Automating the discovery of partial differential equations in dynamical systems

IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 035046 https://doi.org/10.1088/2632-2153/ad682f

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
3 May 2024

REVISED
14 June 2024

ACCEPTED FOR PUBLICATION
26 July 2024

PUBLISHED
14 August 2024

PAPER

Automating the discovery of partial differential equations in dynamical systems

Weizhen Li[1] and **Rui Carvalho**[1,2,*]

[1] Department of Engineering, Durham University, Lower Mountjoy, South Road, Durham DH1 3LE, United Kingdom
[2] Institute for Data Science, Durham University, Lower Mountjoy, South Road, Durham DH1 3LE, United Kingdom
[*] Author to whom any correspondence should be addressed.

E-mail: rui.carvalho@durham.ac.uk

Keywords: system identification, machine learning, sparse regression, partial differential equations, nonlinear dynamics

Abstract

Identifying partial differential equations (PDEs) from data is crucial for understanding the governing mechanisms of natural phenomena, yet it remains a challenging task. We present an extension to the ARGOS framework, ARGOS-RAL, which leverages sparse regression with the recurrent adaptive lasso to identify PDEs from limited prior knowledge automatically. Our method automates calculating partial derivatives, constructing a candidate library, and estimating a sparse model. We rigorously evaluate the performance of ARGOS-RAL in identifying canonical PDEs under various noise levels and sample sizes, demonstrating its robustness in handling noisy and non-uniformly distributed data. We also test the algorithm's performance on datasets consisting solely of random noise to simulate scenarios with severely compromised data quality. Our results show that ARGOS-RAL effectively and reliably identifies the underlying PDEs from data, outperforming the sequential threshold ridge regression method in most cases. We highlight the potential of combining statistical methods, machine learning, and dynamical systems theory to automatically discover governing equations from collected data, streamlining the scientific modeling process.

1. Introduction

In recent years, scientists have increasingly employed statistical and machine learning methods to uncover the governing equations of dynamical systems, particularly differential equations, from observational data [1–7]. Data-driven methods offer several advantages over traditional approaches that rely on first principles and expert knowledge [8, 9]. These methods can reveal patterns and relationships in the data that may not be apparent from first principles, providing new insights into complex systems [10–14]. They are also adept at working with noisy or incomplete data commonly encountered in real-world applications, employing techniques from machine learning to enhance the robustness of discoveries [15–18]. Furthermore, by reducing the need for manual intervention and domain expertise, data-driven methods can significantly streamline the discovery process [19].

Data-driven discovery in dynamical systems has evolved from early parameter estimation using spline approximation and system reconstruction [20, 21], to leveraging statistical methods such as least squares [22–24], mixed-effects models [25, 26], and Bayesian approaches [2, 11, 14, 27] for parameter estimation in ordinary and partial differential equations (ODEs and PDEs). The advent of high-performance computing has further propelled symbolic regression to find differential operators and approximate the response function of the underlying equations, enabling the discovery of governing equations from data in physics and engineering [1, 8, 28–30]. A notable development in this field is the Sparse Identification of Nonlinear Dynamics (SINDy) approach [3, 5], which constructs an extensive library of potential terms and employs the Sequential Threshold Ridge Regression (STRidge) algorithm [5] to select significant terms iteratively.

SINDy and its various enhancements [31–38] have been extensively used to discover a broad spectrum of ODEs and PDEs, describing diverse phenomena such as fluid mechanics [39], turbulence models [40], aerodynamics [41], and biological and chemical systems [42, 43]. Recent developments have combined neural network-based techniques and SINDy, leading to innovative approaches that enhance noise tolerance in identifying PDEs [6, 30, 44–48]. Neural networks can learn complex nonlinear relationships and effectively filter out noise, complementing SINDy's ability to identify parsimonious models. Methods, such as deep learning for model discovery (DeepMoD) [9] and physics-informed neural networks with sparse regression (PINN-SR) [45], combine physical information, neural networks, sparse regression (the lasso in DeepMoD and STRidge in PINN-SR) and hard thresholding to find the governing equations from data. However, both neural network and SINDy methods require specific hyperparameter tuning, such as setting regularization parameters or choosing network architectures. For example, STRidge requires setting a threshold to select active terms from the candidate library [5, 37, 45, 47]. Additionally, SINDy-based methods typically approximate numerical derivatives from noisy data using the Savitzky–Golay filter, a technique for smoothing data by fitting local low-degree polynomials [49]. The parameters of this filter, such as the polynomial degree and window size, must be carefully tuned for optimal performance [5, 19]. Neural network approaches, on the other hand, require detailed decisions regarding their architecture and functioning, such as the number of neurons, the structure of hidden layers, the types of activation and loss functions, and the learning rate. In particular, using physics-informed neural networks [6, 45, 47] requires a prior understanding of the equation terms, as well as initial and boundary conditions. Consequently, using neural networks and SINDy-based methods presents a trade-off: the absence of fully automated algorithms requires users to engage in manual tuning and iterative usage of semi-automated algorithms. This scenario highlights a key challenge in the field: developing an automated algorithm to identify PDEs with minimal manual intervention, streamlining the process, and improving its applicability across diverse scientific domains.

To address the challenge of parameter tuning, Egan *et al* [19] proposed the Automatic Regression for Governing Equations (ARGOS) algorithm, which identifies ODEs by automating the parameter tuning process. ARGOS assumes the underlying system is unknown, automates the fine-tuning of parameters for numerical differentiation, and leverages sparse regression with bootstrap confidence intervals to select active terms from the candidate library. To automatically identify PDEs, we develop ARGOS with the Recurrent Adaptive Lasso (ARGOS-RAL). This extension of the ARGOS framework employs only sparse regression to identify equations rather than engaging in large-scale bootstrapping.

We evaluate the performance of the ARGOS-RAL algorithm through a series of three numerical tests, each designed to assess its ability to identify canonical PDEs across diverse fields, including biology, neuroscience, earth science, fluid mechanics, and quantum mechanics. The first test explores the algorithm's resilience against varying noise levels by altering the signal-to-noise ratio (SNR) in Gaussian random noise integrated into the PDE solutions, which is crucial for understanding the robustness of ARGOS-RAL under realistic noisy conditions. The second test addresses the practical challenges encountered in real-world data collection, which often results in non-uniformly distributed data points in space and time, by exploring the minimum percentage of data points necessary for the algorithm to accurately identify the underlying equation. The final evaluation assesses the algorithm's ability to process datasets characterized by significant noise, challenging its limits and practical applicability in scenarios where data quality is compromised. Our results demonstrate that ARGOS-RAL can effectively and reliably identify the underlying PDEs from data, outperforming the STRidge method used in SINDy.

2. Methods

2.1. Overview of the ARGOS-RAL framework

The general form of a homogeneous PDE is

$$u_t + F(x, t, u, u_x, u_{xx}, \ldots) = 0 \tag{1}$$

where $F(\cdot)$ governs the behavior of the system, with $u = u(x, t)$ denoting its state. The notation $u_t, u_x, u_{xx}, \cdots$ represents the partial derivatives of u with respect to time and space, respectively. Equation (1) serves as a foundational representation of the dynamical system, encapsulating a wide range of phenomena through its generalized form, which can be adapted to include multiple spatial dimensions or to model systems without explicit time dependence.

To focus on data-driven modeling of spatiotemporal dynamical systems, we incorporate empirical data directly into the modeling process:

$$\mathbf{U}_t = \frac{\partial \mathbf{U}}{\partial t} = \mathbf{F}\left(x, \mathbf{U}, \mathbf{U}_x, \mathbf{U}_{xx}, \dots\right), \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{n\times m}$ is a matrix representing the solution of the PDE as a function of x and t, and $\mathbf{F}(\cdot)$ denotes the unknown mapping inferred from the collected data, which contains linear and nonlinear operators.

We aim to estimate the unknown mapping $\mathbf{F}(\cdot)$ with sparse regression by constructing a comprehensive library of potential terms and assuming that only a few of them are active [3, 5, 19]. To cover a broad spectrum of possible influences on the dynamics of the system, this library includes a wide variety of functions, such as constants, monomials, interaction terms (products of variables), possibly trigonometric, and other functions, depending on the dynamical system being studied [5]. In the case of Burgers' equation, $u_t = -uu_x + 0.1u_{xx}$, the true dynamics involves only two terms: the nonlinear convection term uu_x and the linear diffusion term u_{xx}. When applying sparse regression to data generated from Burgers' equation, the method should ideally select only these two terms from the candidate library.

All features related to $\mathbf{U}(x,t)$ in equation (2) are matrices. Implementing this matrix data in sparse regression leads to the creation of m distinct regression models. Each model captures the spatial dynamics of the system at a specific time point t_j, where $j = 1, 2, \dots, m$. To consolidate the m regression models into a single linear regression problem, we reshape the matrix $\mathbf{U}(x,t)$ and its derivative matrices into vectors. These vectors then serve as predictors within the candidate library $\boldsymbol{\Theta}$, which can be represented in $\mathbb{R}^{(n\cdot m)\times p}$ or $\mathbb{C}^{(n\cdot m)\times p}$. By stacking the vectorized data and candidate terms, we can estimate a single sparse coefficient vector β that represents the governing equation across all time points rather than estimating separate models for each time point. Here, $\mathbf{U} \in \mathbb{R}^{n\times m}$ is represented in matrix form as

$$\mathbf{U}(x,t) = \begin{pmatrix} u(x_1,t_1) & u(x_1,t_2) & \cdots & u(x_1,t_m) \\ u(x_2,t_1) & u(x_2,t_2) & \cdots & u(x_2,t_m) \\ \vdots & \vdots & \ddots & \vdots \\ u(x_n,t_1) & u(x_n,t_2) & \cdots & u(x_n,t_m) \end{pmatrix}. \tag{3}$$

Vectorizing equation (3) yields:

$$\begin{aligned} \mathbf{u} &= \mathrm{vec}\left(\mathbf{U}\right) \\ &= \begin{pmatrix} u(x_1,t_1) & \cdots & u(x_n,t_1) & \cdots & u(x_1,t_m) & \cdots & u(x_n,t_m) \end{pmatrix}^T. \end{aligned} \tag{4}$$

Similarly, $\mathbf{u}_t = \mathrm{vec}(\mathbf{U}_t) = \mathrm{vec}(\partial\mathbf{U}/\partial t)$, $\mathbf{u}_x = \mathrm{vec}(\mathbf{U}_x) = \mathrm{vec}(\partial\mathbf{U}/\partial x)$, $\mathbf{u}_{xx} = \mathrm{vec}(\mathbf{U}_{xx}) = \mathrm{vec}(\partial^2\mathbf{U}/\partial x^2)$, $\mathbf{u}^2 = \mathrm{vec}(\mathbf{U}\odot\mathbf{U})$, and $\mathbf{u}\mathbf{u}_x = \mathrm{vec}(\mathbf{U}\odot\mathbf{U}_x)$, where $\odot$ denotes the Hadamard product. The design matrix is given by

$$\boldsymbol{\Theta}\left(\mathbf{u}\right) = \begin{pmatrix} | & | & & | & & | & | & & | & & | & \\ \mathbf{1} & \mathbf{u} & \cdots & \mathbf{u}^d & \cdots & \mathbf{u}_x & \mathbf{u}_{xx} & \cdots & \mathbf{u}\mathbf{u}_x & \cdots & \mathbf{u}^d\mathbf{u}_{xx} & \cdots \\ | & | & & | & & | & | & & | & & | & \end{pmatrix}, \tag{5}$$

where $\mathbf{u}^d$ is a vector where all elements denote a dth degree monomial. For example, if our data $\mathbf{U}(x,t)$ is on a 200×100 grid (i.e. 200 spatial measurements and 100 time-steps) and the candidate library has 30 terms, then $\boldsymbol{\Theta} \in \mathbb{R}^{20000\times 30}$.

