb{N},\ c \in \mathbb{R} \tag{28}$$

defined in a domain $x \in [0, 2\pi]$ and $T = 1$. We create a data set using the exact f on a grid with $N_t = 10$ time intervals and $N_x = 100$ spatial discretization points. Let us consider a similar setup as the heat equation example 3.1.1 with derivatives and polynomials indices $d \in \{1, ..., 6\}$ and $p = 1$ as the initial guess for the forward model. This leads to 6 terms with unknown coefficients $\boldsymbol{\alpha}$. Here, we enable averaging and use a finer discretization in time (100 steps for forward and backward solvers in each time interval) to cope with the instabilities of the Finite Difference solver due to the inclusion of the high-order derivatives. We also disable thresholding except at the end of the algorithm.

The proposed Adjoint method returns the solution

$$f_t + 0.996 f_x = 0 \tag{29}$$

which is the PDE with the least number of terms compared to all possible PDEs. We note that for the same problem setting, PDE-FIND finds

$$f_t + 0.9897 f_x = 0\ . \tag{30}$$

The identified form of PDE can be explained by the use of regularization term in the cost function 3, which enforces the minimization of the PDE coefficients. Clearly, the regularization term may be changed to find other possible solutions of this ill-posed problem.

3.5 Incomplete guessed PDE space

In this section, we investigate the outcome of the adjoint method when the exact terms are not included in the initial guessed PDE form. Here, we define the space of PDE where the exact terms are included in the general forward model 1 as complete. If the considered general form of PDE 1 does not include all the terms of the exact PDE, we denote that as an incomplete guessed PDE space.

Let us take the data from the numerical solution to Burgers' equation used in section 3.1.2 with discretization $N_t = N_x = 100$. For the complete forward model, we again consider derivatives and polynomials with indices $d, p \in \{1, 2, 3\}$ in the construction of the forward model. This leads to 9 terms whose coefficients we find using the proposed adjoint method. For the incomplete space of PDE, we take derivatives and polynomials with indices as $d \in \{1, 2, 3\}$ and $p \in \{1, 3\}$, leading to 6 terms. Clearly, the incomplete guessed PDE space does not include the term $\alpha_{d=1,p=2}\partial f^2/\partial x$. Now, we would like to see which PDE is returned by the adjoint method.

In Fig. 17, we made a comparison between the evolution of coefficients and L_2 norm error of the estimated forward model against the data. While the complete space monotonically converges to the exact solution up to machine accuracy, the incomplete space of PDE delivers another PDE, i.e.

$$\frac{\partial f}{\partial t} + \frac{\partial^2}{\partial x^2}(0.04f - 0.01f^3) = 0, \tag{31}$$

with the relative L_2 error of $\mathcal{O}(10^{-5})$ between forward model estimation and the data points. The fact that the L_2 error between f and f^* does not decrease is an indication that the considered space of PDE is

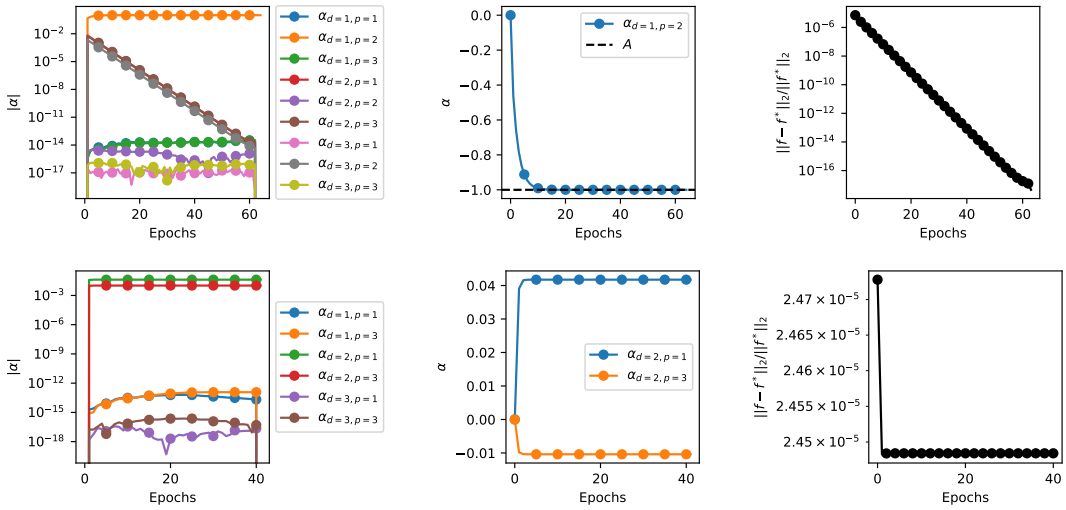


Figure 17: The adjoint method applied to the Burgers' equation for complete (top) and incomplete (bottom) space of guessed PDEs.

incomplete and additional terms must be included. We note that here we assumed there is no noise in the data set. However, in the presence of noise, the L_2 error between f and f^* may stagnate at the noise level, which makes the analysis on the completeness of the PDE space more challenging.

4 Discussion

Below we highlight and discuss strengths and weaknesses of the proposed adjoint method.

Strengths. The proposed method has several strengths:

1. The proposed adjoint-based method of discovering PDEs can provides coefficients of the true governing equation with significant accuracy.

2. Since the gradient of the cost function with respect to parameters are derived analytically, the optimization problem converges fast. In particular, the adjoint method becomes cheaper than PDE-FIND as the size of the data set increases. The adjoint method by construction finds the optimal relation between the gradient of the cost function and the error in the data points. This was achieved by finding the extremum of the objective functional using the variational derivative.

   We note that a clear difference from the point-wise loss $||f - f^*||_2$ used in the PDE-FIND method is that for the adjoint method weights the error at discrete points with the Lagrange multipliers; see Eq. 4 and the final condition Eq. 7. For Adjoint method more data often means finer resolutions which allows solving the forward/backward equations more accurately on a finer grid, and obtain higher accuracy in estimating the gradients that lead to less number of epochs in training, see Appendix C.

3. Since the adjoint method uses a PDE solver to find the underlying governing equation, there is a guarantee that the recovered PDE has a solution and can be solved numerically.

4. The adjoint method can use a finer mesh in time compared to the available discretization of the data set. This allows an accurate recovery of the underlying PDE compared to the PDE-FIND, where the error in the latter increases as the data set gets coarser since it estimates derivatives directly (either with Finite Difference or a polynomial fit) using the given data set.

