Discovering Governing Equations of Dynamical Systems

LI, WEIZHEN

Discovering Governing Equations of Dynamical Systems

Weizhen Li

A Thesis presented for the degree of
Doctor of Philosophy

Department of Engineering
Durham University
United Kingdom

November 2024

Dedication

To Mother, Father, grandmother and grandfather,
Thank you for your unconditional support!

Abstract

Data-driven discovery methods are gradually changing the study of dynamical systems in many fields, such as engineering, physics, economics, biology, and chemistry. However, traditional methods of finding governing equations are based on mathematical derivation and require a lot of data, expertise, and computing time. This thesis explores automating the discovery of governing equations from data, especially using the Automatic Regression for Governing Equations (ARGOS) framework, which marks an advancement in the data-driven identification of dynamical systems. The research begins with an overview of data-centric engineering, underlining the essential statistical and machine learning methodologies. It then focuses on the Sparse Identification of Nonlinear Dynamical Systems (SINDy) framework to demonstrate the requirements of automating the identification process. Subsequent chapters present two innovative methods for automatically calculating derivatives and identifying partial differential equations (PDEs) from data with limited prior knowledge. These methods are developed to expand the usage of the ARGOS framework and rigorously benchmarked against other state-of-the-art algorithms using success rates with varying signal-to-noise ratios (SNRs) and sample sizes, demonstrating the accuracy of the proposed methods in model discovery. The application of the ARGOS framework is demonstrated through the case study of COVID-19 data in mainland China using Bayesian ARGOS, which is a new ARGOS extension and provides interpretable and meaningful results. The thesis also tackles the computational challenges and outlines future directions in automating the discovery of dynamical systems, including stochastic differential equations. Overall, this thesis has three contributions. First, it proposes an automatic framework to reduce manual parameter tuning for dynamical system identification. Second, it offers systematic testing templates, facilitating advancements across diverse scientific disciplines. Third, analysing COVID-19 data illustrates that the ARGOS framework is a candidate method for dealing with real-world problems.

Declaration

The work in this thesis is based on research carried out at the Department of Engineering, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Parts of this work have been or will be submitted to the following refereed journals. The following provides further detail regarding each author's contribution to the papers.

- **Chapter 4**
  - This chapter is a part of the work of the paper "Automatically discovering ordinary differential equations from data with sparse regression" [1], for which I cooperated with Dr. Kevin Egan, whose supervisors are the same as mine. This paper has been published in Communications Physics. We wrote this paper together.
  - This chapter has no overlap with Kevin's thesis. The results of evaluating ARGOS were written in Kevin's thesis, while I assessed ESINDy and showed the results in my thesis.
  - I also reproduced the results of other SINDy-based methods (UQ SINDy and modified SINDy), which were only submitted to referees to reply to

their comments and have not been published in the paper or shown in Kevin's thesis.

  - I wrote testing codes for running on personal computers and demos for running on Hamilton (Durham University's high-performance computer), and Kevin converted demos to executable scripts and submitted all tests to Hamilton.
  - Dr. Rui Carvalho provided oversight and suggestions for improvement on the text.

- **Chapter 5**
  - I plan to submit this chapter to IEEE Access upon the review reply of Chapter 6 because the proposed method has been used there without evaluation. I need to cooperate with others to apply the proposed method to solve a specific real-world problem.
  - Dr. Rui Carvalho provided oversight and suggestions for improvement on the text.

- **Chapter 6**
  - I have submitted this chapter to Machine Learning: Science and Technology and resubmitted a revision to reply referees on 14 June 2024. I am awaiting the second round of peer review.
  - Dr Rui Carvalho provided expansive insight, ideas, and collaboration for writing the paper. Rui helped develop the method, along with editing, maintaining, and aligning the paper so that it was written in the style of the top journals. Rui's constant time, leadership, and guidance have significantly impacted the structure and overall quality of the paper.

- **Chapter 7**
  - I plan to submit this chapter to Cell Reports Physical Science as analysing COVID-19 data is still important in epidemiology.

- Dr. Rui Carvalho provided oversight and suggestions for improvement on the text.

Acknowledgements

First and foremost, I am profoundly grateful to Dr Rui Carvalho for his exceptional mentorship during my research project. As my primary supervisor, Rui consistently challenged me to better myself and provided extraordinary patience and guidance in helping me achieve my goals throughout this research project. Rui's insights have greatly deepened my understanding of complicated methods discussed in my thesis, teaching me valuable experience in becoming an excellent scientific researcher. I am honoured to learn from Rui's rigorous and nuanced approach to solving problems and seeking truth, a mindset I will carry forward in my future career and personal endeavours.

I would also like to express my deep appreciation to my second supervisor, Professor Hongjian Sun, for his guidance during my research project. Hong has provided me with opportunities, support, and supportive advice to achieve great success in machine learning. I opened my eyes and learned many potential research topics in the Smart Grid Seminar. Hong's optimism and support have advanced my career and enriched my personal development.

My thanks also go to Dr Peter Matthews and Dr Stefano Giani for their valuable advice during my review sessions, which greatly assisted me in understanding and presenting my research findings. Furthermore, I am thankful to the Department of Engineering at Durham University for their financial support through the Durham Doctoral Studentship programme, enabling me to pursue my research with adequate resources.

I want to thank my teammate, Dr Kevin Egan, who graduated at the beginning of 2024. Spending time in the office with Kevin, discussing different fields of machine learning, system identification, differential equations, and competing over whose code was better for implementation, has given me some of the best experiences of my life.

My thanks also go to my other teammate, Yuzheng Zhang, who is experienced in project management. His unique insight greatly improved my working efficiency and has driven me to think critically about traditional and new techniques.

Thanks to all my teachers. Without their teaching, I would not have had the broad base of knowledge to support me step by step in getting to where I am now.

Lastly, I would like to thank my parents and grandparents. Their constant support has helped me stay positive and work hard throughout my project, and I cannot express my thanks enough for their belief in my success.

Contents

List of Figures

List of Tables

List of Symbols

General Notation

$\mathbb{R}$	Real numbers, p. 18
$\mathbb{C}$	Complex numbers, p. 54
$\mathbb{E}$	The expected value of a function, p. 12
n	Number of observations, p. 5
t	Time measurements, p. 8
p	Number of predictors, p. 5
m	Number of the spatial grid in partial differential equation, p. 8 or Number of state space variables in ordinary differential equations, p. 49
ϵ	Irreducible error from noise and residuals of a model, p. 8
x_{ij}	The ith observation of the jth predictor in regression analysis, p. 6
$x(t)$	State of the system at times $t_1, t_2, \ldots, t_n$, p. 49
u_{ij}	The partial differential equation solution at time i and space j, p. 8
$\mathbf{X}$	Design matrix ($n \times p$) in normal regression analysis & the state space matrix ($n \times m$) for describing ordinary differential equations, p. 50
$\dot{\mathbf{X}}$	Derivative matrix of $\mathbf{X}$ ($N \times m$), p. 50
$\mathbf{y}$	Response (dependent) variable vector (length n) p. 7
$\hat{\mathbf{y}}$	Estimated values of $\mathbf{y}$ (length n) p. 7

$\mathbf{U}$	Matrix containing solutions of a partial differential equation, p. 8
$\mathbf{u}$	Vectorised $\mathbf{U}$, p. 55
$\mathbf{\Theta}(\mathbf{X})$	Design matrix for system identification ($N \times p$), p. 50
$\theta(x)_{\mathrm{F}}$	Feature vector of symbolic functions ($p \times 1$), p. 49
$\mathbf{\Theta}$	The candidate library when identifying dynamical systems, p. 54
β	Vector of regression coefficients ($p \times 1$), p. 8
$\hat{\beta}$	Vector of estimated regression coefficients ($p \times 1$), p. 7
w	Regression weights vector (px1), p. 30
log	The natural logarithm, also written as ln, p. 21
λ	Regularisation parameter in shrinkage regression, p. 28
vec	Vectorising operation of a matrix, p. 52
μ	The mean value of a variable, p. 15
Var	The variance of a random variable, p. 11
$\mathcal{N}(\mu, \sigma^2)$	The normal (Gaussian) distribution with mean μ and variance σ^2 p. 10
$P(y; \psi)$	The probability density function of variable y with parameter ψ, p. 13
$\mathbf{A} \otimes \mathbf{B}$	Kronecker (tensor) product of matrix $\mathbf{A}$ and $\mathbf{B}$, p 36

Nomenclature

RSS	Residual sum of squares, p. 10
TSS	Total sum of squares, p. 12
OLS	Ordinary least squares, p. 10
MSE	Mean squared error, p. 20
RMSE	Root mean squared error, p. 21
AIC	Akaike information criterion, p. 21
BIC	Bayesian information criterion, p. 22
MLE	Maximum likelihood estimation, p. 10
SNR	Signal-to-noise ratio, p. 35
MCMC	Markov chain Monte Carlo, p. 15

ODE	Ordinary differential equation, p. 3
PDE	Partial differential equation, p. 3
lasso	Least absolute shrinkage and selection operator, p. 28
ARGOS	Automatic regression for governing equations, p. 5
SINDy	Sparse identification of nonlinear dynamics, p. 4
ARGOS-RAL	Automatic regression for governing equations with recurrent adaptive lasso, p. 89

CHAPTER 1

Introduction

1.1 Background

The advent of data-driven methods marks a revolutionary era in modern scientific and engineering research [1–4], which leverages the power of big data and advanced machine learning algorithms to analyse complex problems across various domains, such as manufacturing [5–8], materials science [9, 10], bioengineering [11], and construction [12]. Traditionally, engineering solutions relied heavily on empirical and theoretical models based on physical laws. However, with the exponential increase in data volume, velocity, and variety, driven by advancements in sensor technology and digital connectivity, the engineering discipline has witnessed a paradigm shift towards data-centric approaches.