After vectorization, we estimate $\mathbf{F}(\cdot)$ by transforming equation (2) to a linear regression model

$$\mathbf{u}_t = \boldsymbol{\Theta}\left(\mathbf{u}\right)\beta + \epsilon, \tag{6}$$

where $\beta \in \mathbb{R}^p$ is a sparse coefficient vector in which only a few values are nonzero, and ϵ is the vector of residuals.

2.2. Automated Numerical Differentiation using the Savitzky–Golay Filter and the Gaussian Blur

A crucial step in constructing the candidate library in equation (5) is the numerical calculation of derivatives, see figures 1(a) and (b). The Savitzky–Golay filter [49] has become a favored solution in system identification for signal smoothing and differentiation [5, 50]. This method applies a least squares polynomial fit over a sliding window of data points, thereby achieving simultaneous signal smoothing and differentiation. The selection of the Savitzky–Golay filter is grounded in its proven ability to accurately maintain the original contour of the signal while significantly reducing noise and to approximate higher-order numerical derivatives with symbolic differentiation [51].

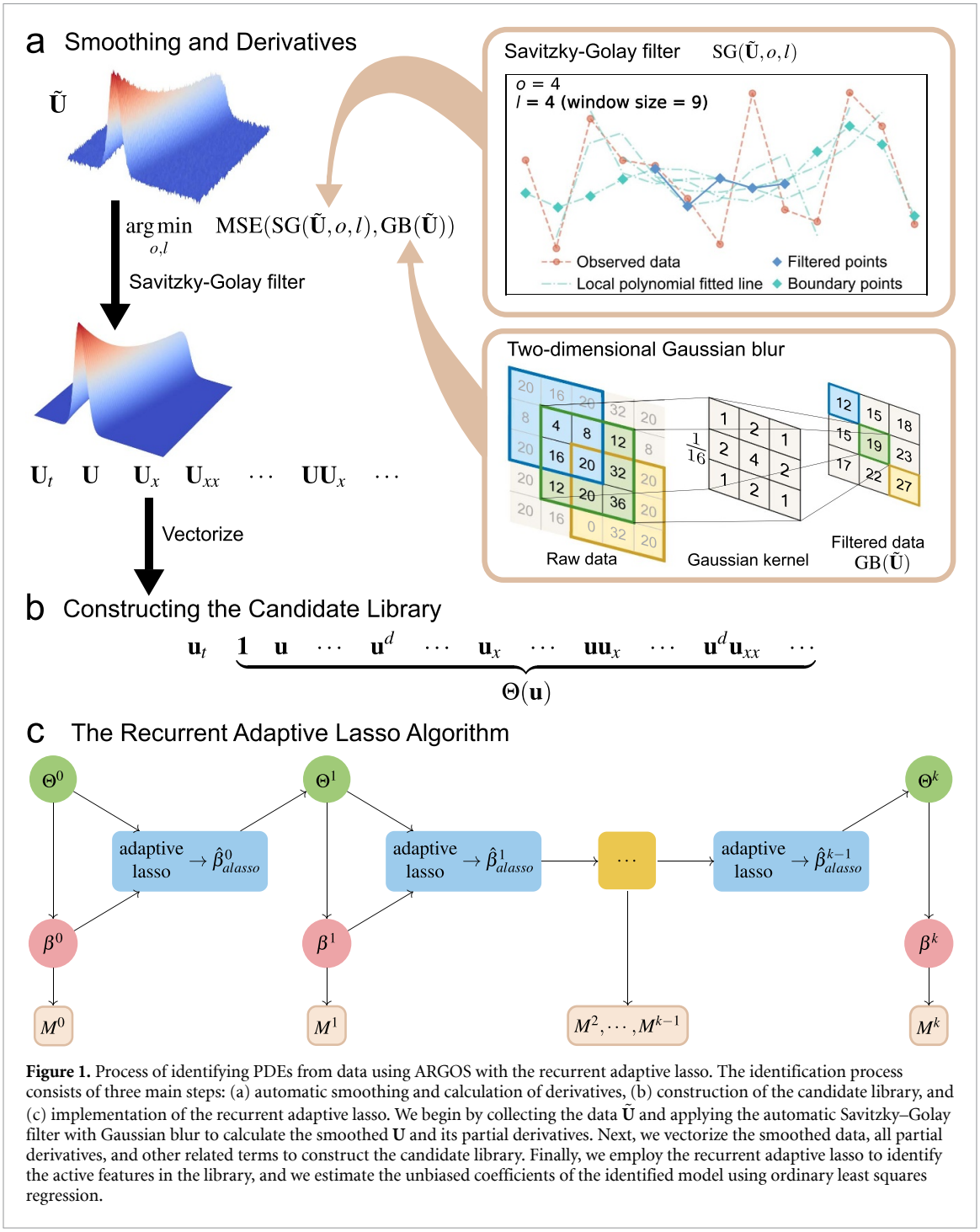

Figure 1. Process of identifying PDEs from data using ARGOS with the recurrent adaptive lasso. The identification process consists of three main steps: (a) automatic smoothing and calculation of derivatives, (b) construction of the candidate library, and (c) implementation of the recurrent adaptive lasso. We begin by collecting the data $\tilde{\mathbf{U}}$ and applying the automatic Savitzky–Golay filter with Gaussian blur to calculate the smoothed $\mathbf{U}$ and its partial derivatives. Next, we vectorize the smoothed data, all partial derivatives, and other related terms to construct the candidate library. Finally, we employ the recurrent adaptive lasso to identify the active features in the library, and we estimate the unbiased coefficients of the identified model using ordinary least squares regression.

The Savitzky–Golay filter is characterized by two integer hyperparameters: the polynomial order o and the window length l, which are constrained by the conditions that o must be at least 2, l should be an odd number, and $o+1+\mathrm{mod}(o) \leqslant l \leqslant n-1$ [51]. To automate the selection of these hyperparameters, we first apply a Gaussian blur with the kernel $(1,2,1)$ to smooth the observational data (see A.1). We then treat this smoothed data, denoted as $\mathrm{GB}(\tilde{\mathbf{U}})$, as the ground truth. Next, we find the optimal set of hyperparameters $\{o^*, l^*\}$ by minimizing the mean squared error (MSE) between the Savitzky–Golay filtered data $\mathrm{SG}(\tilde{\mathbf{U}}, o, l)$ and the ground truth $\mathrm{GB}(\tilde{\mathbf{U}})$ (see algorithm 1). After finding the optimal set $\{o^*, l^*\}$, we use the Savitzky–Golay filter with these parameters to compute the smoothed data and its derivatives.

2.3. Sparse regression with the recurrent adaptive Lasso

The adaptive lasso is a two-step method [19, 52]. The first step uses the ordinary least squares (OLS) to obtain unbiased estimates and derive the weights w:

$$w = |\hat{\beta}_{\mathrm{ols}}|^{-\gamma}, \quad \gamma > 0 \tag{7}$$

Algorithm 1. Automatic Savitzky–Golay Filter.

Input: $\mathbf{U} \in \mathbb{R}^{n\times m}$ or $\mathbb{C}^{n\times m}$, dt, dx.
Output: partial derivatives $\mathbf{U}_t$, $\mathbf{U}_x$, $\mathbf{U}_{xx}$, $\cdots$.
$\mathbf{U}_{\text{GB}} = \text{Gaussian\_Blur}(\mathbf{U})$; `// use Gaussian blurred data as the ground truth;`
$(o_t^*, l_t^*) = \arg\min_{o,l} \text{MSE}\ (\text{Savitzky–Golay}(\tilde{\mathbf{U}}(t), o, l), \mathbf{U}_{\text{GB}})$;
$(o_x^*, l_x^*) = \arg\min_{o,l} \text{MSE}\ (\text{Savitzky–Golay}(\tilde{\mathbf{U}}(x), o, l), \mathbf{U}_{\text{GB}})$;
$\mathbf{U}_t = \text{Savitzky–Golay}(\mathbf{U}_{GB}, o_t^*, l_t^*, \text{derivative} = 1)$;
$\mathbf{U}_x = \text{Savitzky–Golay}(\mathbf{U}_{GB}, o_x^*, l_x^*, \text{derivative} = 1)$;
$\mathbf{U}_{xx} = \text{Savitzky–Golay}(\mathbf{U}_{GB}, o_x^*, l_x^*, \text{derivative} = 2)$;
$\vdots$

where $\hat{\beta}_{\text{ols}}$ is the OLS estimate, and γ is an exponent tuning the shape of the soft-thresholding function. In the second step, we obtain the estimated coefficients $\hat{\beta}_{\text{alasso}}$ using the glmnet package [53] in R by solving the problem

$$\hat{\beta}_{\text{alasso}} = \arg\min_{\beta} \|u_t - \boldsymbol{\Theta}\beta\|_2^2 + \lambda \sum_{j=1}^{p} w_j \left|\beta_j\right|, \tag{8}$$

where λ is a nonnegative regularization parameter controlling the amount of shrinkage applied to the coefficients of the predictors. Unlike the lasso, where the weight vector is $w = \mathbf{1}$, the adaptive lasso varies the weights in the regularization function, resulting in a stronger penalty on smaller coefficients, thus driving more of them to zero and leading to a sparser model compared to the standard lasso. The recurrent adaptive lasso applies the adaptive lasso repeatedly until convergence, resulting in a sparse model with fewer non-zero coefficients.