Weaknesses. Our proposed method has some limitations:

1. In order to use the proposed adjoint method for discovering PDEs, a general solver of PDEs needs to be implemented. Here, we used Finite Differences which can be replaced with more advance solvers. Clearly, the proposed adjoint method is most effective when there is a prior knowledge of the underlying PDE form, and an appropriate numerical solver is deployed. We note that an inherent limitation of the proposed adjoint method is the possibility of encountering either ill-posed forward or backward equations during optimization, which limits the time step size.
2. In this work, we used the same spatial discretization as the input data. If the spatial grid of input data is too coarse for the PDE solver, one has to use interpolation to estimate the data on a finer spatial grid that is more appropriate for the PDE solver.
3. In this work, we made the assumption that the underlying PDE can be solved numerically. This can be a limitation when there are no stable numerical methods to solve the true PDE. In this scenario, the proposed method may find another PDE that is solvable and fits to the data with a notable error.
4. Similar to PDE-FIND and similar symbolic regression methods, the Adjoint method considers a library of symbolic terms for the PDE. This can be a limitation when no prior information about the underlying dynamics is available.

5 Conclusion

In this work, we introduce a novel mathematical method for the discovery of partial differential equations given data using the adjoint method. By formulating the optimization problem in the variational form using the method of Lagrange multipliers, we find an analytic expression for the gradient of the cost function with respect to the parameters of the PDE as a function of the Lagrange multipliers and the forward model estimate. Then, using variational calculus, we find a backwards-in-time evolution equation for the Lagrange multipliers which incorporates the error with a source term (the adjoint equation). Hence, we can use the same solver for both forward model and the backward Lagrange equations. Here we used Finite Differences to estimate the spatial derivatives and forward Euler for the time derivatives, which indeed can be replaced with more stable and advanced solvers.

We compared the proposed adjoint method against PDE-FIND in several test cases. While PDE-FIND seems to be faster for small size problems, we observe that the adjoint method equipped with forward/backward solvers becomes faster than PDE-FIND as the size of the data set increases. Also, the adjoint method can provide machine-accuracy in identifying and finding the coefficients when the data set is noise-free. Furthermore, in the case of discovering PDEs for PDFs given its samples, both methods seem to suffer enormously form noise/bias associated with the finite number of samples and the Finite Difference on histogram. This motivates the use of smooth and least biased density estimator in these methods such as Tohme et al. (2023) in future work. In the future work, we intend to combine the adjoint-based method for the discovery of PDE with PINNs as the solver instead of Finite Difference method. This would allow us to handle noisy and sparse data as well as deploying larger time steps in estimating the forward and backward solvers.

Acknowledgements

T. Tohme was supported by the MathWorks Engineering Fellowship. M. Sadr thanks Dr. Andreas Adelmann for his support and Prof. Thomas Antonsen for his constructive discussion on the adjoint method. This research was also supported by the Center for Complex Engineering Systems at the Massachusetts Institute of Technology (MIT) and the King Abdulaziz City for Science and Technology (KACST).

References

E Paulo Alves and Frederico Fiuza. Data-driven discovery of reduced plasma physics models from fully kinetic simulations. *Physical Review Research*, 4(3):033192, 2022.

Thomas Antonsen, Elizabeth J Paul, and Matt Landreman. Adjoint approach to calculating shape gradients for three-dimensional magnetic confinement equilibria. *Journal of Plasma Physics*, 85(2):905850207, 2019.

Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

Josh Bongard and Hod Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

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.

Russel Caflisch, Denis Silantyev, and Yunan Yang. Adjoint dsmc for nonlinear boltzmann equation constrained optimization. *Journal of Computational Physics*, 439:110404, 2021.

Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature communications*, 12(1):6136, 2021.

Bryan C Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature communications*, 6(1):8133, 2015a.

Bryan C Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015b.

Junming Duan and Jan S Hesthaven. Non-intrusive data-driven reduced-order modeling for time-dependent parametrized problems. *Journal of Computational Physics*, 497:112621, 2024.

Sølve Eidnes and Kjetil Olsen Lye. Pseudo-hamiltonian neural networks for learning partial differential equations. *Journal of Computational Physics*, page 112738, 2024.

H Pearl Flath, Lucas C Wilcox, Volkan Akçelik, Judith Hill, Bart van Bloemen Waanders, and Omar Ghattas. Fast algorithms for bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial hessian approximations. *SIAM Journal on Scientific Computing*, 33(1):407–432, 2011.

Alessandro Geraldini, Matt Landreman, and Elizabeth Paul. An adjoint method for determining the sensitivity of island size to magnetic field variations. *Journal of Plasma Physics*, 87(3):905870302, 2021.

Dimitrios Giannakis and Andrew J Majda. Nonlinear laplacian spectral analysis for time series with intermittency and low-frequency variability. *Proceedings of the National Academy of Sciences*, 109(7): 2222–2227, 2012.

Mariano Giaquinta and Stefan Hildebrandt. *Calculus of variations I*, volume 310. Springer Berlin, Heidelberg, 2004.

Raul González-García, Ramiro Rico-Martìnez, and Ioannis G Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & chemical engineering*, 22:S965–S968, 1998.

Thomas JR Hughes, Gonzalo R Feijóo, Luca Mazzei, and Jean-Baptiste Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer methods in applied mechanics and engineering*, 166(1-2):3–24, 1998.

Antony Jameson. Aerodynamic shape optimization using the adjoint method. *Lectures at the Von Karman Institute, Brussels*, 2003.

Kadierdan Kaheman, J Nathan Kutz, and Steven L Brunton. Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A*, 476(2242):20200279, 2020.

Ioannis G Kevrekidis, C William Gear, James M Hyman, Panagiotis G Kevrekidis, Olof Runborg, Constantinos Theodoropoulos, et al. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun. Math. Sci*, 1(4):715–762, 2003.

John H Lagergren, John T Nardini, G Michael Lavigne, Erica M Rutter, and Kevin B Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A*, 476(2234):20190800, 2020.

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International conference on machine learning*, pages 3208–3216. PMLR, 2018.

Zichao Long, Yiping Lu, and Bin Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

Andrew J Majda, Christian Franzke, and Daan Crommelin. Normal forms for reduced stochastic climate models. *Proceedings of the National Academy of Sciences*, 106(10):3649–3653, 2009.

Niall M Mangan, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, 2016.

Takashi Matsubara, Ai Ishikawa, and Takaharu Yaguchi. Deep energy-based modeling of discrete-time physics. *Advances in Neural Information Processing Systems*, 33:13100–13111, 2020.

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

Igor Mezić. Analysis of fluid flows via spectral properties of the koopman operator. *Annual review of fluid mechanics*, 45:357–378, 2013.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

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.

Anthony John Roberts. *Model emergent dynamics in complex systems*, volume 20. SIAM, 2014.

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.

Nihar Sawant, Boris Kramer, and Benjamin Peherstorfer. Physics-informed regularization and structure preservation for learning stable reduced models from data with operator inference. *Computer Methods in Applied Mechanics and Engineering*, 404:115836, 2023.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324 (5923):81–85, 2009.

Michael D Schmidt, Ravishankar R Vallabhajosyula, Jerry W Jenkins, Jonathan E Hood, Abhishek S Soni, John P Wikswo, and Hod Lipson. Automated refinement and inference of analytical models for metabolic networks. *Physical biology*, 8(5):055011, 2011.