One popular research area in data-centric engineering is system identification [2, 3, 13]. This area solves an inverse problem by deriving governing equations of dynamical systems, particularly differential equations, from observational data [3, 4, 14–17]. Data-driven methods use datasets to model, predict, and optimise practical problems, surpassing the limitations of traditional approaches that often require simplified assumptions. These methods employ machine learning algorithms, statistical tools, and signal processing techniques to denoise, analyse, interpret and predict empirical data from real-world systems, leading to more accurate, reliable, and efficient models.

Machine learning plays an important role in developing data-driven discovery frameworks by adjusting the parameters of governing equations using collected data. This helps to identify symbolic terms of governing equations and uncover complex dynamical systems. By incorporating statistical theory into this process, the impact of randomness in machine learning algorithms is minimised, resulting in more robust and consistent outcomes [1]. Additionally, applying signal processing techniques within these frameworks helps identify dynamical systems from noisy data, a frequent challenge in real-world scenarios.

Using these composite data-driven methods, researchers can reveal patterns and relationships in the data that may not be apparent from first principles, studying new insights into complex systems [9, 18]. These methods 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 [19–22]. Furthermore, by reducing the need for manual intervention and domain expertise, data-driven methods can significantly streamline the discovery process [1].

Despite the advancements in identifying differential equations using neural networks methods, such as symbolic regression [15, 23–26] and deep learning [17, 27, 28], significant challenges remain, particularly in the interpretation and setting parameters of these methods [29]. However, sparse regression has emerged as an alternative to identifying governing equations by selecting nonzero active terms from a candidate library. This advancement promises to accelerate discoveries across various scientific fields, such as smart grids [21], fluid mechanics [30], biology [11] and epidemiology [19]. Using these statistical-based data-driven methods, scientists and engineers can evaluate their identified model, ensuring the assumptions and predictions of employed methods align with the observed data. Moreover, these approaches allow for the automation of the model discovery process since sparse regression only requires a few parameters, which optimisation algorithms can fine-tune. This automation discovery enables scientists and engineers to improve the efficiency of the research by focusing more on systems' insight properties rather than finding interpretable systems.

This thesis will demonstrate contributions to automatically discovering ordinary and partial differential equations (ODEs and PDEs) of dynamical systems from data by employing signal processing, sparse regression and Bayesian methods. Different from previous studies in this area [3, 4, 31], the methods shown in the thesis enable engineers to effectively find various types of ODEs and PDEs without expert knowledge and manually parameters tuning. Furthermore, the thesis uses both numerical and real-world datasets to explore the advantages and drawbacks of the proposed data-driven methods for dealing with the inverse problem. Hopefully, these developments will improve the scientific methods for automatically discovering underlying laws describing many intricate dynamics.

1.2 Motivation

Dynamical systems are integral to numerous disciplines, reflecting the broad applicability and essential need for such models in science and engineering. Understanding their governing equations facilitates critical functions like predicting, estimating, controlling, and analysing structural stability and bifurcations.

In analysing engineering applications, complex systems can mathematically model many physical phenomena. Knowing the governing equations of these systems enables precise forecasting and control, which are vital for engineering applications such as robotics, aerospace, and automotive industries, where anticipating system behaviour under various conditions can lead to innovations in design and functionality. Traditional methods based on first principles are insufficient due to the intricacies involved or incomplete understanding of underlying mechanisms. Data-driven methods simplify deriving these complex models, especially in finding the governing equations of turbulent fluid flows where full-scale modelling is computationally expensive. Although current data-driven methods automate tuning system parameters (e.g., diffusion coefficients), they still require algorithm parameters (e.g., number of iteration times and threshold values). Consequently, using these methods presents a trade-off between the absence of fully automated algorithms requiring users to engage in manual tuning and iterative usage of semi-automated algorithms. This scenario highlights a key challenge in the field: developing automated algorithms to identify governing equations with minimal manual intervention, streamlining the process, and improving its applicability across diverse scientific domains.

Through these motivations, the identification of governing equations not only aims to improve the theoretical understanding and predictive capabilities but also strives to develop practical tools that can cope with the complexity and variability of real-world systems. Therefore, this research is a cornerstone for technological progress and operational efficiency across science and engineering disciplines.

1.3 Thesis Structure

This dissertation introduces the creation of advanced computational methods designed to automatically discover dynamic systems from data. The structure of this work systematically explores a series of research topics, ensuring a clear purpose and emphasising the innovation of the proposed approaches.

- **Chapter 2 (Literature Review):**
  This chapter reviews statistical and machine learning methods, especially regression analysis, numerical derivatives, and signal processing methodologies. All are the groundwork for the research presented herein.

- **Chapter 3 (Review of Dynamical System Identification Framework):**
  This chapter summarises the recent developments in dynamical system identification, including sparse regression and neural network approaches. Particularly, the advantages and drawbacks of one widely-used method, Sparse Identification of Nonlinear Dynamics (SINDy) framework [3], are analysed from the automating perspective.

- **Chapter 4 (Analysis of Other SINDy-based Methods):**
  This chapter is a part of the work in the published paper [1]. It reproduces and compares three popular SINDy-based methods, employing numerical tests

to show their drawbacks in terms of user-friendliness and computational resources.

- **Chapter 5 (Automatic Numerical Differentiation):**
  This chapter offers an automatic method to calculate the derivatives of data, solving a fundamental problem in dynamical system identification. It demonstrates the working processes of the new automatic method and evaluates its performances using one-dimensional signals and two-dimensional images.

- **Chapter 6 (Automating the Discovery of PDEs from Data):**
  Based on the automatic numerical derivative method presented in Chapter 5, this chapter proposes a sparse regression algorithm to identify PDEs from data. This chapter also provides systematic tests to evaluate whether a new identification method is reliable or not.

- **Chapter 7 (Finding COVID-19 Transmission Dynamics):**
  This chapter employs an extension of the automatic regression for governing equations (ARGOS) framework [1], Bayesian ARGOS, to analyse real-world data, COVID-19 data in mainland China, to extract the epidemic dynamics. The results indicate that the ARGOS framework is a potential solution in the real world.

- **Chapter 8 (Conclusion):**
  This chapter summarises the discussion of all previous chapters, elucidating the future development of data-driven system identification. This research has opened up new perspectives for exploring data in engineering and has significantly contributed to the development of automated model discovery.

1.4 Notation

This thesis's notation is similar to [32–34]. As such, n represents the number of observations from the population size N, and p indicates the number of variables for prediction, also called predictors in statistics and features in machine learning.

Lowercase bold will always denote a column vector with length n:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}.$$

Normally, lowercase normal font denotes scalars, but it sometimes denotes row vectors with length m:

$$a = \begin{pmatrix} a_1 & a_2 & \cdots & a_m \end{pmatrix}.$$

If there is a situation where these two notations are conflicted, more detailed explanations will be provided. Bold capitals always designate matrices, e.g., $\mathbf{A}$. The random variables are designated by normal capital font, e.g., A, and will be specified whether the object is an $r \times s$ matrix with $\mathbf{A} \in \mathbb{R}^{r\times s}$ [33, p.11].