To balance the model's complexity against its accuracy, we employ the Pareto curve, which illustrates the optimal trade-off between the regularization penalty and the model residuals [54–56]. We identify the optimal λ as a point of high curvature on the Pareto curve, where slight improvements in one objective require significant sacrifices in the other [57]. On the Pareto curve (see figure 2), the x-axis represents the norm of the model residuals, $\|\mathbf{X}\beta - y\|_2$, while the y-axis represents the penalty, defined as $\|w^T\beta\|_1$ in the adaptive lasso. As shown in figure 2, the Pareto curve initially decreases steeply from the left (the lowest residual norm) as increasing λ allows the algorithm to estimate a sparser model with lower goodness-of-fit. The knee occurs at an intermediate value of λ that trades off between model complexity and goodness-of-fit. To the right of the knee, further increases in λ lead to very small reductions in model complexity at the cost of large increases in the residual norm, indicating underfitting. We use Cultrera and Callegaro's algorithm [56] to find the optimal point on the Pareto curve, represented by the red point in figure 2. Although cross-validation is an alternative method, Cortiella *et al* [34] have shown that it finds a λ optimized for prediction, potentially overfitting the true underlying equation with extra features.

The adaptive lasso regression often detects more terms than those in the true system. To improve parsimony, Egan *et al* [19] suggested combining the adaptive lasso with bootstrap techniques to identify ODEs. Similarly, Cortiella *et al* [34] adopted a modified version of the multi-step adaptive lasso [58] to develop a sparser model that more accurately identifies the true equations. This is achieved by iteratively adjusting the adaptive weights using previous estimates from the adaptive lasso. A significant advancement made by Cortiella *et al* [34] is their method's ability to maintain finite weights in the adaptive lasso equation by ensuring that the estimated coefficients shrink to a small, nonzero value rather than dropping to zero. However, this approximation unintentionally introduces numerical inaccuracies as a trade-off for preventing overflow during the equation identification process.

The recurrent adaptive lasso is an iterative algorithm that estimates an initial sparse model using the adaptive lasso and subsequently refining it by trimming the candidate library, see figure 1(c). At each iteration, it removes terms whose coefficients the adaptive lasso penalized to zero (see algorithm 2 step 9). It then employs least squares to re-estimate the coefficients of the remaining terms, which are used to update the adaptive weights in the next adaptive lasso iteration. This focuses the regularization on the terms that had small coefficients in the previous iteration. As this process repeats, the recurrent adaptive lasso increasingly concentrates the ℓ_1-norm shrinkage on terms that are likely irrelevant, driving their coefficients to

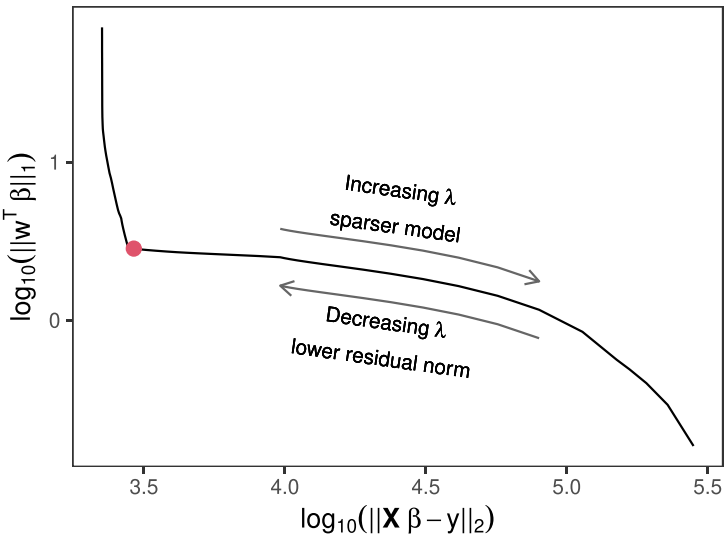

Figure 2. Pareto curve of the adaptive lasso for a sampled dataset from a Navier–Stokes system with an SNR of 36 dB. The Pareto curve balances the trade-off between sparsity and goodness-of-fit. The red point on the curve indicates the optimal value of the regularization parameter λ that achieves the best balance between these two competing objectives. Increasing λ leads to sparser solutions at the cost of a poorer fit to the data, while decreasing λ improves the fit but yields less sparse solutions.

Algorithm 2. The recurrent adaptive lasso with Pareto curve and AIC.

Iutput: $\Theta(u) \in \mathbb{R}^{(n\cdot m)\times p}$ or $\mathbb{C}^{(n\cdot m)\times p}$, $u_t \in \mathbb{R}^{(n\cdot m)\times 1}$ or $\mathbb{C}^{(n\cdot m)\times 1}$.
Output: $\hat{\beta}$

for γ in 1:5 **do**
  $\mathcal{J}^{(\gamma,0)} = \text{NULL}$; `// initialize` $\mathcal{J}$;
  $k = 1$; `// iteration counter;`
  $\mathcal{J}^{(\gamma,k)} = \{1, 2, \cdots, p\}$; `// selected columns from` Θ;
  while $\mathcal{J}^{(\gamma,k)} \neq \mathcal{J}^{(\gamma,k-1)}$ **do**
    $w^{(\gamma,k)} = \left(\arg\min_{\beta_{\mathcal{J}^{(\gamma,k)}}} \left\| u_t - \Theta(u)_{\mathcal{J}^{(\gamma,k)}} \beta_{\mathcal{J}^{(\gamma,k)}} \right\|_2^2\right)^{-\gamma}$; `// ols weights;`
    $\hat{\beta}^{(\gamma,k)} = \arg\min_{\beta_{\mathcal{J}^{(\gamma,k)}}} \left\| u_t - \Theta(u)_{\mathcal{J}^{(\gamma,k)}} \beta_{\mathcal{J}^{(\gamma,k)}} \right\|_2^2 + \lambda^* \sum_{j=1}^{p} w_j^{(\gamma,k)} |\beta_{j\cdot\mathcal{J}^{(\gamma,k)}}|$;
    `//` λ^* `is the optimal point on the Pareto curve;`
    $\mathcal{A}^{(\gamma,k)} = \text{AIC}(\hat{\beta}^{(\gamma,k)})$;
    $\mathcal{J}^{(\gamma,k)} = \left\{ j : \hat{\beta}_j^{(\gamma,k)} \neq 0 \right\}$; `// select active terms;`
    $k = k + 1$;
  end
end
$\mathcal{J}^* = \mathcal{J}^{(\gamma^*,k^*)}$ where (γ^*, k^*) is the index of the minimum $\mathcal{A}$;
$\hat{\beta} = \arg\min_{\beta_{\mathcal{J}^*}} \| u_t - \Theta(u)_{\mathcal{J}^*} \beta_{\mathcal{J}^*} \|_2^2$;

zero [52, 59]. Meanwhile, it relaxes the regularization on terms that consistently have larger coefficients, allowing the model to retain them. The candidate set gets smaller at each iteration until the algorithm converges on a sparse model containing only the key terms. This iterative re-weighting allows the recurrent adaptive lasso to prune irrelevant terms more aggressively than the standard adaptive lasso while retaining good predictive performance. The result is a parsimonious model that identifies the true governing equation more reliably, even in the presence of many extraneous candidate terms.

Increasing the number of iterations may cause the recurrent adaptive lasso to underestimate the model. This can lead to the omission of active terms that should be included in the true underlying equation. Therefore, while iterating the candidate library $\boldsymbol{\Theta}$, we record all candidate models and calculate the Akaike information criterion (AIC) for each model to determine the final governing equation corresponding to the lowest AIC. Given the uncertainty that the true model falls within all candidates, the AIC serves to select the model that best approximates the true model [60, 61].

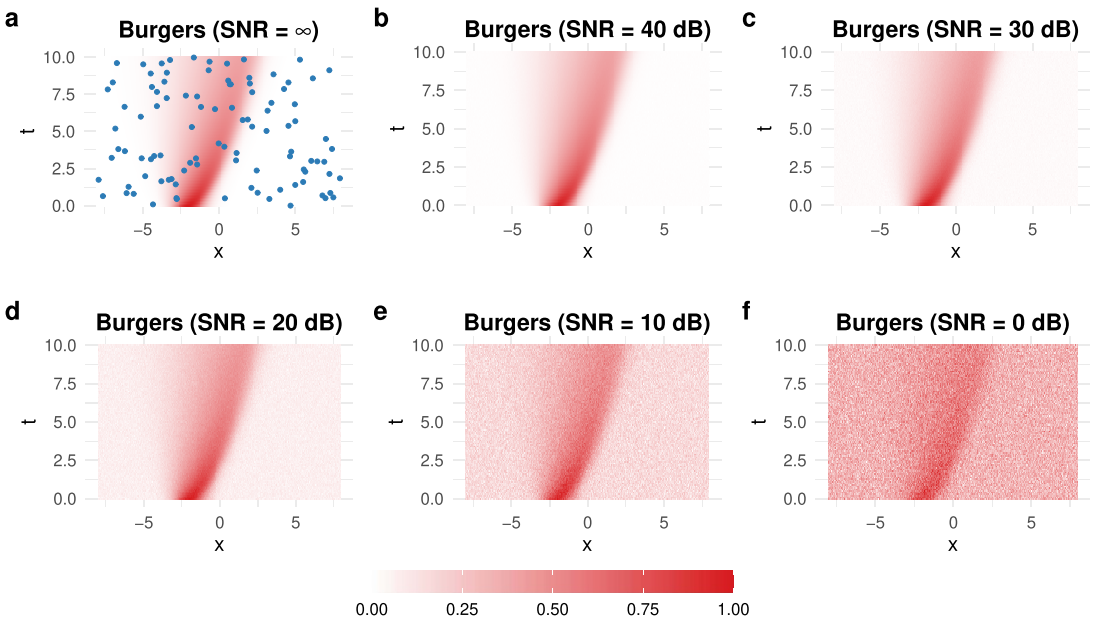

Figure 3. Influence of SNR on the Burgers' equation dataset. (a) Noiseless data points (blue) serve as a reference for evaluating the impact of sample size on PDE identification accuracy. (b)–(f) Noisy datasets are generated by adding Gaussian noise at SNR levels of 40 dB, 30 dB, 20 dB, 10 dB and 0 dB, respectively, to comprehensively characterize the system's behavior under varying noise conditions.