Robert Stephany and Christopher Earls. Pde-learn: Using deep learning to discover partial differential equations from noisy, limited data. *Neural Networks*, 174:106242, 2024.

George Sugihara, Robert May, Hao Ye, Chih-hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting causality in complex ecosystems. *science*, 338(6106):496–500, 2012.

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

Tony Tohme, Dehong Liu, and Kamal Youcef-Toumi. GSR: A generalized symbolic regression approach. *Transactions on Machine Learning Research*, 2022.

Tony Tohme, Mohsen Sadr, Kamal Youcef-Toumi, and Nicolas G Hadjiconstantinou. Messy estimation: Maximum-entropy based stochastic and symbolic density estimation. *arXiv preprint arXiv:2306.04120*, 2023.

Henning U Voss, Paul Kolodner, Markus Abel, and Jürgen Kurths. Amplitude equations from spatiotemporal binary-fluid convection data. *Physical review letters*, 83(17):3422, 1999.

Hao Ye, Richard J Beamish, Sarah M Glaser, Sue CH Grant, Chih-hao Hsieh, Laura J Richards, Jon T Schnute, and George Sugihara. Equation-free mechanistic ecosystem forecasting using empirical dynamic modeling. *Proceedings of the National Academy of Sciences*, 112(13):E1569–E1576, 2015.

A Derivation of the Adjoint equation

In this section, we provide a detailed derivation of the adjoint equations presented in this paper. Let us consider the forward model

$$\partial_t f_i + \sum_{\boldsymbol{d},\boldsymbol{p}} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[f^{\boldsymbol{p}}] = 0 \tag{32}$$

for $i = 1, ..., N$. In order to find the parameters $\alpha_{i,\boldsymbol{d},\boldsymbol{p}}$ of the such model, we use method of Lagrange multipliers to formulate a cost functional for the function $\boldsymbol{f}$ that has the minimum error from the data points $\boldsymbol{f}^*$ with the constraint that $\boldsymbol{f}$ also solves the forward model. For simplicity, let us consider the cost functional only within each time interval $[t^{(j)}, t^{(j+1)}]$, i.e.

$$\mathcal{C}[\boldsymbol{f}] = \sum_{i=1}^{N} \left(\underbrace{\sum_k (f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}))^2}_{I} + \underbrace{\int \lambda_i(\boldsymbol{x}, t) \mathcal{L}_i[\boldsymbol{f}(\boldsymbol{x}, t)] d\boldsymbol{x} dt}_{J} \right) + \epsilon_0 ||\boldsymbol{\alpha}||_2^2 \ . \tag{33}$$

Assuming the solution $\boldsymbol{f}$ and the Lagrange multipliers $\boldsymbol{\lambda}$ are sufficiently smooth, we derive functional derivatives, more precisely Gateaux derivatives Giaquinta and Hildebrandt (2004), of $\mathcal{C}$ with respect to $\boldsymbol{f}$ within the time interval $[t^{(j)}, t^{(j+1)}]$. We perform these operations first for the term denoted by I,

$$\begin{aligned}
\delta I[\boldsymbol{f}] &= \lim_{\epsilon \to 0} (\frac{d}{d\epsilon} I[\boldsymbol{f} + \epsilon \delta f_i \boldsymbol{e}_i]) && (34)\\
&= \lim_{\epsilon \to 0} (\frac{d}{d\epsilon} \sum_k (f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) - \epsilon \delta f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}))^2) && (35)\\
&= \lim_{\epsilon \to 0} (\sum_k -2\delta f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) - \epsilon \delta f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}))) && (36)\\
&= \sum_k -2\delta f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})) && (37)
\end{aligned}$$

and the second term denoted by J,

$$\begin{aligned}
\delta J[\boldsymbol{f}] &= \lim_{\epsilon \to 0} (\frac{d}{d\epsilon} J[\boldsymbol{f} + \epsilon \delta f_i \boldsymbol{e}_i]) && (38)\\
&= \lim_{\epsilon \to 0} (\frac{d}{d\epsilon} \int \lambda_i \mathcal{L}_i[\boldsymbol{f} + \epsilon \delta f_i \boldsymbol{e}_i] d\boldsymbol{x} dt) && (39)\\
&= \lim_{\epsilon \to 0} (\frac{d}{d\epsilon} \int \lambda_i \partial_t [f_i + \epsilon \delta f_i] + \lambda_i \sum_{\boldsymbol{d},\boldsymbol{p}} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})} [f_1^{p_1} f_2^{p_1} ... (f_i + \epsilon \delta f_i)^{p_i} ... f_N^{p_N}] d\boldsymbol{x} dt) && (40)\\
&= \lim_{\epsilon \to 0} (\frac{d}{d\epsilon} \int -\partial_t [\lambda_i](f_i + \epsilon \delta f_i) + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})} [\lambda_i] f_1^{p_1} f_2^{p_1} ... (f_i + \epsilon \delta f_i)^{p_i} ... f_N^{p_N} d\boldsymbol{x} dt \\
&\quad + \frac{d}{d\epsilon} (\lambda_i (f_i + \epsilon \delta f_i)) \Big|_{t^{(j)}}^{t^{(j+1)}}) && (41)\\
&= \lim_{\epsilon \to 0} (\int -\partial_t [\lambda_i] \delta f_i + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \ p_i \delta f_i f_1^{p_1} f_2^{p_1} ... (f_i + \epsilon \delta f_i)^{p_i - 1} ... f_N^{p_N} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})} [\lambda_i] d\boldsymbol{x} dt) \\
&\quad + (\lambda_i \delta f_i) \Big|_{t^{(j)}}^{t^{(j+1)}}) && (42)\\
&= \int \delta f_i \left(-\partial_t [\lambda_i] + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \ p_i \frac{f^{\boldsymbol{p}}}{f_i} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})} [\lambda_i] \right) d\boldsymbol{x} dt + (\lambda_i \delta f_i) \Big|_{t^{(j)}}^{t^{(j+1)}} && (43)
\end{aligned}$$

$$= \int \delta f_i \left(-\partial_t[\lambda_i] + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] \right) d\boldsymbol{x}dt + \lambda_i(\boldsymbol{x}, t^{(j+1)}) \delta f_i(\boldsymbol{x}, t^{(j+1)}) \ . \tag{44}$$

Here, $\boldsymbol{e}_i$ denotes the standard basis vector in the direction of ith-dimension. In this derivation, we used integration by parts, the divergence theorem, and considered compact support for $\boldsymbol{\lambda}$ in the spatial solution domain Ω, i.e. $\boldsymbol{\lambda} \to 0$ on the boundaries $\partial\Omega$. Since we assume that the solution at the initial time is given, there is no variation of $\mathcal{C}$ at the beginning of the time interval, that is, $\delta f_i(\boldsymbol{x}, t^{(j)}) = 0$. Hence, the total variation of the cost functional with respect to the function $\boldsymbol{f}$ for the time interval $t \in (t^{(j)}, t^{(j+1)}]$ becomes

$$\begin{aligned} \delta\mathcal{C}[\boldsymbol{f}] = \sum_{i=1}^{N} \Bigg(& - \sum_k 2(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})) \delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}} \\ & + \int (-\frac{\partial \lambda_i}{\partial t} + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i]) \delta f_i d\boldsymbol{x}dt \\ & + \sum_k \lambda_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) \delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}} \Bigg). \end{aligned} \tag{45}$$