The design matrix with n rows and p columns is denoted as $\mathbf{X} \in \mathbb{R}^{n\times p}$, and x_{ij} means the ith observation and the jth predictor, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, p$. After expansion,

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix},$$

where the jth predictor is a column vector with length n:

$$\mathbf{x}_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}, \quad j = 1, 2, \ldots, p.$$

Furthermore, the rows of $\mathbf{X}$ as $x_1, x_2, ..., x_n$ denote x_i as a vector with length p. The

ith observation with p predictors is referred as [33, p.10; 32,34]:

$$x_i = \begin{pmatrix} x_{i1} & x_{i2} & \cdots & x_{ip} \end{pmatrix}, \quad i = 1, 2, \ldots, n.$$

Hence, $\mathbf{X}$ can also be written as

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \end{pmatrix}$$

or

$$\mathbf{X} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix}.$$

The transpose operator ^{T} mirrors a matrix along its main diagonal.

$$\mathbf{X}^T = \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1p} & x_{2p} & \cdots & x_{np} \end{pmatrix},$$

and transforms a column vector to a row vector

$$x_i^T = \begin{pmatrix} x_{i1} & x_{i2} & \cdots & x_{ip} \end{pmatrix}.$$

The notation y_i denotes the ith observation of the response variable, and the set of all n observations forms a vector

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

With this notation, y_i pairs with x_i as $\{(y_1, x_1), (y_2, x_2), \ldots, (y_n, x_n)\}$ to describe an observed dataset. The prediction of $\mathbf{y}$ is denoted as $\hat{\mathbf{y}}$, whereas $\hat{\beta}$ represents the estimation of an unknown parameter β.

Following the notations in James et al. [33] for representing the linear regression, X denotes the input or independent variable, distinguished by the subscript X_j. In the same way, Y indicates the output or the dependent variable [33, p.15; 32,34]. The residuals of the response variable explain errors between observations and predictions and are denoted by $\epsilon = Y - \hat{Y}$ [32, p.52; 33, p.61].

In modelling spatiotemporal dynamical systems, the empirical data that researchers collected is denoted as $\mathbf{U} = \mathbf{U}(t, x)$ with n points on the temporal dimension t and m points on the spatial dimension x. In this way, $\mathbf{U}$ is a matrix having n rows and m columns:

$$\mathbf{U}_{n\times m} = \begin{pmatrix} u(t_1, x_1) & u(t_1, x_2) & \cdots & u(t_1, x_m) \\ u(t_2, x_1) & u(t_2, x_2) & \cdots & u(t_2, x_m) \\ \vdots & \vdots & \ddots & \vdots \\ u(t_n, x_1) & u(t_n, x_2) & \cdots & u(t_n, x_m) \end{pmatrix}.$$

When the PDE has two spatial dimensions, the PDE solution extents to a three-dimensional tensor $\mathbf{U} = \mathbf{U}(t, x, y)$, where the temporal dimension t has n points, and the two spatial dimensions x and y have m_1 and m_2 points, respectively.

CHAPTER 2

Literature Review

2.1 Linear Regression

To find the relationship between the response variable Y and p independent variables $X_1, X_2, \dots, X_p$, the following equation can be used:

$$Y = f(X_1, X_2, \dots, X_p) + \epsilon, \tag{2.1}$$

where f is some fixed but unknown function and ϵ is the random error term, which typically has a zero mean and a variance of σ^2, and is independent of X [35, p.12; 36, p.16]. If f depends on some parameters, Eq. (2.1) is a parametric model; otherwise, it is a non-parametric model [36, chpt.2.1.2], such as Savitzy-Golay (see Section 2.5). One of the simplest assumptions: f is a linear model in X, i.e.,

$$f(X_1, X_2, \dots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p, \tag{2.2}$$

Hence,

$$Y = f(X_1, X_2, \dots, X_p) + \epsilon = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \epsilon \tag{2.3}$$

where $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are unknown parameters and β_0 is the intercept term. The parameters enter linearly in the model, while the predictors can be nonlinear [35, p.12]. For instance, $Y = \beta_0 + \beta_1 X_1 + \beta_2 \log X_2 + \beta_3 X_2 X_3 + \epsilon$ is a linear model, but $Y = \beta_0 + \beta_1 X_1^{\beta_2} + \beta_3 X_3 + \epsilon$ is not because the parameter β_2 is not linear in the model.

The matrix form of a linear model is:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon, \tag{2.4}$$

where the design matrix $\mathbf{X}$ contains the intercept term $\mathbf{1}$, and $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$ [32, p.11].

The standard way to estimate the unknown parameters β is to minimise the ℓ_2-norm squared of the residuals ϵ, ordinary least squares (OLS) estimation, or maximise the likelihood function, maximum likelihood estimation (MLE). Both methods are equivalent as long as the model residuals obey a normal distribution, i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The derivation of OLS estimates and MLE and the relationship between them are shown in the following parts.

2.1.1 Least squares estimation

First, the ith prediction of Y is denoted as $\hat{y}_i = \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i x_i$, and $\epsilon_i = y_i - \hat{y}_i$ represents the ith residual, the difference between the observed value and the predicted estimate from a regression model [33, p.61]. The most common loss function, i.e., error metric for the predictions from the regression model, is the residual sum of squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n} \epsilon_i^2 = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j \right)^2 . \tag{2.5}$$

To estimate the unknown parameter β, the gradient of the RSS with respect to β is derived:

$$\begin{aligned}\frac{d\mathrm{RSS}}{d\beta} &= \frac{d}{d\beta}(\mathbf{y}-\mathbf{X}\beta)^T(\mathbf{y}-\mathbf{X}\beta) = \frac{d}{d\beta}(\mathbf{y}^T-\beta^T\mathbf{X}^T)(\mathbf{y}-\mathbf{X}\beta)\\ &= \frac{d}{d\beta}\left(\mathbf{y}^T\mathbf{y}-2\mathbf{y}^T\mathbf{X}\beta+\beta^T\mathbf{X}^T\mathbf{X}\beta\right)\\ &= -2\mathbf{X}^T\mathbf{y}+2\mathbf{X}^T\mathbf{X}\beta.\end{aligned} \tag{2.6}$$

After setting the gradient of the loss to zero,

$$\left(\mathbf{X}^T\mathbf{X}\right)\beta = \mathbf{X}^T\mathbf{y}. \tag{2.7}$$

Under the assumption that $\mathbf{X}^T\mathbf{X}$ is a nonsingular matrix, β has the unique solution

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.8}$$

After obtaining $\hat{\beta}$, the variance of β, $Var(\beta)$, still needs to be estimated. The assumption in estimating $Var(\beta)$ is that each y_i is uncorrelated and has a constant variance σ^2 and that each x_i is not random. The variance-covariance matrix of β is

$$\begin{aligned}Var(\hat{\beta}) &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\right)^T Var(\mathbf{y})\\ &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\sigma^2\end{aligned} \tag{2.9}$$

To estimate σ^2, the hat matrix is needed [35, chpt.2.4]:

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T. \tag{2.10}$$

It can be derived that the hat matrix is symmetric

$$\mathbf{H}^T = \left(\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\right)^T = \mathbf{H}$$

and idempotent

$$\mathbf{H}\cdot\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T = \mathbf{H}.$$

Using $\mathbf{H}$ to rewrite the RSS, it can be

$$\mathrm{RSS} = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) = \mathbf{y}^T(\mathbf{1} - \mathbf{H})\mathbf{y}.$$

Suppose the design matrix $\mathbf{X}$ has n observations (rows) and p predictors (columns), the expectation of RSS can be

$$\begin{aligned}
\mathbb{E}(\mathrm{RSS}) &= \mathbb{E}(\mathbf{y}^T(\mathbf{1} - \mathbf{H})\mathbf{y}) \\
&= \sigma^2(n-p) + \mathbb{E}(\mathbf{y})^T(\mathbf{1} - \mathbf{H})\mathbb{E}(\mathbf{y}) \\
&= \sigma^2(n-p) + (\mathbf{X}\beta)^T(\mathbf{1} - \mathbf{H})(\mathbf{X}\beta) \\
&= \sigma^2(n-p) + (\mathbf{X}\beta)^T\mathbf{X}\beta - (\mathbf{X}\beta)^T\mathbf{H}\mathbf{X}\beta \\
&= \sigma^2(n-p).
\end{aligned}$$

Therefore,

$$\hat{\sigma}^2 = \frac{\mathrm{RSS}}{n-p} \tag{2.11}$$

is an unbiased estimation of σ^2 [35, chpt.2.4].

Typically, the coefficient of determination is used to quantify the goodness-of-fit:

$$R^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \tag{2.12}$$

where $\bar{y}$ is the mean value of the response variable Y, and TSS (Total Sum of Squares) describes the total variance of the observations Y. Hence, R^2 is a proportion variation in Y explained by X, ranging from zero to one – the closer to one, the better the model performs [36, p.70; 37, p.23; 38, p.12].

The OLS is the most basic estimation method of β, other least squares methods, such as penalised least squares, will be illustrated in Section 2.4.