3. Results and discussion

3.1. Evaluating the performance of ARGOS-RAL under varying noise levels and sample sizes

We compare the performance of ARGOS-RAL and STRidge [5] in identifying ten canonical PDEs under various SNRs and sample sizes (N). We evaluate their performance on both noisy and noiseless data. Figure 3 demonstrates the impact of introducing increasing levels of Gaussian random noise into the solution of the Burgers' equation, effectively decreasing the SNR values. Adding Gaussian noise to simulated data is the most common way to mimic naturally occurring random processes [62–64].

In the evaluation of noise-contaminated data, we express the SNR as $\mathrm{SNR} = 20\log_{10}(\sigma_{\mathbf{U}}/\sigma_Z)$, where $\sigma_{\mathbf{U}}$ is the standard deviation of the original data, and σ_Z represents the standard deviation of the added noise. We systematically vary σ_Z to span a broad range of noise levels, facilitating a comprehensive evaluation of the efficacy of ARGOS-RAL and STRidge in identifying various PDEs under different noise conditions. For this purpose, we generate datasets with SNRs set at $\{0, 2, \cdots, 58, 60, \infty\}$ [19], each comprising paired elements $\{\mathbf{u}_t, \boldsymbol{\Theta}(\mathbf{u})\}$. This approach allows us to examine the robustness of each PDE identification method as it copes with varying noise levels.

In investigating sample size, N, our objective is to determine the smallest number of samples needed to reliably identify PDEs with a success rate exceeding 80%. To achieve this, we first generate a full dataset for each PDE by calculating partial derivatives and assembling a candidate library as described in equation (5). The size of the full dataset, denoted as $\mathbf{N}$, varies depending on the specific PDE under consideration. Specifically, $\mathbf{N} = 10^4$ for the advection-diffusion, Burgers, and cable equations, $\mathbf{N} = 10^5$ for the quantum harmonic oscillator (QHO), transport, Navier–Stokes, and reaction-diffusion equations, and $\mathbf{N} = 10^{4.8}$ for the diffusion and Korteweg–De Vries (KdV) equations. Next, we randomly sample smaller subsets of size N from the full dataset, where N is chosen from a logarithmically spaced grid: $N = 10^2, 10^{2.2}, 10^{2.4}, \cdots, \mathbf{N}$ [19], see the blue points in figure 3(a). By applying the PDE identification methods to these subsets and evaluating their success rates, we can determine the smallest sample size required for reliable identification of each PDE.

3.2. Quantifying success rates in identifying canonical PDEs

To evaluate the impact of different SNRs and data sizes on the method, we measure the uncertainty of model identification caused by random sampling. To do so, we create 100 unique datasets at each point on the grid, corresponding to different SNRs and N values. For each dataset, we quantify the identification accuracy with the success rate, $\eta = \#\mathrm{correct}/100$, where #correct represents the number of times the model correctly identifies all active terms. Our accuracy assessment ignores small differences between theoretical and empirical coefficients, such as a theoretical value of 0.1 compared to an estimated value of 0.098. Figure 4

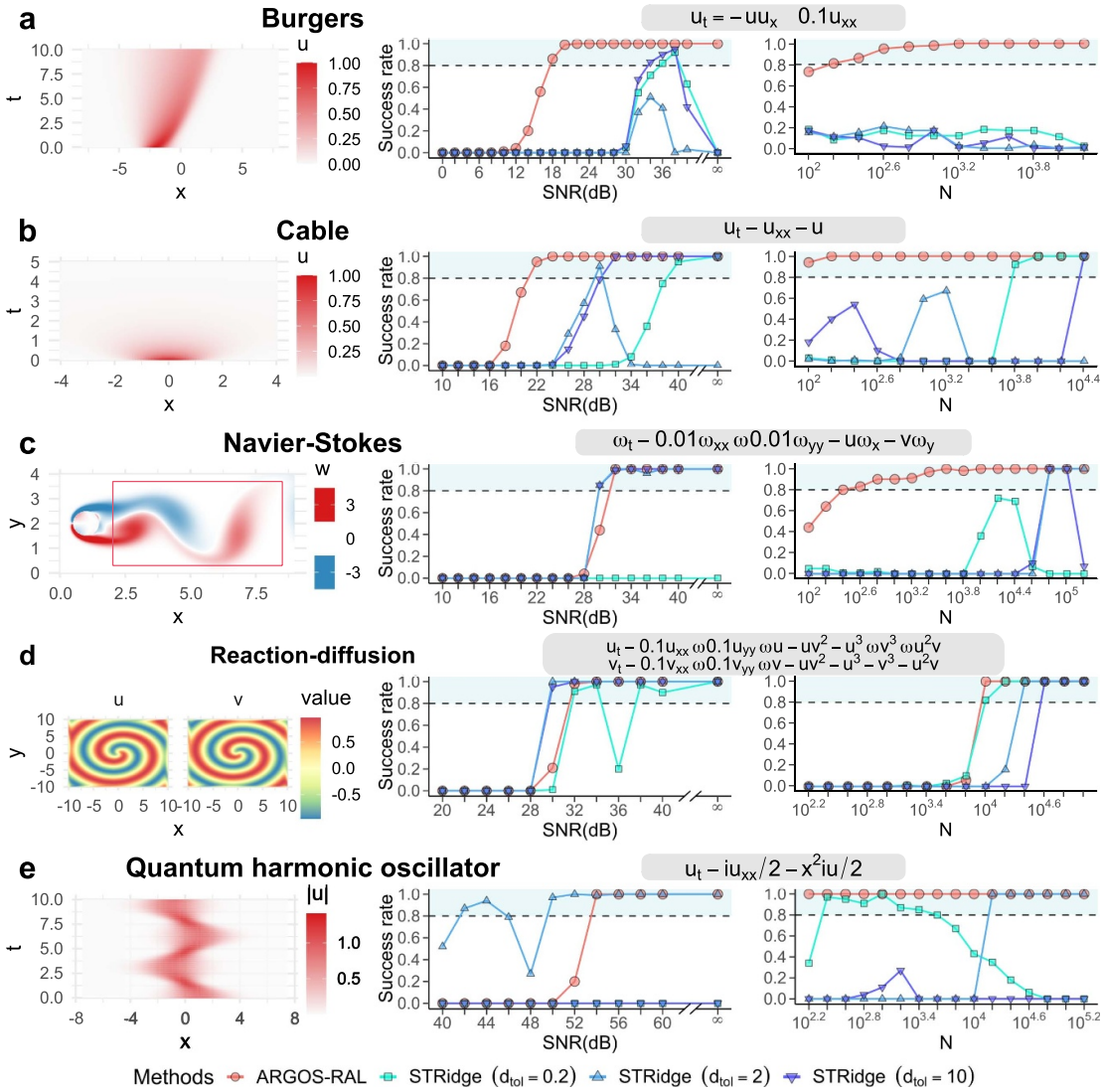

Figure 4. Success rates of ARGOS-RAL and STRidge in identifying (a) Burgers', (b) cable, (c) Navier–Stokes, (d) reaction-diffusion, and (e) quantum harmonic oscillator equations with varying SNRs and sample sizes. We analyze the noise tolerance by adding noise of different SNRs to the PDE solutions. For the sample size analysis, we randomly sample points from the set $\{\mathbf{u}_t, \boldsymbol{\Theta}(\mathbf{u})\}$ based on noiseless data. In panel (c), we use the region indicated by the red rectangle to implement both the SNR and sample size tests by sampling points within this area. PDE solution plots display time snapshots at $t = 306$ for Navier–Stokes in panel (c) and $t = 1$ for reaction-diffusion in panel (d). Lines connecting the points are used for visual guidance only and do not represent a fit to the data. Shaded regions represent model discovery accuracy above 80%.

illustrates these results for a selection of systems: the Burgers', Cable, Navier–Stokes, reaction-diffusion, and QHO models. We provide further analysis on additional PDEs—transport, diffusion, advection-diffusion, and KdV equations—in (A.2) figure A1.

ARGOS-RAL identifies Burgers', cable, Navier–Stokes, reaction-diffusion, and advection-diffusion equations, achieving a success rate of 100% when the SNR exceeds 30 dB (see figures 4 and A1). However, accurately detecting specific equations requires a high SNR, particularly for the QHO, KdV, transport, and diffusion equations. The KdV equation, which involves third-order partial derivatives, presents challenges due to the significant biases in numerical approximations of these derivatives [48], resulting in datasets unsuitable for system identification with sparse regression. To implement sparse regression within the real number domain for the complex number QHO PDE, we apply the transformation shown in A.2 equation (A.9). This transformation expands the design matrix $\boldsymbol{\Theta}$ from $nm \times p$ to $2nm \times 2p$, effectively quadrupling its size and potentially leading to high correlations between the variables in $\boldsymbol{\Theta}$. The transport and diffusion equations, containing only terms u_x and u_{xx} respectively, exhibit high correlation with their correlated terms in the library, such as $\{u_x, uu_x\}$ and $\{u_{xx}, uu_{xx}\}$, which hinders the effectiveness of ℓ_1-norm shrinkage regression in identifying correct terms [52].

Figures 4 and A1 illustrate that ARGOS-RAL achieves a higher success rate than STRidge in identifying PDEs with limited data points. ARGOS-RAL consistently identifies a significant number of PDEs using as

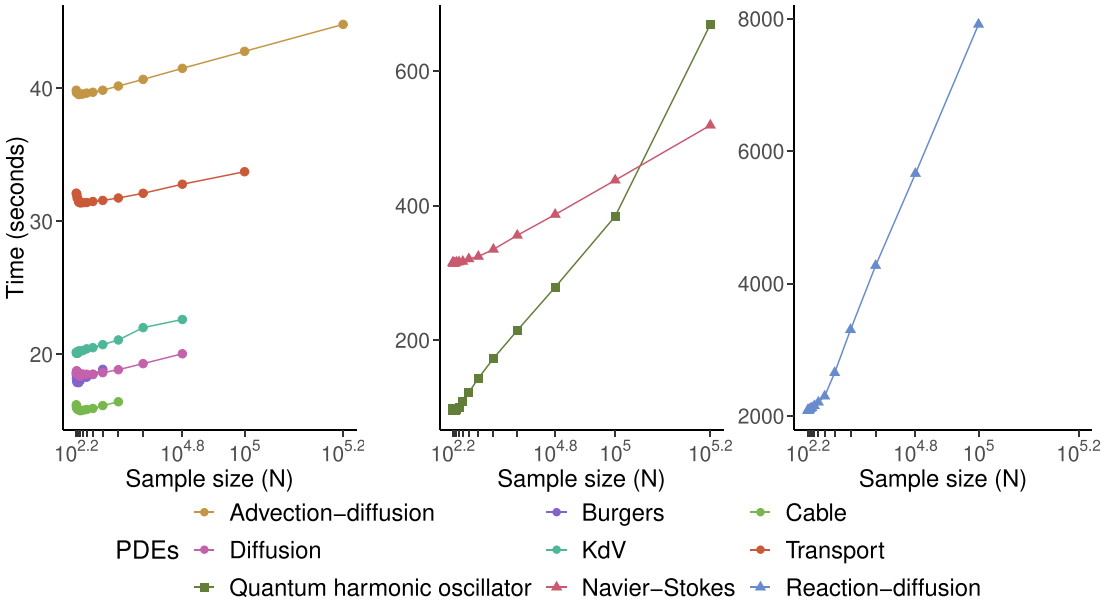

Figure 5. Computational costs of ARGOS-RAL as sample size (N) increases for noiseless PDE systems. We run all benchmarks using parallel processing and convert them to total computing time on a single core. We use circles and triangles for PDEs with one and two spatial dimensions and squares for the complex-valued quantum harmonic oscillator.

few as 1000 data points, maintaining a success rate above 80%. However, some equations, such as the reaction-diffusion and KdV equations, require larger sample sizes of approximately 10^4 and $10^{3.8}$ data points, respectively, for reliable identification. We thus demonstrate ARGOS-RAL as a consistent and efficient method for PDE identification with non-uniformly sampled and noiseless datasets.