Now that have found the total variation of the cost functional $\mathcal{C}$ with respect to $\boldsymbol{f}$, we can find the adjoint equation by considering only variation with respect to $\boldsymbol{f}$ in $t \in (t^{(j)}, t^{(j+1)})$ and ignoring $\delta f_{i,\boldsymbol{x}^{(k)},t^{(j+1)}}$, leading to

$$\delta\mathcal{C}[\boldsymbol{f}] \Big|_{\boldsymbol{f}(\boldsymbol{x},t^{(j+1)})} = 0 \tag{46}$$

$$\implies \int \left(-\frac{\partial \lambda_i}{\partial t} + \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] \right) \delta f_i d\boldsymbol{x}dt = 0 \tag{47}$$

$$\implies \frac{\partial \lambda_i}{\partial t} = \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] \ . \tag{48}$$

This equation is called adjoint equation presented in equation 6. Similarly, we can find the final condition for the adjoint equation by only considering variation of $\boldsymbol{f}$ at final time $t = t^{(j+1)}$ and ignoring δf_i which we used to denote the variation of f_i for $t \in (t^{(j)}, t^{(j+1)})$, leading to

$$\delta\mathcal{C}[\boldsymbol{f}] \Big|_{\boldsymbol{f}(\boldsymbol{x},t) \ \forall t \in (t^{(j)},t^{(j+1)})} = 0 \implies \lambda_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) = 2(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)})) \tag{49}$$

for $i = 1, ..., N$ and $j = 0, ..., N_t - 1$.

B Derivation of the Adjoint gradient with respect to PDE parameters

Consider the cost functional

$$\mathcal{C}[\boldsymbol{f}] = \sum_{i=1}^{N} \left(\sum_k (f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}))^2 + \int \lambda_i(\boldsymbol{x}, t) \mathcal{L}_i[\boldsymbol{f}(\boldsymbol{x}, t)] d\boldsymbol{x}dt \right) + \epsilon_0 ||\boldsymbol{\alpha}||_2^2 \ . \tag{50}$$

where

$$\mathcal{L}_i[\boldsymbol{f}] := \partial_t f_i + \sum_{\boldsymbol{d},\boldsymbol{p}} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[f^{\boldsymbol{p}}] \ . \tag{51}$$

Assume that we have solved the adjoint equation equation 6 and found the Lagrange multipliers $\boldsymbol{\lambda}$ for each time interval $[t^{(j)}, t^{(j+1)}]$. Next, we can find the gradient of $\mathcal{C}$ with respect to PDE parameters simply by

$$\frac{\partial \mathcal{C}}{\partial \alpha_{\boldsymbol{i},\boldsymbol{d},\boldsymbol{p}}} = \int \lambda_i \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[f^{\boldsymbol{p}}] d\boldsymbol{x}dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}}. \tag{52}$$

Using integration by parts, Divergence theorem, and the fact that $\boldsymbol{\lambda}$ has a compact support on the boundaries, we obtain

$$\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} = \int \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i f^{\boldsymbol{p}}] d\boldsymbol{x}dt + (-1)^{|\boldsymbol{d}|} \int f^{\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] d\boldsymbol{x}dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \tag{53}$$

$$= (-1)^{|\boldsymbol{d}|} \int f^{\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] d\boldsymbol{x}dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \ . \tag{54}$$

C Error in the numerical estimate of the Adjoint gradient

Consider the second-order central finite difference scheme as spatial and the first-order Euler as the temporal discretization scheme for the forward equation 1 and backward equations equation 6. Let us denote the discretized approximation of the solution with $\hat{f}$ and $\hat{\lambda}$, leading to

$$f = \hat{f} + \mathcal{O}(h^2) + \mathcal{O}(\Delta t) \tag{55}$$

$$\lambda = \hat{\lambda} + \mathcal{O}(h^2) + \mathcal{O}(\Delta t) \tag{56}$$

where h is the spatial spacing and Δt is the time step size. By plugging the discretization into equation 4, and using the same second-order numerical integration scheme in $\boldsymbol{x}$ and the first-order scheme in t, it can be shown that

$$\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} = (-1)^{|\boldsymbol{d}|} \int (\hat{f} + \mathcal{O}(h^2) + \mathcal{O}(\Delta t))^{\boldsymbol{p}} \, (\widehat{\nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i]} + \mathcal{O}(h^2) + \mathcal{O}(\Delta t)) d\boldsymbol{x}dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \tag{57}$$

$$\approx (-1)^{|\boldsymbol{d}|} \int \hat{f}^{\boldsymbol{p}} \, \widehat{\nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i]} d\boldsymbol{x}dt + \mathcal{O}(h^2) + \mathcal{O}(\Delta t) + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \tag{58}$$

$$\approx \widehat{\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}}} + \mathcal{O}(h^2) + \mathcal{O}(\Delta t) \tag{59}$$

where $\widehat{\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}}}$ denotes the discretized estimate of the gradient of cost function. Therefore, the adjoint estimate used in this work is second order in $\boldsymbol{x}$ and first order in t. Clearly, this may be improved using stable higher-order scheme. Interestingly, the accuracy of the adjoint gradient is only a function of mesh spacing and time step size, and not number of data points.

D Justification for the choice of the learning rate

In the proposed adjoint method, we considered the update rule

$$\alpha_{i,\boldsymbol{d},\boldsymbol{p}} \leftarrow \alpha_{i,\boldsymbol{d},\boldsymbol{p}} - \eta \frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} \tag{60}$$

for $i = 1, \ldots, N$. Here, we give a justification for our choice of the learning parameter η.

From the expression for the gradient of cost function with respect to parameters 4, i.e.

$$\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} = (-1)^{|\boldsymbol{d}|} \int f^{\boldsymbol{p}} \, \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] d\boldsymbol{x} dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}}, \tag{61}$$

we can see that

$$\begin{aligned} \left|\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}}\right| &= \mathcal{O}(\nabla_{\boldsymbol{x}}^{(\boldsymbol{d})} d\boldsymbol{x}) && (62) \\ &\leq \mathcal{O}(h^{-|\boldsymbol{d}|+1}) && (63) \end{aligned}$$

where $h = \min(\Delta \boldsymbol{x})$. So, the magnitude of the gradient scales exponentially with the order of the derivative d. The highest order terms, i.e. the terms with $d = d_{\max} = \max(|\boldsymbol{d}|)$, have the largest magnitude for their gradients. This means that by taking a constant learning rate η, the adjoint method would find the coefficients of the highest order terms first. This effect leads to the non-uniform convergence of the adjoint method.