2.1.2 Maximum likelihood estimation (MLE)

From the probability distribution perspective, the response variable's probability density function is assumed to be $P(y_i; \psi)$, so that the joint density function, also called likelihood function, can be derived [39, p.144]:

$$L(\mathbf{y};\psi) = P(y_1, y_2, \ldots, y_n; \psi) = \prod_{i=1}^{n} P(y_i; \psi), \tag{2.13}$$

where ψ represents parameter(s), including β and ϵ. Then, another two assumptions for the linear model are required [40, p.70]:

- Equation (2.2) represents the expectation of Y, i.e.,

$$Y = \mathbb{E}(Y|X_1, X_2, \ldots, X_p) + \epsilon = \beta_0 + \sum_{j=1}^{p} X_j\beta_j + \epsilon; \tag{2.14}$$

- The residual ϵ obeys a normal distribution

$$\epsilon \sim \mathcal{N}(0,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right). \tag{2.15}$$

Hence, $y_i \sim \mathcal{N}(x_i^T\beta, \sigma^2)$, and the likelihood function of the observation is:

$$L\left(\beta, \sigma^2; \mathbf{y}\right) = \left(2\pi\sigma^2\right)^{-\frac{n}{2}} \prod_{i=1}^{n} \exp\left(-\frac{(y_i - x_i^T\beta)^2}{2\sigma^2}\right). \tag{2.16}$$

Maximising the likelihood function, Eq. (2.16) is equivalent to maximising the corresponding log-likelihood function, Eq. (2.17), which is mathematically tractable.

$$\ell_{MLE}\left(\beta, \sigma^2; \mathbf{y}\right) = -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 \tag{2.17}$$

$$\frac{\partial \ell_{MLE}}{\partial \beta} = \frac{1}{2\sigma^2}\sum_{i=1}^{n} 2x_i^T(y_i - x_i^T\beta) = 0 \tag{2.18}$$

$$\frac{\partial \ell_{MLE}}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 = 0 \tag{2.19}$$

Equation (2.18) leads to the same expression as Eq. (2.6), thus, $\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ and $Var(\hat{\beta}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\sigma^2$. However, the estimation of σ^2 from Eq. (2.19), $\hat{\sigma}^2 = \frac{RSS}{n}$, is biased because of omitting the uncertainty in the estimation of β [40, p.74]. To obtain an unbiased $\hat{\sigma}^2$, the Restricted Maximum Likelihood Estimation (RMLE) is needed:

$$\begin{aligned}\ell_{\mathrm{RMLE}}\left(\sigma^2;\mathbf{y}\right) &= -\frac{n-p}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - x_i^T\left(\sum_{i=1}^{n}x_i x_i^T\right)^{-1}\sum_{i=1}^{n}x_i y_i\right)^2 \\ &= -\frac{n-p}{2}\log(\sigma^2) - \frac{\mathrm{RSS}}{2\sigma^2} \qquad (2.20)\end{aligned}$$

In this way,

$$\frac{\partial \ell_{\mathrm{RMLE}}}{\partial \sigma^2} = \frac{n-p}{2\sigma^2} + \frac{\mathrm{RSS}}{2\sigma^4} = 0 \quad \Longleftrightarrow \quad \hat{\sigma}^2 = \frac{\mathrm{RSS}}{n-p},$$

which derives the same as Eq. (2.11).

2.1.3 Bayesian regression

Based on the MLE, the Bayesian regression introduces Bayesian probability into the linear regression to interpret the estimated parameters. Instead of assuming β and σ^2 are fixed values, from the Bayesian perspective, β and σ^2 are all probability distributions with a joint prior $P(\beta, \sigma^2)$ [41]. Bayesian regression aims to estimate the posterior distributions for β and σ^2, which requires the Bayes' theorem [42]:

$$P(\beta, \sigma^2|Y) = \frac{P(Y|\beta, \sigma^2)P(\beta, \sigma^2)}{P(Y)} \propto P(Y|\beta, \sigma^2)P(\beta, \sigma^2). \qquad (2.21)$$

As β and σ are independent events, the posterior can be written as

$$P(\beta, \sigma^2|Y) = P(\beta|Y, \sigma^2)P(\sigma^2|Y, \beta) \propto P(Y|\beta, \sigma^2)P(\beta)P(\sigma^2). \qquad (2.22)$$

Here, $P(\beta)$ and $P(\sigma^2)$ are priors determined by users.

Although both priors can be arbitrary distributions, the analytic expression for the probability density function of the posterior distributions may be difficult to derive. Normally, if priors are noninformative, β and σ can be set as uniform distributions [43, p.335]. If the mean and variance of β are known, the conjugate prior distribution, which has the same functional form as the likelihood function, can be used. In most cases, the conjugate prior of β is normal distribution, and σ^2 follows inverse-gamma (IG) distribution [43, p.43]. With these prior distributions, a Bayesian regression model can be described as:

$$\begin{aligned} y_i \mid (\mu_i, \sigma^2, \mathbf{x}_i) &\sim \mathcal{N}\left(\mu_i, \sigma^2\right), \quad i = 1, \ldots, n \\ \mu_i &= \mathbf{x}_i^T \boldsymbol{\beta} \\ \beta_j &\sim \mathcal{N}\left(\mu_{\beta_j}, \sigma_{\beta_j}^2\right), \quad j = 0, \ldots, p \\ \sigma^2 &\sim IG\left(a, b\right) \end{aligned}, \tag{2.23}$$

where $\mu_{\beta_j}, \sigma_\beta, a, b$ are determined by users based on prior information. In this way, the statistical inference methods can be applied to the posterior distributions β and σ.

As the analytical expressions of the posterior distributions are too complex to derive, direct sampling from joint distributions is difficult. The numerical sampling method, Markov chain Monte Carlo (MCMC), is always used to simulate posterior distributions [43, p.275]. Gibbs sampler, one of the popular MCMC algorithms, iteratively samples from multivariate probability distributions when sampling from conditional distributions is easier than joint distributions [43, p.276]. Another MCMC algorithm is Metropolis-Hastings, which uses a proposal distribution to generate candidate moves whose acceptance probability ensures the stationary distribution of the Markov chain is the target distribution [43, p.278].

2.1.4 Inference for estimated parameters

Statistical inference is fundamental and involves drawing conclusions about a population based on observed data, such as predictions or hypotheses. The inference for the estimated parameters contains hypothesis tests and building confidence intervals.

Under assumptions (2.14) and (2.15), the distribution of $\hat{\beta}$ is conclude:

$$\hat{\beta} \sim \mathcal{N}(\beta, (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2) \tag{2.24}$$

where $\mathcal{N}$ is a multivariate normal distribution. Also, due to the χ^2 distribution's definition, $\hat{\sigma}^2$ obeys a χ^2 distribution with $n-p$ degrees of freedom:

$$\hat{\sigma}^2 \sim \frac{\sigma^2}{n-p}\chi^2_{n-p}. \tag{2.25}$$

Additionally, $\hat{\beta}$ and $\hat{\sigma}$ are statistically independent. According to the distributional properties of both $\hat{\beta}$ and $\hat{\sigma}$, the hypothesis tests and confidence intervals for each β_j can be built.

For each $j = 1, 2, \cdots, p$, the null hypothesis is $\beta_j = 0$. The Z-score, a t distribution with $n-p-1$ degree of freedom ($\mathrm{t}_{(n-p-1)}$), is employed to perform the test [32, chpt.3.2]:

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}} \sim \mathrm{t}_{(n-p-1)} \tag{2.26}$$

where $v_j = (\mathbf{X}^T\mathbf{X})^{-1}_{jj}$ is the jth diagonal element of $(\mathbf{X}^T\mathbf{X})^{-1}$. Under the model's degree of freedom, a Z-score has its corresponding p-value. If the estimated p-value is less than the significance level α (always 0.05), the null hypothesis can be rejected.

Confidence intervals (CIs) are another important statistical tool for visualising the uncertainty of estimated parameters based on the hypothesis test. The confidence interval for β_j at the α level is:

$$(CI_{\text{low}}, \quad CI_{\text{up}}) = \Big(\hat{\beta}_j - z^{(1-\frac{\alpha}{2})}SE(\hat{\beta}_j), \quad \hat{\beta}_j + z^{(1-\frac{\alpha}{2})}SE(\hat{\beta}_j)\Big) \tag{2.27}$$

where $z^{(1-\frac{\alpha}{2})}$ is the percentile of the $\mathrm{t}_{(n-p-1)}$ distribution, and $SE(\hat{\beta}_j) = \sqrt{v_j}\hat{\sigma}$ is the standard error of β_j. Note that by replacing the estimated $\hat{\sigma}$ with a known σ, z_j has a standard normal distribution. As n increases, the difference in tail quantiles between a $\mathrm{t}_{(n-p-1)}$ distribution and a standard normal distribution will decrease. For easier computation, the $\mathrm{t}_{(n-p-1)}$ distribution can be approximated by the normal distribution [32, chpt.3.2]. For example, $z^{(1-\frac{\alpha}{2})} \approx 1.96$ at $\alpha = 0.05$ can be used to calculate the confidence interval. However, the null hypothesis can be rejected if zero is not located in the confidence interval, i.e., $0 > CI_{\text{up}}$ or $0 < CI_{\text{low}}$.