ARGOS-RAL shows a remarkable ability to identify PDEs accurately and consistently across a wide range of SNRs and sample sizes. Its success rate improves as the SNR and sample size increase, reaching 100% when both values are sufficiently large. This trend highlights the robustness of ARGOS-RAL in handling various data conditions and underscores its effectiveness in identifying PDEs, even when faced with varying levels of data quality and quantity. However, in certain scenarios, STRidge [5] with specific d_{tol} thresholds exceeds the performance of ARGOS-RAL. For instance, STRidge achieves higher success rates in identifying Navier–Stokes and reaction-diffusion equations at a 30 dB SNR, using d_{tol} settings of 2 and 10, respectively, see figures 4(c) and (d). Moreover, STRidge with $d_{\text{tol}} = 2$ is more proficient in identifying the QHO and the transport equation with an SNR lower than 52 dB, see figures 4(e) and A1(c), respectively. These results from the SNR and N experiments reveal that using a single fixed threshold in STRidge can lead to performance variability depending on the input data, highlighting the difficulty of selecting an appropriate d_{tol} threshold without prior knowledge of the system. This variability underscores the sensitivity of STRidge to specific threshold settings, which can impact its consistency across different datasets. Overall, STRidge surpasses ARGOS-RAL in identifying simpler PDEs, such as the transport and diffusion equations; see figures A1(c) and (d).

3.3. Computational costs

We analyzed the computing time for all examined PDEs to demonstrate the computational cost of ARGOS-RAL. We performed all evaluations on a high-performance computing cluster equipped with 120 standard compute nodes, each featuring 128 CPU cores (2x AMD EPYC 7702) and 256 GB RAM (246 GB available to users). We then converted the multi-core processing time to a single-core

Figure 5 illustrates the computational costs of ARGOS for the selected PDEs as the data size (N) increases. Except for the Navier–Stokes and reaction-diffusion equations, the solution grid size is lower than $10^{5.2}$, causing some lines in the graph to terminate. The computing times for identifying PDEs increase linearly with N, except for the QHO, a PDE containing complex-valued terms whose candidate library is four times larger than those of real-coefficient PDEs (see figure 4(e) and equation (A.9) in A.2.5).

3.4. Robustness analysis using white Gaussian noise

To better understand the limits of identification algorithms, we designed an extreme test on a single spatial dimension. This test effectively creates a situation without valid data collection ($\sigma_U = 0$), equivalent to an SNR of negative infinity, representing a dataset entirely composed of random noise. This scenario sets the

Table 1. Models identified from random noise by ARGOS-RAL and STRidge. We construct the candidate library with monomials and derivatives of orders ranging from three to five. We define parsimonious models as having three or fewer nonzero coefficients. We evaluate each identified model with an F-test to determine statistical significance, with the number of significant models (p-value < 0.05) noted in parentheses. Numbers outside parentheses indicate the number of models that did not significantly differ from the null hypothesis according to the F-test. c_1, c_2, c_3 are constants. For the ordinary differential equation (ODE) models, the monomial order d is a positive integer, with the maximum order corresponding to the highest order in the candidate library.

Candidate library	Parsimonious model (%)				Non-parsimonious model (%)
	ODE $u_t = c_1 u^d$	Transport $u_t = c_2 u_x$	Diffusion $u_t = c_3 u_{xx}$	Others	
$\sigma_Z = 0.1$					
STRidge ($d_{\text{tol}} = 2$)					
$(1, u, u^2, \ldots, u^3 u_{xxx})$	4 (1)	3	2 (1)	82 (11)	9
$(1, u, u^2, \ldots, u^4 u_{xxxx})$	0	0	0	91 (10)	9 (2)
$(1, u, u^2, \ldots, u^5 u_{xxxxx})$	0	0	0	81 (11)	19 (1)
ARGOS-RAL					
$(1, u, u^2, \ldots, u^3 u_{xxx})$	1 (1)	2 (1)	4 (2)	54 (29)	39 (23)
$(1, u, u^2, \ldots, u^4 u_{xxxx})$	0	2 (1)	3 (2)	63 (32)	32 (23)
$(1, u, u^2, \ldots, u^5 u_{xxxxx})$	0	0	0	34 (18)	66 (47)
$\sigma_Z = 1$					
STRidge ($d_{\text{tol}} = 2$)					
$(1, u, u^2, \ldots, u^3 u_{xxx})$	4 (1)	4	4 (2)	69 (8)	19 (1)
$(1, u, u^2, \ldots, u^4 u_{xxxx})$	1	0	2	54 (7)	43 (10)
$(1, u, u^2, \ldots, u^5 u_{xxxxx})$	0	0	0	0	100 (18)
ARGOS-RAL					
$(1, u, u^2, \ldots, u^3 u_{xxx})$	0	0	0	0	100 (16)
$(1, u, u^2, \ldots, u^4 u_{xxxx})$	0	0	0	0	100 (17)
$(1, u, u^2, \ldots, u^5 u_{xxxxx})$	0	0	0	0	100 (13)
$\sigma_Z = 10$					
STRidge ($d_{\text{tol}} = 2$)					
$(1, u, u^2, \ldots, u^3 u_{xxx})$	0	0	0	0	100 (11)
$(1, u, u^2, \ldots, u^4 u_{xxxx})$	0	0	0	0	100 (4)
$(1, u, u^2, \ldots, u^5 u_{xxxxx})$	0	0	0	0	100 (12)
ARGOS-RAL					
$(1, u, u^2, \ldots, u^3 u_{xxx})$	0	0	0	0	100 (12)
$(1, u, u^2, \ldots, u^4 u_{xxxx})$	0	0	0	0	100 (9)
$(1, u, u^2, \ldots, u^5 u_{xxxxx})$	0	0	0	0	100 (6)

ultimate test stage for an algorithm: identifying dynamical systems without signal, where we expect success rates to drop to zero. When faced with this condition, an effective algorithm should identify either a null model (with no coefficients) or a dense model (with many terms from the candidate library). However, if the algorithm incorrectly identifies canonical PDEs from pure white noise data, it indicates that further improvements are needed to prevent such misidentifications and ensure the robustness of the method.

We generate 100 white Gaussian noise datasets, each consisting of 2000 spatial (x) and 1000 temporal (t) data points, forming a matrix in $\mathbb{R}^{2000\times1000}$. To investigate the influence of noise variance on the identification process, we use three Gaussian distributions with variances spanning three orders of magnitude: $N(0, 0.1^2)$, $N(0, 1)$, and $N(0, 10^2)$. We aim to determine whether ARGOS-RAL and STRidge can identify canonical PDEs under these noise conditions. Table 1 shows the percentages of different identified models. Based on the PDEs tested by Rudy *et al* [5] and our own study, we define parsimonious models as those having three or fewer nonzero coefficients, suggesting they may correspond to specific physical phenomena. In particular, we highlight three classic differential equations: the ODE $u_t = c_1 u^d$, the transport equation $u_t = c_2 u_x$, and the diffusion equation $u_t = c_3 u_{xx}$. In contrast, we classify models with more than three nonzero coefficients as non-parsimonious, indicating that their coefficient vectors have a dense composition.

Table 1 and figure 6 demonstrate that as the standard deviation of the Gaussian noise increases, both ARGOS-RAL and STRidge tend to identify more non-parsimonious models, as indicated by the probability distributions of the number of identified terms shifting into the shaded region of figure 6. This is the desired behavior when the input signal is pure white noise, as we want to ensure that the algorithms do not identify

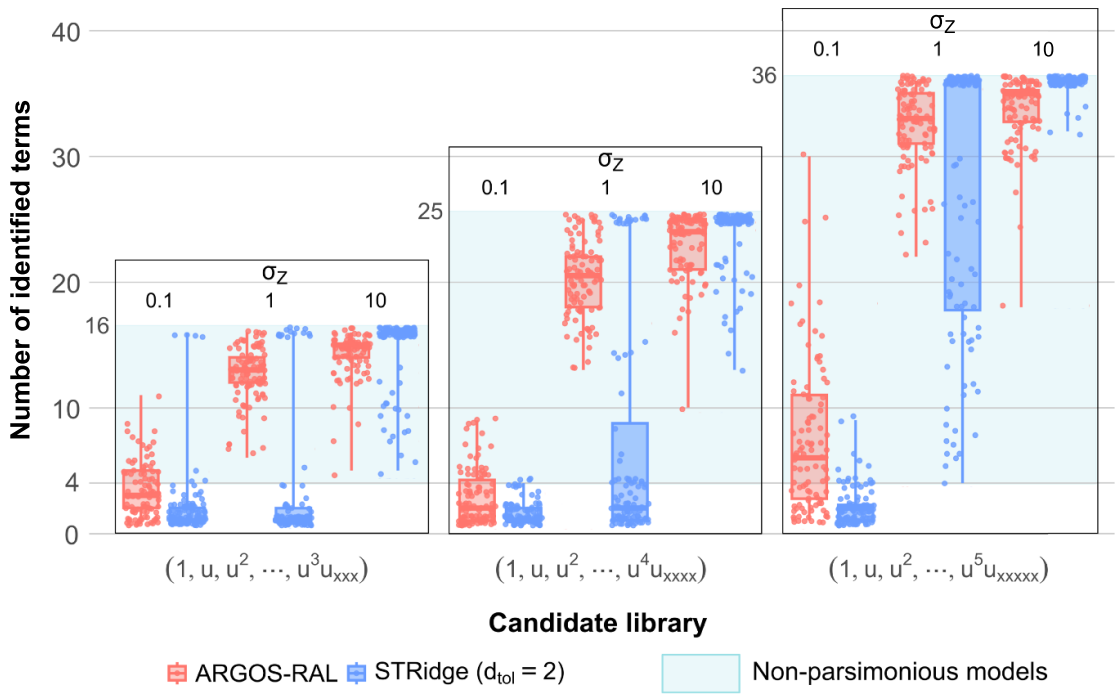

Figure 6. Number of nonzero terms identified from 100 random noise datasets using different candidate function libraries. For each case, we count the number of nonzero coefficients in the sparse regression. We display the distribution of these counts using dots for each of the 100 trials and summarize the results using box plots. Each box plot shows the median (solid horizontal line), interquartile range (box), and minimum and maximum values (whiskers) for the 100 trials. The optimal algorithm should produce boxes located either at zero, indicating a null model, or above four, representing a dense model. The box may span a wide range from four to the maximum number of terms in the library.