In order to enforce uniform convergence on all PDE parameters, in this paper we consider

$$\eta = \beta \min(\Delta \boldsymbol{x})^{|\boldsymbol{d}| - d_{\max}} \tag{64}$$

as the learning rate which encodes the scaling with respect to the order of derivative for each PDE term. With this choice of learning rate, we have

$$\begin{aligned} \eta \left|\frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}}\right| &\leq \mathcal{O}(h^{|\boldsymbol{d}| - d_{\max}}) \mathcal{O}(h^{-|\boldsymbol{d}|+1}) && (65) \\ &\leq \mathcal{O}(h^{-d_{\max}+1}) \ , && (66) \end{aligned}$$

for all $i, \boldsymbol{d}, \boldsymbol{p}$. Hence, our choice of η, i.e. Eq. 64, enforces uniform convergence on all PDE parameters.

E Further Details on Adjoint method

Here, we present flowcharts to illustrate the proposed adjoint algorithms 1-2 with and without averaging the gradients in Fig. 18. We also provide instructions to help practitioners to deploy the Adjoint method for their data set

- **Diverging Coefficients.** There are two scenarios in which the learning of coefficients in the Adjoint method can become divergent. First, the intermediate guessed PDE may become so unstable that numerical solution to either forward or backward adjoint equations diverges even for short time intervals $t \in [t^{(j+1)}, t^{(j)}]$. Second, the time step size from the dataset $t^{(j+1)} - t^{(j)}$ is too large for the underlying numerical solver, leading to diverging coefficients. In order to control the diverging coefficients during training, one should reduce the time step size for the forward and backward Adjoint equations which are independent of time step size in the dataset. Second, in these scenarios, we recommend deploying smaller learning rate η.
- **Denoise Before Train.** If the input dataset is noisy, we recommend deploying a denoising method before considering the PDE discovery task. In this work, we considered SVD and drop out components with singular values below a threshold of $\mathcal{O}(10^{-4})$ to smooth out the noise. Clearly, this is problem dependent. Since the noise to signal ratio is not known in advance, the commonly used denoising approaches introduce bias in the discovered PDE, regardless of deployed PDE discovery method.

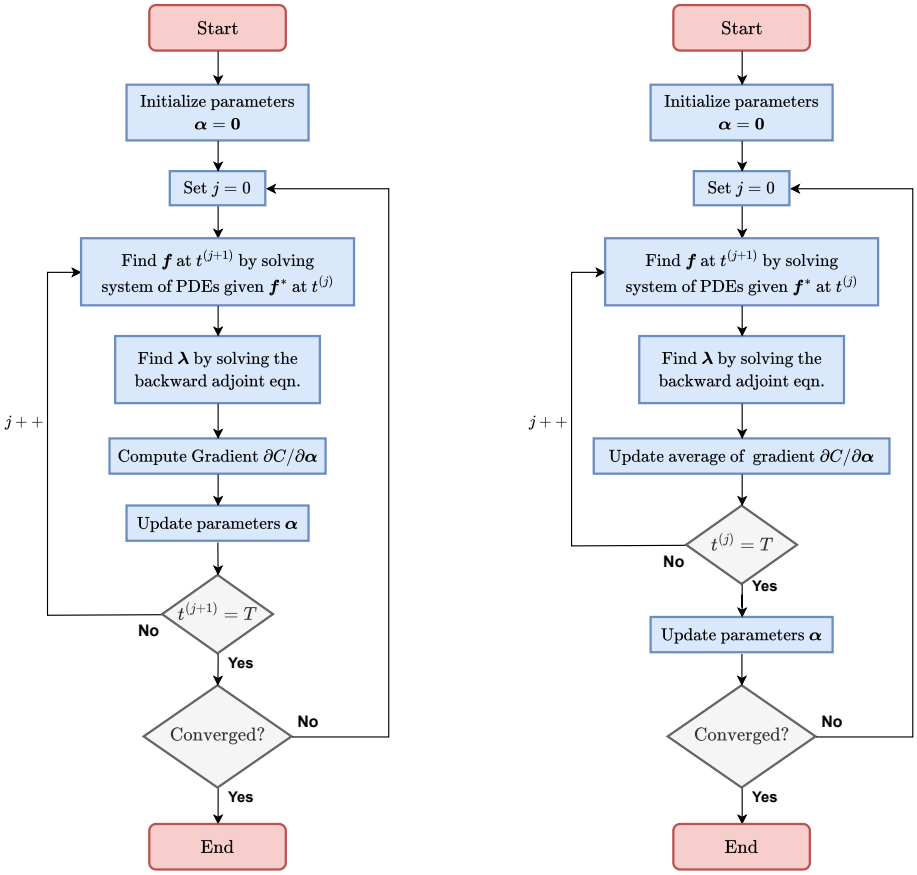


Figure 18: Training flowchart of the Adjoint method in finding PDEs (left) without and (right) with gradient averaging.

F Impact of hyperparameters on adjoint method

In this section, we study the impact of some of the hyperparameters used in the adjoint algorithm. We repeat the PDE discovery experiment for Burgers's and Kuramoto Sivashinsky equation with data on a grid with $N_x = N_t = 100$, as described in 3.1.2 and 3.1.3. We check the error in the outcome coefficients and the solution of estimated forward model compared to the data.

As shown in Fig. 19, by increasing the regularization factor ϵ_0, the optimization problem seems to converge faster to a stationary solution. In case of Burgers' equation, we considered $\epsilon_0 \in \{10^{-4}, 10^{-8}, 10^{-12}, 10^{-16}\}$, where for all values of ϵ_0 the exact solution is recovered. However, in the case of Kuramoto Sivashinsky with $\epsilon_0 \in \{10^{-10}, 10^{-12}, 10^{-14}, 10^{-16}\}$, the solution seems to be more sensitive to ϵ_0. Here, we fix the other hyperparameters $\gamma_{\text{thr}} = 10^{-16}$ and $\beta = 2 \times 10^{-3}$ for the Burgers's equation and $\beta = 20$ for Kuramoto Sivashinsky equation. We observe that high regularization factor deteriorates the accuracy, while stabilizing the regression problem.

Next, we investigate how the error changes with the thresholding tolerance where $\gamma_{\text{thr}} \in \{10^{-4}, 10^{-8}, 10^{-12}, 10^{-16}\}$. Here, we fix the other hyperparameters $\epsilon_0 = 10^{-16}$ and $\beta = 2 \times 10^{-3}$ for the Burgers's equation and $\beta = 20$ for Kuramoto Sivashinsky equation. Although using smaller γ_{thr} allows faster convergence to a stationary solution almost in all cases, we remind the reader that γ_{thr} should be large enough to allow enough training of the coefficients before truncating terms. In other words, the user should avoid trivial scenarios where the initial guesses for coefficients $\boldsymbol{\alpha}$ are zero and the thresholding is applied from the very beginning of the training.