2.1.5 Multicollinearity

When analysing real-world data, a multiple regression model may have collinearity or multicollinearity, which refers to linear relationships between predictors [38, p.109]. The term multicollinearity is for three or more correlated predictors, whereas collinearity occurs when only two correlated predictors are present.

The extreme case of collinearity is the linear combination between predictors (e.g., $x_1 = 2x_2$), which leads to the singular matrix $\mathbf{X}^T\mathbf{X}$ and no unique least squares estimate [37, p.106]. In most cases, the challenge arises when $\mathbf{X}^T\mathbf{X}$ tends to be singular. As $\hat{\beta}$ and $Var(\hat{\beta})$ are calculated by $(\mathbf{X}^T\mathbf{X})^{-1}$, the estimate of $\hat{\beta}$ will be inaccurate, and $Var(\hat{\beta})$ will inflate, which also causes a wider confidence interval and wrong decisions on t-test [38, p.110].

Multicollinearity often arises during the data collection process when observations are limited to specific physical conditions, design constraints, or sample data from a narrow subspace of independent variables. It also emerges with the expansion of variables' interactions, such as adding terms like x_1^3 or x_1x_2 to the design matrix. Furthermore, an abundance of outliers can contribute to multicollinearity [38, p.109].

Pairwise scatterplots, the variance inflation factor (VIF), and eigenvalue methods are three important techniques to detect multicollinearity [38, chpt.7.2]. Pairwise scatterplots visualize the relationships between all independent variables by inspecting all $\binom{p-1}{2}$ pairwise scatterplots for $p-1$ independent variables. This allows for the visual distinction of linear relationships between pairs of predictors. However, this method is inappropriate for a dataset with a large amount of predictors. For example, if a dataset contains 20 predictors, users need to repeat analysing similar graphs 171 times.

The VIF assesses the severity of multicollinearity by measuring the variance of the estimated regression coefficient that increases due to collinearity. The VIF of the jth predictor is defined as:

$$\text{VIF}_j = \frac{1}{1 - R_j^2}, \quad j = 1, 2, \ldots, p, \tag{2.28}$$

where R_j^2 is the coefficient of determination, Eq. (2.12), for the regression model that the jth predictor, X_j, is regressed against all other predictors [44, p.250]. Hence, $\mathrm{VIF}_j \in [1, \infty]$. When $\mathrm{VIF}_j \to \infty$, the given X_j is highly related to other predictors, while $\mathrm{VIF}_j = 1$ represents no collinearity. In practice, multicollinearity between predictors always exists. Typically, $\mathrm{VIF}_j = 1$ means there is no multicollinearity, $1 < \mathrm{VIF}_j < 5$ represents predictors have some moderate multicollinearity, and $\mathrm{VIF}_j > 5$ strongly supports the multicollinearity between predictors [44, p.250; 38, p.111].

Specifically, when there are only two predictors, both predictors have the same VIF values because of the same R_j^2 values. Suppose there is a regression model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$, the following two models are fitted with the same VIF values.

$$\begin{cases} X_1 = a + bX_2 \\ X_2 = c + dX_1 \end{cases} \Longleftrightarrow \begin{cases} X_1 = a + bX_2 \\ X_2 = -\frac{a}{b} + \frac{1}{b}X_1 \end{cases} \quad \Longrightarrow \quad R_1^2 = R_2^2 \quad \Longleftrightarrow \quad \mathrm{VIF}_1 = \mathrm{VIF}_2$$

The third popular approach employs eigenvalue methods [38, p.113]. The correlation matrix for the standardised data is denoted as $\mathbf{r} \in \mathbb{R}^{p \times p}$, reflecting the inclusion of p predictors but excluding the intercept column. Subsequently, the eigenvalues λ_j for $\mathbf{r}$ are computed and arranged in descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. The presence of multicollinearity is present if the empirical criterion $\sum_{j=1}^{p} \lambda_j^{-1} > 5p$. The condition index (number) of the data correlation matrix κ is used for measuring overall multicollinearity, extending from eigenvalue methods [44, p.252]:

$$\kappa = \sqrt{\frac{\text{The maximum eigenvalue of the correlation matrix}}{\text{The minimum eigenvalue of the correlation matrix}}} = \sqrt{\frac{\lambda_1}{\lambda_p}}. \tag{2.29}$$

Empirical evidence suggests $\kappa > 30$ illustrates the non-negligible multicollinearity in the regression model [44, p.252; 38, p.113].

Multicollinearity detrimentally impacts regression results, yet several remedies exist. A straightforward method involves excluding the problematic variables from the model. Removing variables can be implemented based on the highest VIF or a strategic subset selection (Section 2.3) to mitigate multicollinearity. Alternatively, keeping collinear predictors in the regression model necessitates the usage of ridge regression instead of variable exclusion [38, p.288][45]. Ridge regression (reviewed in Section 2.4.1) is a classical solution to multicollinearity and adjusts coefficients towards zero, thereby diminishing the standard errors of the regression model [45].

2.2 Model Selection and Assessment

Model selection and assessment refer to evaluating and choosing the most appropriate model to find the relationships between the response and predictors among a set of candidate models for a given dataset. Model selection involves selecting the form of the regression equation, the variables to include, and any interaction terms or transformations that may be necessary. Model assessment guarantees that the selected model has best balanced the trade-off between goodness-of-fit and complexity, providing accurate and robust predictions without overfitting the data.

James et al. [36] highlights three popular model selection methods: subset selection, shrinkage regression, and dimension reduction. However, these selection methods require metrics to quantify which model is the best. The following subsections discuss some metrics derived from RSS, information criteria, cross-validation, and the bootstrap technique. Subset selection will be introduced in Section 2.3, whereas shrinkage regression will be elaborated in Section 2.4. Dimension reduction is pertinent when the number of variables surpasses the number of observations, i.e., $p > n$. However, this condition is not met in dynamical system identification, where $n > p$ implies that dimension reduction is inapplicable.

2.2.1 Model evaluation metrics

Evaluating model accuracy is essential for an objective assessment of model performance in the prediction. Typically, the training set (X_{train}, Y_{train}) is employed to estimate unknown parameters (e.g., $\hat{\beta}$ in regression models), and performance metrics are calculated using the test set (X_{test}, Y_{test}).

Residual sum of squares (RSS) measures the squared differences between the observations and predictions. The definition, Eq. (2.5), indicates that RSS tends to zero when the model's predictions are close to the observations. Thus, this value should be as small as possible.

Mean Squared Error (MSE) measures the average squared difference between the observations and predictions. It is defined as:

$$\text{MSE} = \frac{\text{RSS}}{n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{2.30}$$

The same as RSS, a lower MSE indicates a more accurate prediction. The MSE is larger when the model poorly predicts [36, p.30]. Furthermore, MSE can be decomposed into the bias and variance components:

$$\begin{aligned}
\text{MSE} &= \mathbb{E}\left[y - \hat{y}\right]^2 = \mathbb{E}\left[y - \hat{f}(x)\right]^2 = \mathbb{E}\left[\left(y - f\left(x\right) + f\left(x\right) - \hat{f}(x)\right)\right]^2 \\
&= \mathbb{E}\left[\left((y - f(x))^2 + 2(y - f(x))(f(x) - \hat{f}(x)) + (f(x) - \hat{f}(x))^2\right)\right] \\
&= \mathbb{E}\left[(y - f(x))^2\right] + 2\mathbb{E}\left[(y - f(x))(f(x) - \hat{f}(x))\right] + \mathbb{E}\left[(f(x) - \hat{f}(x))^2\right] \\
&= Var(\epsilon) + 0 + \mathbb{E}\left[f^2(x) - 2f(x)\hat{f}(x) + \hat{f}^2(x)\right] \\
&= Var(\epsilon) + \mathbb{E}\left[f^2(x)\right] - 2\mathbb{E}\left[f(x)\hat{f}(x)\right] + \mathbb{E}\left[\hat{f}^2(x)\right] \\
&= Var(\epsilon) + f^2(x) - 2f(x)\mathbb{E}\left[\hat{f}(x)\right] + Var(\hat{f}(x)) + \mathbb{E}\left[\hat{f}(x)\right]^2 \\
&= Var(\epsilon) + Var(\hat{f}(x)) + \left[\mathbb{E}\left[\hat{f}(x)\right] - f(x)\right]^2 \\
&= Var(\epsilon) + Var(\hat{f}(x)) + Bias^2\left(\hat{f}(x)\right) \\
&= Var(\epsilon) + Var(\hat{y}) + Bias^2\left(\hat{y}\right).
\end{aligned}$$

MSE calculated by the training set provides insights into the model's performance on historical observations. However, the primary interest lies in the model's efficacy on a new dataset, referred to as the test set in statistical learning [36, p.30]. Consequently, given an adequate number of observations in the test set, the model's MSE is determined by the average squared prediction error for the test dataset:

$$\frac{\left(Y_{test} - \hat{Y}_{test}\right)^2}{n_{test}}, \tag{2.31}$$

and the lowest MSE on test data characterises the optimal model. Furthermore, an overfitted model always exhibits a low MSE for training data but a high MSE for test data. This discrepancy often arises when the model excessively captures patterns attributable to random fluctuations rather than the "true properties of the unknown function f" [33, p.32]. To mitigate overfitting, the selection of the model should also incorporate the MSE on test data.