parsimonious models in such cases. The difference in behavior between the two methods is most apparent when the noise level is low to moderate ($\sigma_Z \leqslant 1$). In these cases, STRidge's distributions are more spread out and partially located in the parsimonious region, while ARGOS-RAL's distributions are more concentrated in the non-parsimonious region. This suggests that ARGOS-RAL is more effective at avoiding the identification of parsimonious models when the input signal is pure white noise with low to moderate noise levels. As the noise level increases to $\sigma_Z = 10$, both ARGOS-RAL and STRidge consistently identify non-parsimonious models, as evidenced by the concentration of their distributions in the non-parsimonious region of figure 6. This indicates that both methods are effective at avoiding the identification of parsimonious models when the input signal is pure white noise with high noise levels.

4. Conclusions

We designed ARGOS-RAL to automatically tune algorithm hyperparameters, enabling the identification of closed forms of PDEs directly from data. ARGOS-RAL offers several advantages over existing PDE identification methods. First, it automates the process of calculating partial derivatives and constructing the candidate library, reducing manual intervention and streamlining the modeling process. Second, the recurrent adaptive lasso employed by ARGOS-RAL provides a more robust and efficient sparse regression technique compared to the STRidge used in SINDy-based methods. This enables ARGOS-RAL to handle noisy and limited data more effectively, as demonstrated in our numerical experiments. Finally, the linearly increasing computing time indicates that ARGOS-RAL is an efficient method, allowing users to parse large datasets.

ARGOS-RAL shares limitations with other library-based methods like SINDy [3, 5]. First, its effectiveness depends on including the correct governing terms in the candidate library; their absence leads ARGOS-RAL to approximate the PDE with available terms, yielding a non-sparse model. Second, ARGOS-RAL can identify PDEs with non-linear predictors but requires prior knowledge of the function arguments. For example, identifying $\sin(\omega x)$ requires knowing the symbolic form ωx and the numerical value of ω. Third, transforming the identification into a regression problem requires knowing the response variables. We focus on first-order time derivatives [5, 65], but users must know if higher-order derivatives exist in the true equation. This limitation also applies to reconstructing latent space variables, thus limiting the application of ARGOS-RAL in this type of unknown source problems [12]. Finally, while ARGOS-RAL

provides a more computationally efficient approach than ARGOS [19] by focusing on point estimates rather than bootstrapping for confidence intervals, this comes at the cost of losing uncertainty quantification for the estimated coefficients.

When applying ARGOS-RAL to different scientific domains, several challenges arise. One key challenge is determining the appropriate range of candidate terms to include in the library, which often requires domain expertise. In some fields, the governing equations may involve complex nonlinearities or unconventional terms that are difficult to anticipate without prior knowledge. Another challenge is the computational cost of handling high-dimensional data, which is common in many scientific applications. As the number of variables and the complexity of the PDE increase, the size of the candidate library grows exponentially, leading to increased computational demands for sparse regression.

Despite these challenges, ARGOS-RAL offers a promising framework for automating PDE identification in various scientific domains. By leveraging sparse regression techniques and automating key steps in the modeling process, ARGOS-RAL has the potential to accelerate discovery and insight in fields ranging from physics and engineering to biology and climate science.

Data availability statement

The data that support the findings of this study will be openly available following an embargo at the following URL/DOI: https://github.com/Weizhenli/ARGOS-RAL.

Acknowledgments

Rui Carvalho has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No 872172 (TESTBED2 project: www.testbed2.org). Weizhen Li has received funding from the Department of Engineering at Durham University under the Durham Doctoral Studentship programme.

Conflict of interest

The authors declare that they have no conflict of interest.

CRediT authorship contribution statement

Weizhen Li: Writing—original draft, Writing—review & editing, Visualization, Validation, Methodology, Investigation, Software, Formal analysis, Data Curation, Resources, Conceptualization.
Rui Carvalho: Writing—review & editing, Supervision, Funding acquisition, Conceptualization.

Appendix A. Supplementary materials

A.1. Gaussian Blur Kernels

The Gaussian blur convolves data with a Gaussian kernel to smooth it, regardless of the data's dimensionality. This convolution method offers significant benefits for filtering out Gaussian noise, a common noise distribution encountered in data analysis [62]. For one-dimensional spatial PDEs, such as the Burgers' and cable equations, we employ the simplest 2-dimensional Gaussian kernel:

$$\frac{1}{16}\begin{bmatrix}1\\2\\1\end{bmatrix}\otimes\begin{bmatrix}1&2&1\end{bmatrix}=\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}. \tag{A.1}$$

In contrast, for two-dimensional spatial PDEs, the Navier–Stokes and reaction-diffusion equations, we use a 3-dimensional Gaussian kernel:

$$\begin{aligned}&\frac{1}{64}\begin{bmatrix}1\\2\\1\end{bmatrix}\otimes\begin{bmatrix}1&2&1\end{bmatrix}\otimes\begin{bmatrix}1\\2\\1\end{bmatrix}\\&=\frac{1}{64}\left[\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\begin{bmatrix}2&4&2\\4&8&4\\2&4&2\end{bmatrix}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\right]\end{aligned} \tag{A.2}$$

A.2. Additional PDE Test Cases

Here, we demonstrate how to solve the PDEs presented in this paper. As we use the Fourier spectral method to solve the Burgers', Cable, reaction-diffusion, advection-diffusion, transport, and diffusion systems, the boundary conditions for them are the periodic boundary conditions.

A.2.1. Burgers' equation

We can derive Burgers' equation from the Navier–Stokes equation for the velocity field by dropping the pressure gradient term. Unlike the Navier–Stokes equation, Burgers' equation does not exhibit turbulent behavior, and we can transform it to linear form via the Cole-Hopf transformation [66]:

$$u_t = -uu_x + \nu u_{xx}. \tag{A.3}$$

We solve Burgers' equation (A.3) using the Fourier spectral method [67] with the *ode45* function in MATLAB. We set $\nu = 0.1$, $x \in [-8, 8]$ with 256 points, $t \in [0, 10]$ with 101 points, and the initial condition is a Gaussian function: $\exp\left(-(x+2)^2\right)$.

A.2.2. Cable equation

The cable equation, shown in equation (A.4), quantitatively describes the electrical behavior of nerve axons and other cable-like structures in biological systems. It captures the electrical circuit of current flow and voltage change both within and between neurons. The equation is derived from a circuit model of the membrane and its intracellular and extracellular space. The cable equation plays a crucial role as an important PDE in biophysical studies, helping researchers understand how electrical signals change in diseases and disorders. By identifying the cable equation, researchers can diagnose these negative conditions by checking for changes in capacitances c_m, resistances r_m, and axial resistance r_a, r_e:

$$\lambda^2 \frac{\partial^2 V}{\partial x^2} = \tau \frac{\partial V}{\partial t} + V \quad \text{where} \quad \lambda = \sqrt{\frac{r_m}{r_e + r_a}} \quad \text{and} \quad \tau = r_m c_m. \tag{A.4}$$

We solve the cable equation using *odeint* function in Python with the Fourier spectral method. We set $\lambda = 1$, $\tau = 1$, $x \in [-4, 4]$ with $\Delta x = 0.1$, $t \in [0, 5]$ with $\Delta t = 0.01$, and use a Gaussian function $\exp\left(-x^2\right)$ as the initial condition.

A.2.3. Navier–Stokes

We simulate the two-dimensional Navier–Stokes equation for fluid flow around a circular cylinder using the Immersed Boundary Projection Method (IBPM) [68, 69]. The two-dimensional velocity components are denoted by u and v, while ω represents the vorticity away from the circular cylinder of diameter one and mass center at $(x = 1, y = 2)$. The boundary condition for the IBPM is the no-slip boundary condition. We set the Reynolds number to 100 and aim to identify the equation

$$\omega_t = 0.01\omega_{xx} + 0.01\omega_{yy} - u\omega_x - v\omega_y. \tag{A.5}$$

The spatial domain spans $x \in [0, 9]$ with $\Delta x = 0.02$, $y \in [0, 4]$ with $\Delta y = 0.02$, and the temporal domain covers $t \in [300, 330]$ with $\Delta t = 0.02$. We save the flow data every ten snapshots. This setup generates a simulated dataset containing approximately 13.5 million points ($449 \times 199 \times 151$). However, constructing the candidate library $\boldsymbol{\Theta}$ for such a large dataset poses computational challenges. To facilitate the evaluation, we randomly sample points within the red rectangular area shown in figure 4(c) at each snapshot.

A.2.4. Reaction-diffusion

Reaction-diffusion systems offer a versatile framework to model pattern formation in various natural phenomena in chemistry, biology, geology, physics, and ecology. These systems give rise to a rich tapestry of periodic patterns, including spots, zigzags, spiral waves, and rolls. In our analysis, we focus on a widely studied class of reaction-diffusion systems known as the $\lambda - \omega$ systems, described by the following coupled PDEs:

$$u_t = 0.1u_{xx} + 0.1u_{yy} + u - uv^2 - u^3 + v^3 + u^2 v \tag{A.6}$$

$$v_t = 0.1v_{xx} + 0.1v_{yy} + v - uv^2 - u^3 - v^3 - u^2 v. \tag{A.7}$$

To generate data for our analysis, we employ the simulation method described in [5]. We discretize the spatial domain $x, y \in [-10, 10]$ using a 512×512 grid and evolve the system over the time interval $t \in [0, 10]$ using 201 time steps. This procedure yields a rich dataset comprising 52 690 944 spatiotemporal points on a $512 \times 512 \times 201$ grid, providing a comprehensive characterization of the system's dynamics.