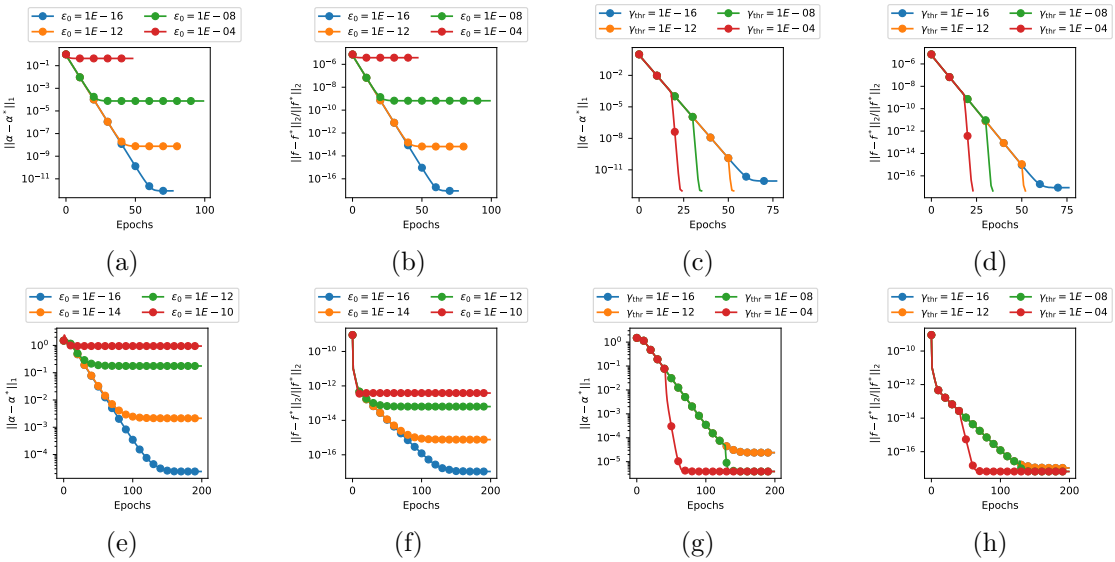
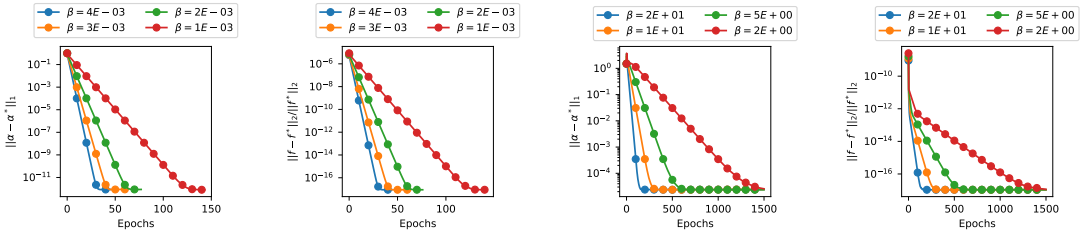


Figure 19: Impact of regularization factor ϵ_0 and thresholding tolerance γ_{thr} on the error of adjoint method for Burgers' equation (a-d) and Kuramoto Sivashinsky equation (e-h).

Finally, we show the impact of the free parameter β in the learning rate on the resulting PDE discovered by the adjoint method. We compared the solution of adjoint method using $\beta \in \{10^{-3}, 2 \times 10^{-3}, 3 \times 10^{-3}, 4 \times 10^{-3}\}$ for the Burgers' equation, and $\beta \in \{2, 5, 10, 20\}$ for the Kuramoto Sivashinsky equation. Also, we fix the other hyperparameters $\gamma_{\text{thr}} = \epsilon_0 = 10^{-16}$. As shown in Fig. 20, regardless of the value of β, adjoint method delivers the same solution. However, larger values of β lead to faster convergence to the solution, if the numerical solver does not become unstable. The upper bound of β is limited by the stability of the guessed PDE, and can be found with try-and-error.



(a) (b) (c) (d)

Figure 20: Impact of the free parameter β in the learning rate on the error of adjoint method for Burgers' equation (a-b) and Kuramoto Sivashinsky equation (c-d).

G Illustration of deployed notation for the considered cases.

Although the proposed method and its algorithm can be and has been computed in an automated fashion, here we show two detailed illustrative examples for 1-dimensional and 2-dimensional cases presented in Section 3 for the sake of better understanding the used notation and how the library of candidate terms looks like.

G.1 Heat and Burgers' Equations

As mentioned in Sections 3.1.1 and 3.1.2, for these two cases, we consider a system consisting of a single PDE, i.e. $N = \dim(\boldsymbol{f}) = \dim(\boldsymbol{p}) = 1$ where $\boldsymbol{f} = f$ and $\boldsymbol{p} = p$, in a one-dimensional input space, i.e. $n = \dim(\boldsymbol{x}) = \dim(\boldsymbol{d}) = 1$ where $\boldsymbol{x} = x$, and $\boldsymbol{d} = d$. In addition, we consider candidate terms consisting of derivatives with indices $d \in \{1, 2, 3\}$ and polynomials with indices $p \in \{1, 2, 3\}$). In other words, $d_{\max} = 3$ and $p_{\max} = 3$. The resulting forward model in Eq. 1 takes the form

$$\begin{aligned}\mathcal{L}[f] &= \frac{\partial f}{\partial t} + \sum_{d=1}^{3}\sum_{p=1}^{3} \alpha_{d,p} \frac{\partial^d (f^p)}{\partial x^d} \\ &= \frac{\partial f}{\partial t} + \alpha_{1,1}\frac{\partial f}{\partial x} + \alpha_{1,2}\frac{\partial (f^2)}{\partial x} + \alpha_{1,3}\frac{\partial (f^3)}{\partial x} \\ &\quad + \alpha_{2,1}\frac{\partial^2 f}{\partial x^2} + \alpha_{2,2}\frac{\partial^2 (f^2)}{\partial x^2} + \alpha_{2,3}\frac{\partial^2 (f^3)}{\partial x^2} \\ &\quad + \alpha_{3,1}\frac{\partial^3 f}{\partial x^3} + \alpha_{3,2}\frac{\partial^3 (f^2)}{\partial x^3} + \alpha_{3,3}\frac{\partial^3 (f^3)}{\partial x^3}\end{aligned} \tag{67}$$

where $\alpha_{d,p}$ denotes the parameter corresponding to the term with d-th derivative and p-th polynomial order. As we can observe, we have 9 terms with unknown coefficients $\boldsymbol{\alpha} = [\alpha_{d,p}]_{d\in\{1,2,3\},p\in\{1,2,3\}}$ that we aim to find using the proposed adjoint method.