When evaluating various models regressed on the same dataset, the **Root Mean Squared Error** (RMSE) is preferred because its unit is consistent with that of the response variable, facilitating direct interpretation under the same measure.

$$\text{RMSE} = \sqrt{\frac{\text{RSS}}{n}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.32}$$

The **coefficient of determination**, R^2, is another metric that has been shown in Eq. (2.12).

2.2.2 Information criteria

The inclusion of additional predictors in a regression model enhances its accuracy but may lead to overfitting [36, p.210]. To mitigate overfitting, evaluating the impact of the number of predictors is crucial. For instance, with a dataset comprising 10 predictors, the most explanatory model may require only five predictors, even though the most predictive model is the full model. In this way, the information criteria are necessary to achieve a balance between model accuracy and complexity.

Akaike information criterion (AIC) is an in-sample-fit technique for assessing the quality of statistical models [46]. The optimal model should have the lowest AIC within the candidate models. The definition of AIC is based on the maximum likelihood and a penalty:

$$\text{AIC} = -2\log(\hat{L}) + 2d, \tag{2.33}$$

where $\hat{L}$ is the likelihood of the estimated model, and d is the number of parameters for the estimated model. In a linear regression model with p predictors, $d = p + 1$. In shrinkage regression, d is the number of nonzero parameters for the estimated model [47].

The likelihood function $\hat{L}$ assesses the goodness-of-fit, while the penalty $2p$, an increasing function of the number of estimated parameters, discourages overfitting and guarantees simplicity. Therefore, adding more predictors to the model decreases $-2\log(\hat{L})$ but increases $2p$. If AIC reaches its optimal value, incorporating further predictors makes $2p$ dominant in the AIC calculation, thereby increasing its value.

Moreover, when assessing a linear model with Gaussian errors, the maximum likelihood is equivalent to least squares [36, p.212; 38, p.255], transforming Eq. (2.33) to

$$\mathrm{AIC} = n\log(\mathrm{RSS}) + 2d + C, \tag{2.34}$$

where n is the number of observations, and C is a constant value.

Schwarz [48] derived **Bayesian information criterion** (BIC) from the Bayesian perspective to construct a method to assess statistical models. BIC resembles AIC but imposes a more substantial penalty on the number of estimated parameters, favouring a more parsimonious model [36, p.212; 38, p.255]. BIC is calculated as follows:

$$\mathrm{BIC} = -2\log(\hat{L}) + \log(n)d. \tag{2.35}$$

Thus, when $n > 7$, BIC applies a heavier penalty on the estimated parameters compared to AIC. Similar to AIC, the selection of the optimal model using BIC is also based on achieving the lowest BIC value. If the residuals of the model obey normal distribution, BIC becomes

$$\mathrm{BIC} = n\log(\mathrm{RSS}) + \log(n)d + C. \tag{2.36}$$

AIC and BIC may select the same optimal model due to their similarity, but they are suitable under different conditions. Given the true model is contained in the candidate models, BIC always selects the correct one as the number of observations n tends to infinity [32, p.235], while AIC does not fit this property. However, when n is finite or the true model does not fall in the candidate models, AIC outperforms BIC and serves to select the model that best approximates the true model [49–51].

The **adjusted coefficient of determination** (adjusted R^2) is another useful method for evaluating models with different predictors. The R^2, Eq. (2.12), rises with the addition of predictors to the model due to the consequent reduction in the RSS. For a linear regression model with d predictors, the adjusted R^2 is proposed to address this problem:

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-p-1)}{\text{TSS}/(n-1)}. \tag{2.37}$$

Unlike AIC and BIC, the best model has the highest adjusted R^2 value. Once the true model is found, although adding extra predictors leads to a decrease in RSS, the $\text{RSS}/(n-p-1)$ increases, and the adjusted R^2 will decrease eventually. Therefore, the model exhibiting the maximum adjusted R^2 effectively only contains correct predictors, excluding redundant variables [36, p.213].

2.2.3 Cross-validation

Cross-validation (CV) is the most common sampling technique for assessing model performance. When a machine learning method depends on hyperparameters, it necessitates dividing the entire dataset into a training set ($\mathcal{T}$), a validation set ($\mathcal{V}$), and a test set ($\mathcal{S}$). In contrast, for methods not reliant on hyperparameters, such as the linear regression model, dividing only the training set and the test set is sufficient. The training set facilitates the estimation of model parameters, whereas the validation set is instrumental in determining the optimal hyperparameters, ensuring the model exhibits the minimum value of the loss function. Commonly, MSE serves as the loss function, necessitating the computation of $\text{MSE}(\mathcal{T}, \mathcal{V})$. The test set evaluates the goodness-of-fit of the trained model with the optimised hyperparameters through the calculation of MSE between $\mathcal{T}_{best}$ and $\mathcal{S}$. Figure 2.1 illustrates the relationships of these three sets.

The division of data into training, validation, and test sets depends on the size of the training sample and the complexity of the models being fitted to the data. Ideally, the dataset would be sufficiently large to allocate 50% for training purposes and 25% each for validation and testing purposes [32, p.222]. However, the scarcity of extensive datasets in real-world applications commonly necessitates a compromise,

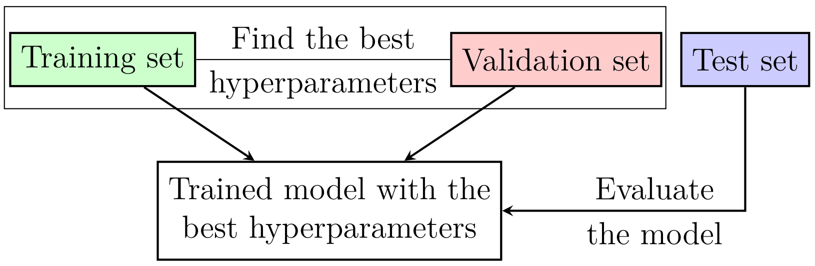


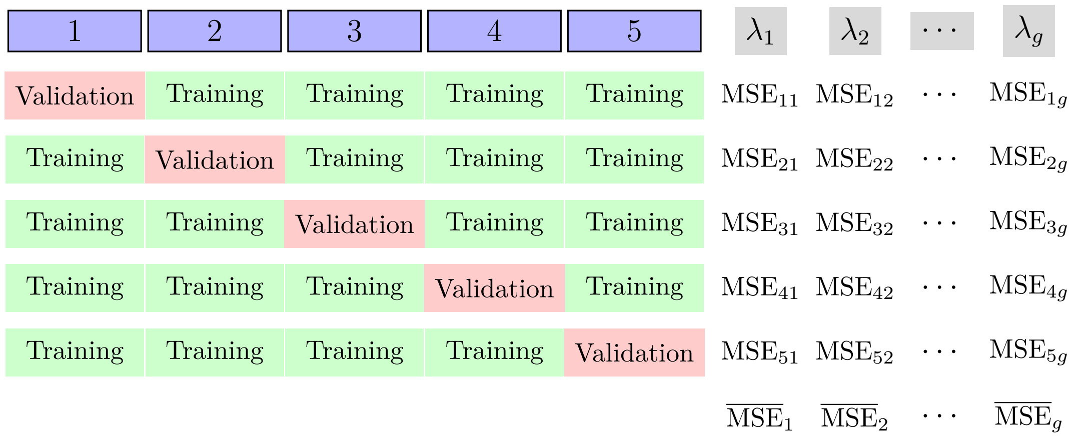
Figure 2.1: The relationships between the training set, validation set, and test set. Training and validation sets are used to find the best hyperparameters for the model. The trained model with the best hyperparameters can be evaluated by the test set.

leading to a 50/50 split between only the training and validation sets.