A.2.5. QHO

The QHO models the parabolic potential of a harmonic oscillator in quantum mechanics. It simulates the time evolution of the wave function associated with a particle in the parabolic potential, providing the probability distribution of the particle's position at any given time by taking the squared magnitude of the wave function. The energy levels of a QHO are quantized, meaning they can only assume specific, discrete values. Furthermore, even if we form a statistical distribution from multiple experiments, it will lack information on the intricate phase of the wave function. We use the following equation:

$$u_t = \frac{1}{2} i u_{xx} - iuV = \frac{1}{2} i u_{xx} - \frac{x^2}{2} iu. \tag{A.8}$$

To obtain data on the QHO, we employ the operator splitting method with the Fourier transform. We consider the time domain $t \in [0, 10]$ with $\Delta t = 0.025$, and the space domain $x \in [-7.5, 7.5]$ with $\Delta x = 15/512$, using a Gaussian $\exp(-((x-1)/2)^2)$ as the initial condition.

When performing a sparse regression on complex numbers, we transform the regression from complex to real numbers. For each $y_i = y_i^R + iy_i^I$, where the normal i is the imaginary number, the subscript $_i$ represents the ith observation, we can reform it as

$$\begin{aligned} y_i^R + iy_i^I &= \beta_0^R + i\beta_0^I + \sum_{j=1}^{p} \left[\left(\beta_j^R + i\beta_j^I \right) \left(x_{ij}^R + ix_{ij}^I \right) \right] + \epsilon^R + i\epsilon^I \\ &= \beta_0^R + i\beta_0^I + \left(\beta_1^R + i\beta_1^I \right) \left(x_{i1}^R + ix_{i1}^I \right) + \left(\beta_2^R + i\beta_2^I \right) \left(x_{i2}^R + ix_{i2}^I \right) \\ &\quad + \cdots + \left(\beta_p^R + i\beta_p^I \right) \left(x_{ip}^R + ix_{ip}^I \right) + \epsilon^R + i\epsilon^I \\ &= \beta_0^R + i\beta_0^I + \sum_{j=1}^{p} \left(x_{ij}^R \beta_j^R - x_{ij}^I \beta_j^I \right) + i \sum_{j=1}^{p} \left(x_{ij}^R \beta_j^I + x_{ij}^I \beta_j^R \right) + \epsilon^R + i\epsilon^I \end{aligned}$$

and split it to extract two equations for both real and imaginary parts

$$\begin{cases} y_i^R = \beta_0^R + \sum_{j=1}^{p} x_{ij}^R \beta_j^R - \sum_{j=1}^{p} x_{ij}^I \beta_j^I + \epsilon^R \\ y_i^I = \beta_0^I + \sum_{j=1}^{p} x_{ij}^I \beta_j^R + \sum_{j=1}^{p} x_{ij}^R \beta_j^I + \epsilon^I \end{cases} . \tag{A.9}$$

Based on equation (A.9), we can organize the dataset as:

$$Y = \begin{bmatrix} y_1^R \\ y_1^I \\ y_2^R \\ y_2^I \\ \vdots \\ y_n^R \\ y_n^I \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{11}^R & -x_{11}^I & x_{12}^R & -x_{12}^I & \cdots & x_{1p}^R & -x_{1p}^I \\ x_{11}^I & x_{11}^R & x_{12}^I & x_{1R}^R & \cdots & x_{1p}^I & x_{1p}^R \\ x_{21}^R & -x_{21}^I & x_{22}^R & -x_{22}^I & \cdots & x_{21}^R & -x_{21}^I \\ x_{21}^I & x_{21}^R & x_{22}^I & x_{22}^R & \cdots & x_{21}^I & x_{21}^R \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1}^R & -x_{n1}^I & x_{n2}^R & -x_{n2}^I & \cdots & x_{n1}^R & -x_{n1}^I \\ x_{n1}^I & x_{n1}^R & x_{n2}^I & x_{n2}^R & \cdots & x_{n1}^I & x_{n1}^R \end{bmatrix}.$$

Finally, we implement the recurrent adaptive lasso and STRidge on the re-posited Y and $\mathbf{X}$.

A.2.6. Advection-diffusion equation

The advection-diffusion equation, which combines advection and diffusion terms, describes the transport and dispersion of quantities such as temperature, substance concentration, or fluid velocity in various scientific and engineering contexts. We can express this equation as follows:

$$c_t = Dc_{xx} - uc_x. \tag{A.10}$$

We solve the advection-diffusion equation using the Fourier spectral method and the *odeint* function in Python. We set the diffusion coefficient $D = 1$, the advection velocity $u = 1$, and consider the spatial domain $x \in [-10, 10]$ with resolution $\Delta x = 0.1$ and the temporal domain $t \in [0, 10]$ with resolution $\Delta t = 0.01$. We use a Gaussian function of the form $\exp\left(-(x+2)^2\right)$ as an initial condition.

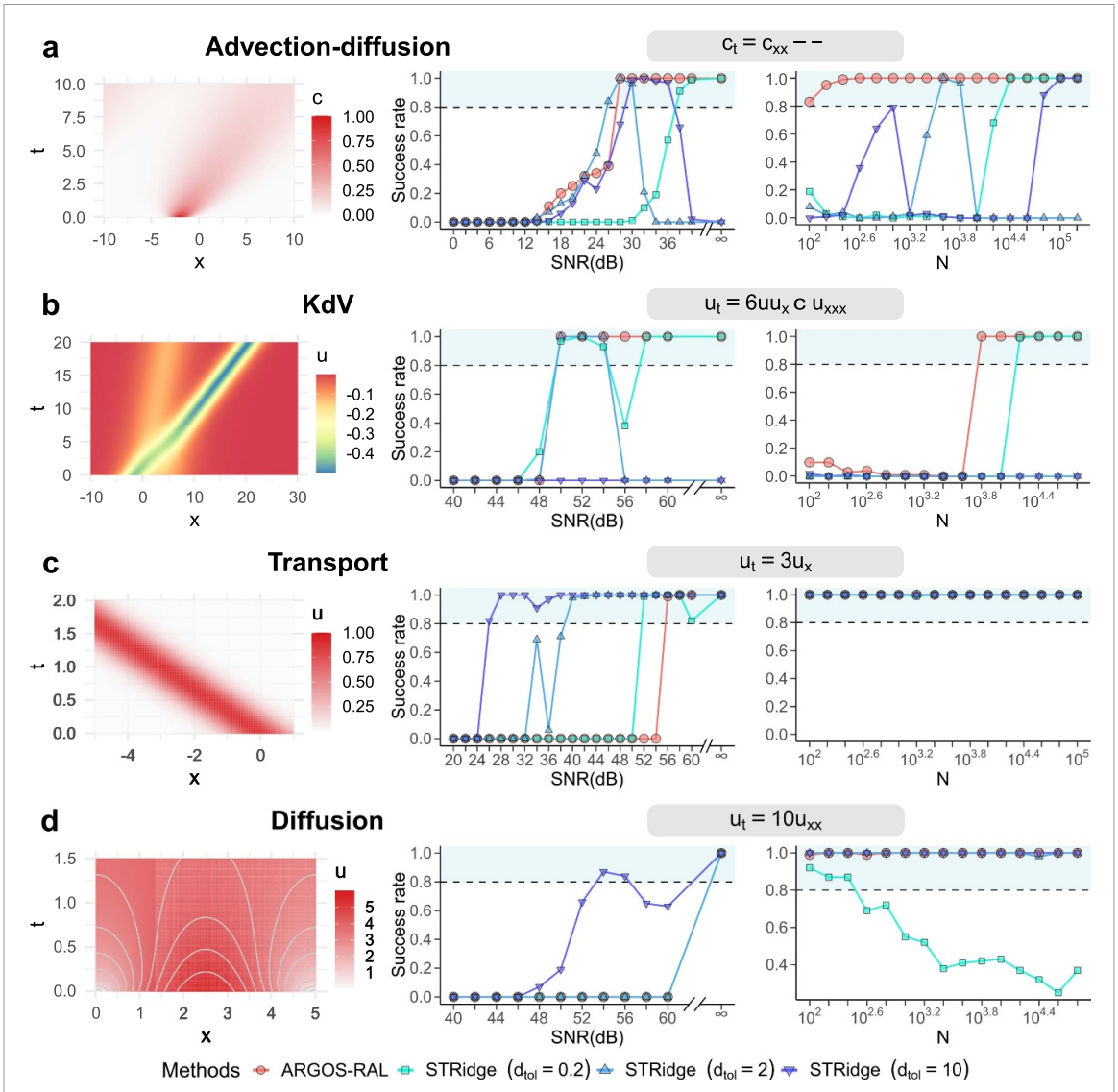

Figure A1. Success rates of ARGOS-RAL and STRidge in identifying (a) advection-diffusion, (b) KdV, (c) transport, and (d) diffusion equations with different SNRs and sample sizes. To analyze the noise tolerance, we added noise to the PDE solutions at different SNR levels. For the sample size analysis, we randomly sampled points from the set $\{\mathbf{u}_t, \boldsymbol{\Theta}(\mathbf{u})\}$ based on noiseless PDE solutions. For sample size analysis, we randomly sample points from $\{\mathbf{u}_t, \boldsymbol{\Theta}(\mathbf{u})\}$ set based on noiseless PDE solutions. Lines are only used to link points, not to fit points. The plots demonstrate that our method maintains high success rates in identifying the correct PDE even under significant noise and with limited sample sizes.

A.2.7. The KdV equation

The KdV equation describes wave propagation on shallow water surfaces. The KdV equation solution reveals that an isolated traveling wave exhibits linear behavior, but nonlinear interactions emerge when multiple waves are present. Moreover, the dependence of wave velocity on wave amplitude ensures that any solution with multiple amplitudes will display nonlinear behavior, regardless of the interaction. Equation (A.11) presents the formula for the KdV equation:

$$u_t = -6uu_x - u_{xxx}. \tag{A.11}$$

We employ the two-soliton solution [70] to solve the KdV equation:

$$\begin{aligned} w(x,t) &= -2\frac{\partial^2}{\partial x^2}\ln\left(1 + B_1 e^{\theta_1} + B_2 e^{\theta_2} + AB_1B_2 e^{\theta_1+\theta_2}\right) \\ \theta_1 &= a_1 x - a_1^3 t, \quad \theta_2 = a_2 x - a_2^3 t, \quad A = \left(\frac{a_1 - a_2}{a_1 + a_2}\right)^2, \end{aligned} \tag{A.12}$$

where a_1, a_2, B_1, and B_2 are arbitrary constants. We set the following parameters: 201 time steps ($n = 201$) with $t \in [0, 20]$, 512 spatial points ($m = 512$) with $x \in [-30, -30]$, and $a_1 = 0.5$, $a_2 = 1$, $B_1 = 1$, $B_2 = 5$. We use the Dirichlet boundary condition, $u(x,t) = 0$ at $x = \pm\infty$.