The cost functional in this case is simply

$$\mathcal{C} = \sum_{j,k} \left(f^*\left(x^{(k)}, t^{(j)}\right) - f\left(x^{(k)}, t^{(j)}\right)\right)^2 + \int \lambda(x,t)\mathcal{L}[f(x,t)]dxdt + \epsilon_0 ||\boldsymbol{\alpha}||_2^2 \ . \tag{68}$$

Letting variational derivatives of $\mathcal{C}$ with respect to f to be zero, and using integration by parts, the corresponding adjoint equation can be obtained as

$$\frac{\partial \lambda}{\partial t} = \sum_{d=1}^{3}\sum_{p=1}^{3} (-1)^d \alpha_{d,p} \frac{\partial (f^p)}{\partial f} \frac{\partial^d \lambda}{\partial x^d}$$

$$
\begin{aligned}
&= -\alpha_{1,1}\frac{\partial\lambda}{\partial x} - \alpha_{1,2}\big(2f\big)\frac{\partial\lambda}{\partial x} - \alpha_{1,3}\big(3f^2\big)\frac{\partial\lambda}{\partial x} \\
&\quad + \alpha_{2,1}\frac{\partial^2\lambda}{\partial x^2} + \alpha_{2,2}\big(2f\big)\frac{\partial^2\lambda}{\partial x^2} + \alpha_{2,3}\big(3f^2\big)\frac{\partial^2\lambda}{\partial x^2} \\
&\quad - \alpha_{3,1}\frac{\partial^3\lambda}{\partial x^3} - \alpha_{3,2}\big(2f\big)\frac{\partial^3\lambda}{\partial x^3} - \alpha_{3,3}\big(3f^2\big)\frac{\partial^3\lambda}{\partial x^3}
\end{aligned} \tag{69}
$$

with final condition $\lambda(x^{(k)}, t^{(j+1)}) = 2(f^*(x^{(k)}, t^{(j+1)}) - f(x^{(k)}, t^{(j+1)}))$ for all j, k. The parameters $\boldsymbol{\alpha}$ are then found using the gradient descent method with update rule

$$
\alpha_{d,p} \leftarrow \alpha_{d,p} - \eta\,\frac{\partial\mathcal{C}}{\partial\alpha_{d,p}} \tag{70}
$$

$$
\text{where} \quad \eta = \beta\min(\Delta x)^{d-d_{\max}} \quad \text{and} \quad \frac{\partial\mathcal{C}}{\partial\alpha_{d,p}} = (-1)^d\int f^p\,\frac{\partial^d\lambda}{\partial x^d}dxdt + 2\epsilon_0\alpha_{d,p}\ . \tag{71}
$$

This leads to the update rule for each coefficient, for example

$$
\begin{aligned}
\alpha_{1,1} &\leftarrow \alpha_{1,1} - \frac{\beta}{\min(\Delta x)^2}\int f\,\frac{\partial\lambda}{\partial x}dxdt - 2\beta\epsilon_0\alpha_{1,1} \\
\alpha_{1,2} &\leftarrow \alpha_{1,2} - \frac{\beta}{\min(\Delta x)^2}\int f^2\,\frac{\partial\lambda}{\partial x}dxdt - 2\beta\epsilon_0\alpha_{1,2} \\
\alpha_{1,3} &\leftarrow \alpha_{1,3} - \frac{\beta}{\min(\Delta x)^2}\int f^3\,\frac{\partial\lambda}{\partial x}dxdt - 2\beta\epsilon_0\alpha_{1,3}\ .
\end{aligned}
$$

G.2 Reaction Diffusion System of Equations

As mentioned in Section 3.1.5, for this case, we consider a system consisting of two PDEs, i.e. $N = \dim(\boldsymbol{f}) = \dim(\boldsymbol{p}) = 2$ where $\boldsymbol{f} = [f_1, f_2]$ and $\boldsymbol{p} = [p_1, p_2]$, in a two-dimensional input space, i.e. $n = \dim(\boldsymbol{x}) = \dim(\boldsymbol{d}) = 2$ where $\boldsymbol{x} = [x_1, x_2]$, and $\boldsymbol{d} = [d_1, d_2]$. In addition, we consider candidate terms with derivatives such that $\boldsymbol{d} \in \mathcal{D}_{\boldsymbol{d}} = \{[0,0], [1,0], [0,1], [2,0], [0,2]\}$ and polynomials such that $\boldsymbol{p} \in \mathcal{D}_{\boldsymbol{p}} = \{[1,0], [0,1], [1,1], [2,0], [0,2], [2,1], [1,2], [3,0], [0,3]\}$. In other words, $d_{\max} = 2$ and $p_{\max} = 3$. The resulting forward model in Eq. 1 takes the form

$$
\mathcal{L}_i[\boldsymbol{f}] = \partial_t f_i + \sum_{\boldsymbol{d},\boldsymbol{p}} \alpha_{i,\boldsymbol{d},\boldsymbol{p}}\nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[f^{\boldsymbol{p}}] \tag{72}
$$

where $i \in \{1, 2\}$, $f^{\boldsymbol{p}} = f_1^{p_1} f_2^{p_2}$ and $\nabla_{\boldsymbol{x}}^{(\boldsymbol{d})} = \nabla_{x_1}^{(d_1)}\nabla_{x_2}^{(d_2)}$. This is equivalent to

$$
\begin{aligned}
\mathcal{L}_i[f_1, f_2] &= \frac{\partial f_i}{\partial t} + \sum_{[d_1,d_2]\in\mathcal{D}_{\boldsymbol{d}}}\ \sum_{[p_1,p_2]\in\mathcal{D}_{\boldsymbol{p}}} \alpha_{i,[d_1,d_2],[p_1,p_2]}\frac{\partial^{\,d_1+d_2}\big(f_1^{p_1}f_2^{p_2}\big)}{\partial x_1^{d_1}\partial x_2^{d_2}} \\
&= \frac{\partial f_i}{\partial t} + \alpha_{i,[0,0],[1,0]}f_1 + \alpha_{i,[0,0],[0,1]}f_2 + \alpha_{i,[0,0],[1,1]}f_1f_2 + \ldots + \alpha_{i,[0,0],[0,3]}f_2^3 \\
&\quad + \alpha_{i,[1,0],[1,0]}\frac{\partial f_1}{\partial x_1} + \alpha_{i,[1,0],[0,1]}\frac{\partial f_2}{\partial x_1} + \alpha_{i,[1,0],[1,1]}\frac{\partial\big(f_1f_2\big)}{\partial x_1} + \ldots + \alpha_{i,[1,0],[0,3]}\frac{\partial\big(f_2^3\big)}{\partial x_1} \\
&\quad + \qquad \ldots \\
&\quad + \alpha_{i,[0,2],[1,0]}\frac{\partial^2 f_1}{\partial x_2^2} + \alpha_{i,[0,2],[0,1]}\frac{\partial^2 f_2}{\partial x_2^2} + \alpha_{i,[0,2],[1,1]}\frac{\partial^2\big(f_1f_2\big)}{\partial x_2^2} + \ldots + \alpha_{i,[0,2],[0,3]}\frac{\partial^2\big(f_2^3\big)}{\partial x_2^2}
\end{aligned} \tag{73}
$$

where $i \in \{1, 2\}$. As we can observe, we have $|\mathcal{D}_{\boldsymbol{d}}| \times |\mathcal{D}_{\boldsymbol{p}}| = 5 \times 9 = 45$ terms with unknown coefficients $\boldsymbol{\alpha}_i = [\alpha_{i,\boldsymbol{d},\boldsymbol{p}}]_{d\in\mathcal{D}_{\boldsymbol{d}},p\in\mathcal{D}_{\boldsymbol{p}}}$ for the i-th PDE, i.e. a total of 90 terms for the considered system, that we aim to find using the proposed adjoint method.