Two significant limitations are associated with the validation set method. Firstly, the random sampling of the validation set can lead to a high variability in the observed test error rate, relying on the specific observations included in each dataset. Additionally, the validation method uses fewer observations to train and test models, leading to the validation error rate overestimating the test error rate for the fit of the model to the whole dataset. Therefore, due to the small size of the dataset, one may lack confidence in the accuracy of the predictive model. Furthermore, the validation set approach may be doubtful if the distribution of test and training data is significantly different, potentially resulting in overfitting the validation set and poorly generalising to the unseen dataset. Consequently, alternative methods with different validation sampling strategies are typically preferred to address these concerns.

After sampling the test set, the k-fold cross-validation (see Fig. 2.2) divides the rest of the data into k subsets of approximately equal size. In this process, the first subset serves as the validation set, and the model is trained using the remaining $k-1$ subsets. Following training, the $\overline{\mathrm{MSE}}_1$ in Fig. 2.2 for the validation set is calculated. This procedure is iterated k times, with each subset assuming the role of the validation set once, and a validation error is computed for each iteration. As a result, k validation errors, denoted as $\overline{\mathrm{MSE}}_1, \overline{\mathrm{MSE}}_2, \cdots, \overline{\mathrm{MSE}}_k$, are estimated. Therefore, the error of the model in k-fold cross-validation is determined by averaging these MSE values.

An alternative to k-fold cross-validation is leave-one-out cross-validation (LOOCV).



1
2
3
4
5
λ_1
λ_2
$\cdots$
λ_g
Validation Training Training Training Training MSE_{11} MSE_{12} $\cdots$ MSE_{1g}
Training Validation Training Training Training MSE_{21} MSE_{22} $\cdots$ MSE_{2g}
Training Training Validation Training Training MSE_{31} MSE_{32} $\cdots$ MSE_{3g}
Training Training Training Validation Training MSE_{41} MSE_{42} $\cdots$ MSE_{4g}
Training Training Training Training Validation MSE_{51} MSE_{52} $\cdots$ MSE_{5g}
$\overline{\mathrm{MSE}}_1$ $\overline{\mathrm{MSE}}_2$ $\cdots$ $\overline{\mathrm{MSE}}_g$

$$\overline{\mathrm{MSE}}_j = \frac{1}{5}\sum_{i=1}^{5} \mathrm{MSE}_{ij}; \quad j = 1, 2, \ldots, g$$

Figure 2.2: The diagram of using 5-fold cross-validation to find the optimal hyper-parameter, λ. Each validation set gives an MSE_{ij} for one λ_j. The optimal λ has the minimum $\overline{\mathrm{MSE}}_j$, $\min\{\overline{\mathrm{MSE}}_1, \overline{\mathrm{MSE}}_2, \cdots, \overline{\mathrm{MSE}}_g\}$.

Rather than creating subsets with similar sizes, it only uses a single observation (x_1, y_1) as the validation set, while the remaining observations $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ are employed as the training set. As LOOCV fits the model to $n - 1$ training observations and evaluates against the excluded observation, it exhibits an approximation to the unbiased estimate of the test error. However, relying on a single observation renders the MSE estimate highly variable. To address this, the procedure is iterated n times to compute an average MSE from n test error estimates, which makes LOOCV more computationally expensive when data size increases [33, p.200].

2.2.4 The bootstrap

The bootstrap is a resampling technique that is widely employed to estimate distributions for the unknown parameters [32, p.249; 33, p.209]. B. Efron [52] proposed this method, Efron and Tibshirani [53], Davison and Hinkley [54] generalised it with more applications. Suited for multivariate systems analysis, it employs random sampling with equal probability and replacement to establish an empirical parameter distribution(s) [53, 55, 56]. This method involves generating new samples from the original dataset, with "replacement" meaning the possibility of selecting the same observation multiple times for inclusion in a bootstrap dataset. In this way, some data points will appear multiple times in the dataset, while some may not be sampled. Additionally, each bootstrap sample maintains the same number of observations as present in the original dataset. Therefore, each sampled bootstrap dataset can fully represent the observations.

Here is a basic process for implementing bootstrap:

1. Sample data from original data with replacement;
2. Construct a model and estimate unknown parameters (e.g., coefficients in linear regression);
3. Repeat the previous steps with B times (B is the bootstrap samples);
4. Estimate statistical features, such as bootstrap means and standard deviations, for the unknown parameters.

The bootstrap is useful for statistical inference. For example, in taking the inference for the standard error of a linear regression model, B standard errors can be estimated to compose the empirical distribution, enabling users to calculate the mean values, standard deviations, and confidence intervals. However, when $B < 50$, the overfitting problem may be caused by an overlap in observations between training and test datasets, making the estimated standard error unreliable. As B increases to larger than 200, the bootstrapped standard error can be robust [57, p.215; 53, p.52; 32, p.250]. Furthermore, when estimating the robust bootstrap confidence intervals for model selection, B is recommended for larger than 2000 [57, p.205; 53, p.52].

2.3 Subset Selection

Subset selection involves identifying the most relevant predictors that contribute to explaining the variation in the response variable [33, p.226]. As the selected model is one of the sub-models, Hastie et al. [32] and James et al. [36] call this method subset selection. From the perspective of statistical inference, these relevant predictors are always statistically significant in the hypothesis test, i.e., β_j rejects the null hypothesis ($\beta_j = 0$) in the t-test. On the contrary, if a predictor cannot reject its null hypothesis, it is statistically insignificant and can be removed from the design matrix [44, p.308]. The subset selection method uses stepwise selection with statistical indexes to measure the model accuracy, including bias and variance. Since the full model has p predictors, the number of predictors of the selected model is less than p. Subset selection is particularly useful in scenarios where an interpretable regression model with a few predictors needs to be found from massive potential predictors. This parsimonious model is always robust, avoiding overfitting and multicollinearity.

Backward stepwise selection, also called backward elimination, initiates with the full model and then drops the most insignificant predictor, identified by the highest p-value in the t-tests, at each step [37, p.151]. Typically, this process iterates through all predictors, beginning with the full model and stopping with the null model. After that, one of the criteria, such as AIC, BIC, cross-validated prediction error, or adjusted R^2, is employed to identify the optimal model [32, p.60]. Backward elimination conducts the fitting of p regression models but necessitates the execution of the t-test at each step. Algorithm 1 illustrates the procedure of backward elimination.

Algorithm 1: Backward stepwise selection

```
1. Denote $\mathcal{M}_p$ as the full model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   (a) Do the linear regression to the $\mathcal{M}_k$ and calculate the $p$-values for all
       estimated coefficients.
   (b) Remove the most insignificant predictor which has the largest $p$-value, and
       call it $\mathcal{M}_{k-1}$.
   End
3. Select the best model among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ by cross-validated prediction error,
   AIC, BIC, or adjusted $R^2$.
```

Conversely, forward stepwise selection mirrors backward stepwise selection but inverses the method of determining submodels. This approach starts with the null model and gradually incorporates the most significant predictor at each phase until all p predictors are included. Hybrid stepwise selection integrates both forward and backward stepwise methods, evolving from the null model towards the full model. It employs forward stepwise for adding predictors, simultaneously considering the application of backward elimination at each juncture.

2.4 Shrinkage Methods

As mentioned in Section 2.2.1, shrinkage regression is important in machine learning and statistics that can select best-prediction models and remove multicollinearity from the regression model. It uses the ℓ_q-norm to regularise or constrain coefficient estimates, which will be shrunk towards or to zero. When $q = 2$, the shrinkage regression gives ridge regression; when $q = 1$, it provides the least absolute shrinkage and selection operator (lasso) solution; when $q = 0$, the regularisation function counts the nonzero estimates and corresponds to the subset selection [32, p.72]. In addition, many methods, such as cross-validation and the Pareto curve, can be used to find the optimal regularisation parameter of the ℓ_q-norm [58]. Cross-validation is widely used because it makes no theatrical assumption [32, 36].