A.2.8. Transport equation

The transport equation, equation (A.13), plays a fundamental role in science and engineering, describing the spatiotemporal evolution of scalar quantities or vector fields. We solve this PDE using the analytical solution, equation (A.14), with $c = 3$ to generate the data for our study. The spatial domain spans $x \in [-5, 1]$ with resolution $\Delta x = 0.01$, while the temporal domain covers $t \in [0, 2]$ with timestep $\Delta t = 0.01$.

$$u_t = cu_x, \quad c > 0 \tag{A.13}$$

$$u(x,t) = \exp\left(-(x+ct)^2\right). \tag{A.14}$$

A.2.9. Diffusion equation

The diffusion (heat) equation, equation (A.15), plays a crucial role in many scientific and engineering fields, including solid-state physics, materials science, environmental science, and computational fluid dynamics. This equation elucidates the fundamental process of heat diffusion, enabling engineers to gain deep insights into heat conduction, thermal conductivity, and temperature-dependent phenomena in solids and other materials. Here, we use the analytic solution, equation (A.16), to generate the data. We set $x \in [0, 5]$ with $\Delta x = 0.01$ and $t \in [0, 1.5]$ with $\Delta t = 0.01$, and choose the initial condition as $6\sin(\pi x/L)$.

$$u_t = 10u_{xx} \tag{A.15}$$

$$u(x,t) = 6\sin\left(\frac{\pi x}{L}\right)\mathbf{e}^{-k\left(\frac{\pi}{L}\right)^2 t}, \quad k = 10. \tag{A.16}$$

The diffusion equation and its analytic solution provide a powerful framework for understanding and predicting heat transfer in various systems. By carefully selecting the spatial and temporal domains and the initial condition, we can model a wide range of real-world scenarios and gain valuable insights into the underlying physical processes.

ORCID iDs

Weizhen Li https://orcid.org/0000-0001-9368-236X
Rui Carvalho https://orcid.org/0000-0002-3279-4218

References

[1] Schmidt M and Lipson H 2009 *Science* **324** 81–85
[2] Xun X, Cao J, Mallick B, Maity A and Carroll R J 2013 *J. Am. Stat. Assoc.* **108** 1009–20
[3] Brunton S L, Proctor J L and Kutz J N 2016 *Proc. Natl Acad. Sci.* **113** 3932–7
[4] Schaeffer H 2017 *Proc. R. Soc.* A **473** 20160446
[5] Rudy S H, Brunton S L, Proctor J L and Kutz J N 2017 *Sci. Adv.* **3** e1602614
[6] Raissi M, Perdikaris P and Karniadakis G 2019 *J. Comput. Phys.* **378** 686–707
[7] Meng P, Chai Y and Yin W 2023 *Universe* **9** 148
[8] Long Z, Lu Y and Dong B 2019 *J. Comput. Phys.* **399** 108925
[9] Both G J, Choudhury S, Sens P and Kusters R 2021 *J. Comput. Phys.* **428** 109985
[10] Schaeffer H, Tran G and Ward R 2018 *SIAM J. Appl. Math.* **78** 3279–95
[11] Guimerá R, Reichardt I, Aguilar-Mogas A, Massucci F A, Miranda M, Pallarés J and Sales-Pardo M 2020 *Sci. Adv.* **6** eaav6971
[12] Lu P Y, Ariño Bernad J and Soljačić M 2022 *Commun. Phys.* **5** 206
[13] Zhang E, Dao M, Karniadakis G E and Suresh S 2022 *Sci. Adv.* **8** eabk0644
[14] Fajardo-Fontiveros O, Reichardt I, De Los Ríos H R, Duch J, Sales-Pardo M and Guimerá R 2023 *Nat. Commun.* **14** 1043
[15] Jiang Y-X, Xiong X, Zhang S, Wang J-X, Li J-C and Du L 2021 *Nonlinear Dyn.* **105** 2775–94
[16] Maddu S, Cheeseman B L, Sbalzarini I F and Müller C L 2022 *Proc. R. Soc.* A **478** 20210916
[17] Cai Y, Wang X, Joós G and Kamwa I 2023 *IEEE Trans. Power Syst.* **38** 2085–99
[18] Sun X, Qian J and Xu J 2024 *Int. J. Mech. Sci.* **265** 108905
[19] Egan K, Li W and Carvalho R 2024 *Commun. Phys.* **7** 1–10
[20] Varah J M 1982 *SIAM J. Sci. Stat. Comput.* **3** 28–46
[21] Crutchfield J P and Mcnamara B S 1987 *Complex Syst.* **1** 417–52
[22] Bär M, Hegger R and Kantz H 1999 *Phys. Rev.* E **59** 337–42
[23] Müller T and Timmer J 2002 *Physica* D **171** 1–7
[24] Liang H and Wu H 2008 *J. Am. Stat. Assoc.* **103** 1570–83
[25] Wu H, Ding A A and De Gruttola V 1998 *Stat. Med.* **17** 2463–85
[26] Wu H and Ding A A 1999 *Biometrics* **55** 410–8
[27] Putter H, Heisterkamp S H, Lange J M A and de Wolf F 2002 *Stat. Med.* **21** 2199–214
[28] Bongard J and Lipson H 2007 *Proc. Natl Acad. Sci.* **104** 9943
[29] Udrescu S-M and Tegmark M 2020 *Sci. Adv.* **6** eaay2631

[30] Xu H and Zhang D 2021 *Phys. Rev. Res.* **3** 033270
[31] Rudy S, Alla A, Brunton S L and Kutz J N 2019 *SIAM J. Appl. Dyn. Syst.* **18** 643–60
[32] Kaheman K, Kutz J N and Brunton S L 2020 *Proc. R. Soc.* A **476** 20200279
[33] Messenger D A and Bortz D M 2021 *J. Comput. Phys.* **443** 110525
[34] Cortiella A, Park K C and Doostan A 2021 *Comput. Methods Appl. Mech. Eng.* **376** 113620
[35] Fasel U, Kutz J N, Brunton B W and Brunton S L 2022 *Proc. R. Soc.* A **478** 20210904
[36] Kaheman K, Brunton S L and Nathan Kutz J 2022 *Mach. Learn.: Sci. Technol.* **3** 015031
[37] Li Y, Wu K and Liu J 2023 *Phys. Rev. Res.* **5** 023126
[38] Wentz J and Doostan A 2023 *Comput. Methods Appl. Mech. Eng.* **413** 116096
[39] Loiseau J-C, Noack B R and Brunton S L 2018 *J. Fluid Mech.* **844** 459–90
[40] Duraisamy K, Iaccarino G and Xiao H 2019 *Annu. Rev. Fluid Mech.* **51** 357–77
[41] Li S, Kaiser E, Laima S, Li H, Brunton S L and Kutz J N 2019 *Phys. Rev.* E **100** 022220
[42] Mangan N M, Brunton S L, Proctor J L and Kutz J N 2016 *IEEE Trans. Mol. Biol. Multi-Scale Commun.* **2** 52–63
[43] Hoffmann M, Fröhner C and Noé F 2019 *J. Chem. Phys.* **150** 025101
[44] Lagergren J H, Nardini J T, Michael Lavigne G, Rutter E M and Flores K B 2020 *Proc. R. Soc.* A **476** 20190800
[45] Chen Z, Liu Y and Sun H 2021 *Nat. Commun.* **12** 6136
[46] Zhang Z and Liu Y 2021 *J. Comput. Phys.* **446** 110657
[47] Thanasutives P, Morita T, Numao M and Fukui K i 2023 *Mach. Learn.: Sci. Technol.* **4** 015009
[48] Jia D, Zhou X, Li S, Liu S and Shi H 2023 *Mach. Learn.: Sci. Technol.* **4** 045008
[49] Savitzky A and Golay M J E 1964 *Anal. Chem.* **36** 1627–39
[50] Breugel F V, Kutz J N and Brunton B W 2020 *IEEE Access* **8** 196865–77
[51] Schafer R W 2011 *IEEE Signal Process. Mag.* **28** 111–7
[52] Zou H 2006 *J. Am. Stat. Assoc.* **101** 1418–29
[53] Friedman J H, Hastie T and Tibshirani R 2010 *J. Stat. Softw.* **33** 1–22
[54] Hansen P C 1992 *SIAM Rev.* **34** 561–80
[55] Nasehi Tehrani J, McEwan A, Jin C and van Schaik A 2012 *Appl. Math. Model.* **36** 1095–105
[56] Cultrera A and Callegaro L 2020 *IOP SciNotes* **1** 025004
[57] Boyd S and Vandenberghe L 2004 Approximation and fitting *Convex Optimization* (Cambridge University Press) pp 291–350
[58] Bühlmann P and van de Geer S 2011 Lasso for linear models *Statistics for High-Dimensional Data: Methods, Theory and Applications* (*Springer Series in Statistics*) (Springer) pp 7–43
[59] Tibshirani R 1996 *J. R. Stat.* B **58** 267–88
[60] Yang Y 2005 *Biometrika* **92** 937–50
[61] Aho K, Derryberry D and Peterson T 2014 *Ecology* **95** 631–6
[62] Gonzalez R C and Woods R E 2018 Intensity transformations and spatial filtering *Digital Image Processing* 4th edn (Pearson) pp 119–202
[63] Rudy S H, Nathan Kutz J and Brunton S L 2019 *J. Comput. Phys.* **396** 483–506
[64] Modonesi R, Dalai M, Migliorati P and Leonardi R 2020 *IEEE Commun. Lett.* **24** 2119–22
[65] Chen B, Huang K, Raghupathi S, Chandratreya I, Du Q and Lipson H 2022 *Nat. Comput. Sci.* **2** 433–42
[66] Cross M C and Hohenberg P C 1993 *Rev. Mod. Phys.* **65** 851–1112
[67] Brunton S L and Kutz J N 2022 Fourier and wavelet transforms *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control* 2nd edn (Cambridge University Press) pp 53–96
[68] Taira K and Colonius T 2007 *J. Comput. Phys.* **225** 2118–37
[69] Colonius T and Taira K 2008 *Immersed Bound. Method Ext.* **197** 2131–46
[70] Polyanin A D and Zaitsev V F 2012 Third-order equations *Handbook of Nonlinear Partial Differential Equations* 2nd edn C and H (Chapman and Hall/CRC) pp 857–976