The cost functional in this case is simply

$$\mathcal{C} = \sum_{i=1}^{2} \left(\sum_{j,k} \left(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) \right)^2 + \int \lambda_i(\boldsymbol{x}, t) \mathcal{L}_i[\boldsymbol{f}(\boldsymbol{x}, t)] d\boldsymbol{x} dt \right) + \epsilon_0 ||\boldsymbol{\alpha}||_2^2 \ . \tag{74}$$

The corresponding adjoint equation is given by

$$\begin{aligned} \frac{\partial \lambda_i}{\partial t} &= \sum_{\boldsymbol{d},\boldsymbol{p}} (-1)^{|\boldsymbol{d}|} \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \nabla_{f_i}[f^{\boldsymbol{p}}] \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] \\ &= \sum_{[d_1,d_2]\in\mathcal{D}_{\boldsymbol{d}}} \sum_{[p_1,p_2]\in\mathcal{D}_{\boldsymbol{p}}} (-1)^{d_1+d_2} \alpha_{i,[d_1,d_2],[p_1,p_2]} \frac{\partial \left(f_1^{p_1} f_2^{p_2} \right)}{\partial f_i} \frac{\partial^{d_1+d_2} \lambda_i}{\partial x_1^{d_1} \partial x_2^{d_2}} \end{aligned} \tag{75}$$

and $\lambda_i(\boldsymbol{x}^{(k)}, t^{(j+1)}) = 2(f_i^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_i(\boldsymbol{x}^{(k)}, t^{(j+1)}))$ for all j, k and where $i \in \{1, 2\}$.

Assume, without loss of generality, that $i = 1$. Then, we can write

$$\begin{aligned} \frac{\partial \lambda_1}{\partial t} = &+ \alpha_{1,[0,0],[1,0]} \lambda_1 + \alpha_{1,[0,0],[1,1]} f_2 \lambda_1 + \alpha_{1,[0,0],[2,0]} \left(2 f_1\right) \lambda_1 + \ldots + \alpha_{1,[0,0],[3,0]} \left(3 f_1^2\right) \lambda_1 \\ &- \alpha_{1,[1,0],[1,0]} \frac{\partial \lambda_1}{\partial x_1} - \alpha_{1,[1,0],[1,1]} f_2 \frac{\partial \lambda_1}{\partial x_1} - \alpha_{1,[1,0],[2,0]} \left(2 f_1\right) \frac{\partial \lambda_1}{\partial x_1} - \ldots - \alpha_{1,[1,0],[3,0]} \left(3 f_1^2\right) \frac{\partial \lambda_1}{\partial x_1} \\ &+ \quad \ldots \\ &+ \alpha_{1,[0,2],[1,0]} \frac{\partial^2 \lambda_1}{\partial x_2^2} + \alpha_{1,[0,2],[1,1]} f_2 \frac{\partial^2 \lambda_1}{\partial x_2^2} + \alpha_{1,[0,2],[2,0]} \left(2 f_1\right) \frac{\partial^2 \lambda_1}{\partial x_2^2} + \ldots + \alpha_{1,[0,2],[3,0]} \left(3 f_1^2\right) \frac{\partial^2 \lambda_1}{\partial x_2^2} \end{aligned} \tag{76}$$

and $\lambda_1(\boldsymbol{x}^{(k)}, t^{(j+1)}) = 2(f_1^*(\boldsymbol{x}^{(k)}, t^{(j+1)}) - f_1(\boldsymbol{x}^{(k)}, t^{(j+1)}))$ for all j, k. We can follow the same procedure for $i = 2$. The parameters $\boldsymbol{\alpha}_i$ are then found using the gradient descent method with update rule

$$\alpha_{i,\boldsymbol{d},\boldsymbol{p}} \leftarrow \alpha_{i,\boldsymbol{d},\boldsymbol{p}} - \eta \frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} \tag{77}$$

where

$$\eta = \beta \min(\Delta \boldsymbol{x})^{|\boldsymbol{d}| - d_{\max}} \quad \text{and} \quad \frac{\partial \mathcal{C}}{\partial \alpha_{i,\boldsymbol{d},\boldsymbol{p}}} = (-1)^{|\boldsymbol{d}|} \int f^{\boldsymbol{p}} \nabla_{\boldsymbol{x}}^{(\boldsymbol{d})}[\lambda_i] d\boldsymbol{x} dt + 2\epsilon_0 \alpha_{i,\boldsymbol{d},\boldsymbol{p}} \tag{78}$$

with $\Delta \boldsymbol{x} = \Delta x_1 \Delta x_2$, leading to the update rule for each coefficient, for example

$$\begin{aligned} \alpha_{i,[0,0],[1,0]} &\leftarrow \alpha_{i,[0,0],[1,0]} - \frac{\beta}{\min(\Delta \boldsymbol{x})^2} \int f_1 \lambda_i d\boldsymbol{x} dt - 2\beta\epsilon_0 \alpha_{i,[0,0],[1,0]} \\ \alpha_{i,[1,0],[1,0]} &\leftarrow \alpha_{i,[1,0],[1,0]} - \frac{\beta}{\min(\Delta \boldsymbol{x})} \int f_1 \frac{\partial \lambda_i}{\partial x_1} d\boldsymbol{x} dt - 2\beta\epsilon_0 \alpha_{i,[1,0],[1,0]} \ . \end{aligned}$$

H Setup for WSINDy

In this section, we detail the parameters used in the discovery of PDEs using WSINDy method in this manuscript following default values from Messenger and Bortz (2021) as the reference for the reader.

Table 12: The configuration of WSINDy hyperparameters used in this work

PDE	max. derivative	max. power	(m_t, m_x)	(s_t, s_x)
Burgers' Eq.	3	3	$(60, 60)$	$(5, 5)$
Heat Eq.	3	3	$(60, 60)$	$(5, 5)$
KS	4	2	$(22, 23)$	$(5, 5)$
Random Walk	3	1	$(60, 60)$	$(5, 5)$