2.4.1 Ridge regression

When the predictors of the model are multicollinear, $\mathbf{X}^T\mathbf{X}$ in Eq. (2.6) is a singular (not full rank) matrix, indicating that its inverse matrix does not exist. This issue is overcome by ridge regression [45] by minimising the ℓ_2 regularised RSS:

$$\hat{\beta}_{\text{ridge}} = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} \beta_j^2, \tag{2.38}$$

where λ is a nonnegative shrinkage parameter, controlling the amount of shrinkage applied to the coefficients of the predictors. The matrix form of the Eq. (2.38),

$$\hat{\beta}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y}, \tag{2.39}$$

illustrates that $\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)$ is always invertible when $\lambda > 0$. Thus, ridge regression always provides a solution, even if $\mathbf{X}^T\mathbf{X}$ is not full rank. When $\lambda = 0$, it simplifies to the OLS estimate. However, ridge estimation is biased:

$$\begin{aligned}
\text{Bias}\left(\hat{\beta}_{\text{ridge}}\right) &= \mathbb{E}\left(\hat{\beta}_{\text{ridge}} - \beta\right) = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} - \beta \\
&= \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{X}\beta - \beta \\
&= \left(\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{X} - \mathbf{I}\right)\beta \neq 0 \text{ for } \lambda > 0.
\end{aligned}$$

Therefore, a trade-off between the OLS estimate and ridge regression needs to be considered.

2.4.2 The lasso

The least absolute shrinkage and selection operator (lasso) is another shrinkage method, which adds ℓ_1 regularisation to the RSS [59], rather than ℓ_2 regularisation. The lasso solves the ℓ_1-norm problem:

$$\hat{\beta}_{\text{lasso}} = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=0}^{p} |\beta_j|, \tag{2.40}$$

where λ is also a nonnegative shrinkage parameter, affecting the same as that in Eq. (2.38). Both the lasso and ridge contain minimising the RSS problem. However, the lasso uses the ℓ_1-norm to penalise coefficients to be exactly equal to zero with a sufficiently large λ, while ridge shrinks some coefficients towards zero via the ℓ_2-norm [36, p.219; 60, p.282]. Hence, instead of subset selection (Section 2.3), the lasso is a sparse regression algorithm, yielding sparse models (submodels) by automatically performing the variable selection with different λ [36, p.219].

Computationally, unlike ridge regression where $\lambda \in (0, +\infty)$, there is a $\lambda_{max} < +\infty$ for the lasso that when $\lambda > \lambda_{max}$, all elements in $\hat{\beta}_{\text{lasso}}$ are penalised to zero (the null model). It has been proven that λ_{max} for the lasso is $\|2\mathbf{X}^T y\|_\infty$ [61].

2.4.3 The elastic net

The elastic net is proposed to solve some limitations in the lasso [62]. For example, when a group of predictors is highly correlated, the lasso only selects one of them and ignores the others. The elastic net combines the lasso and ridge penalties to overcome such an issue:

$$\begin{aligned}
\hat{\beta}_{\text{Enet}} &= \arg\min_{\beta} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} (\beta_j)^2 \\
&= \arg\min_{\beta} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} (\beta_j)^2 \right), \quad (2.41)
\end{aligned}$$

where λ is a nonnegative shrinkage parameter, and $\alpha \in [0, 1]$ adjusts the using ratio between ℓ_1 and ℓ_2 regularisation. Equation (2.41) is a general form for the elastic net [32, p.662; 38, p.291], whereas the R package `glmnet` [61] defines the elastic net as:

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \frac{1}{2} \sum_{j=1}^{p} (\beta_j)^2 \right) \right] \quad (2.42)$$

where β_0 is the intercept, which is excluded in the regularisation. The elastic net has a unique minimum since the included ℓ_2 regularisation makes the problem a convex optimisation problem. Moreover, the lasso and ridge are special cases of the elastic net for $\alpha = 1$ and $\alpha = 0$, respectively.

2.4.4 The adaptive lasso

Fan and Li [63] found that the bias of the coefficients estimated by the lasso cannot be ignored and conjectured that the lasso may not fit the oracle property: consistency in variable selection and asymptotic normality. Zou [64] first proved this conjecture and proposed the adaptive lasso that fits the oracle property with a proper λ. More details of the oracle property can refer to [63, 64]. The adaptive lasso, an advancement of the lasso method, adds adaptive weights w into the regularisation term, leading to a regularised convex optimisation solution.

The implementation of the adaptive lasso is a two-stage process. The first step estimates the adaptive weights w using

$$w = |\hat{\beta}|^{-\gamma}, \quad \gamma > 0, \quad (2.43)$$

where $\hat{\beta}$ is the model's $\sqrt{n}$-consistent estimator, such as the OLS estimator or ridge regression estimator, and γ is an exponent tuning the shape of the soft-thresholding function. In the second step, the estimated coefficient vector of the adaptive lasso $\hat{\beta}_{\text{alasso}}$ is obtained by solving the following convex optimisation problem:

$$\hat{\beta}_{\text{alasso}} = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^{p} w_j \left|\beta_j\right|, \tag{2.44}$$

where λ is a nonnegative regularisation 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 regularisation 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. As this optimisation problem is convex, it has a global minimum that can be solved efficiently [64]. If multicollinearity is not considered for computing the adaptive weights, $\hat{\beta}$ can be OLS estimates.

Computationally, the adaptive lasso problem can be transformed to the lasso problem [64]:

$$\hat{\beta}^{+} = \arg\min_{\beta} \left\| \mathbf{y} - \sum_{j=1}^{p} \frac{\mathbf{X}_j}{w_j}\beta_j \right\|_2^2 + \lambda \sum_{j=1}^{p} \left|\beta_j\right|, \quad \hat{\beta}_{\text{alasso}_j} = \frac{\hat{\beta}_j^{+}}{w_j}. \tag{2.45}$$

Therefore, algorithms for solving the lasso can be used to solve the adaptive lasso. Similar to the lasso, the adaptive lasso also has the $\lambda_{max} = \left\|2\mathbf{X}^{+T}y\right\|_{\infty}$ where $\mathbf{X}_j^{+} = \mathbf{X}_j/w_j$ that for all $\lambda > \lambda_{max}$ the adaptive lasso will eliminate all predictors, penalising all coefficients to zero.

The multi-step adaptive lasso (MSAL), Eq. (2.46), iterates the adaptive weights $\hat{w}^{(k)}$ based on the estimations from the last step, selecting fewer variables than the ℓ_0-norm at each step [65].

$$\begin{aligned} \hat{\beta}_{MSAL} &= \arg\min_{\beta} \|\mathbf{X}\beta - \mathbf{y}\|_2^2 + \lambda^{*(k)} \sum_{j=1}^{p} \hat{w}_j^{(k-1)} \left|\beta_j\right|, \\ \hat{w}_j^{(k)} &= \frac{1}{\left|\hat{\beta}_j^{(k-1)}(\lambda^{*(k-1)})\right|}, \quad j = 1, 2, \ldots, p, \quad k = 1, 2, \ldots M. \end{aligned} \tag{2.46}$$

where $\lambda^{*(k)}$ is the optimal λ at the kth iteration.

2.4.5 The adaptive elastic net

The adaptive lasso still risks poor performance when predictors are highly correlated even if they are independent [66]. Similar to the proposing of the elastic net, Zou and Zhang [66] proposed the adaptive elastic net by adding the ℓ_2 regularisation to solve the multicollinearity for the adaptive lasso. The adaptive elastic net combines the adaptive lasso and ridge regression, fitting the oracle property under weak regularity conditions and performing better than other oracle-like methods in solving multicollinearity. The formula of the adaptive elastic net is [66]:

$$\hat{\beta}_{aEnet} = \left(1 + \frac{\lambda_2}{n}\right) \left\{ \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_2 \|\beta\|_2^2 + \lambda_1^* \sum_{j=1}^{p} \hat{w}_j |\beta_j| \right\}. \tag{2.47}$$

Similarly, the adaptive weights in Eq. (2.47) have the same effect as that in Eq. (2.43).

2.4.6 The Pareto curve

The Pareto curve shows the optimal trade-off curve between the model residuals (accuracy) and the penalty (complexity) on a log-log scale [67, 68]. Although cross-validation (Section 2.2.3) is widely used in machine learning to find the optimal parameter, Cortiella et al. [31] have shown that it finds a λ optimised for prediction, potentially overfitting the true underlying equation with extra features when identifying dynamical systems. Figure 2.3 shows two Pareto curves for the lasso and the adaptive lasso, respectively. The use of the Pareto curve for determining the optimal λ was initially introduced to solve ℓ_2-norm problems [67], where the Pareto curve is also referred to as the L-curve due to its characteristic "L" shape. On the Pareto curve, the x-axis represents the norm of the model residual, $\|\mathbf{X}\beta - y\|_2$, while the y-axis accounts for the penalty, $R(\beta)$. In the lasso, $R(\beta)$ is defined as $\|\beta\|_1$, and for the adaptive lasso, $R(\beta)$ is given by $\|w^T\beta\|_1$.

The purpose of using the Pareto curve is to find the optimal balance between accuracy and complexity. The optimal point should have high curvature where slight

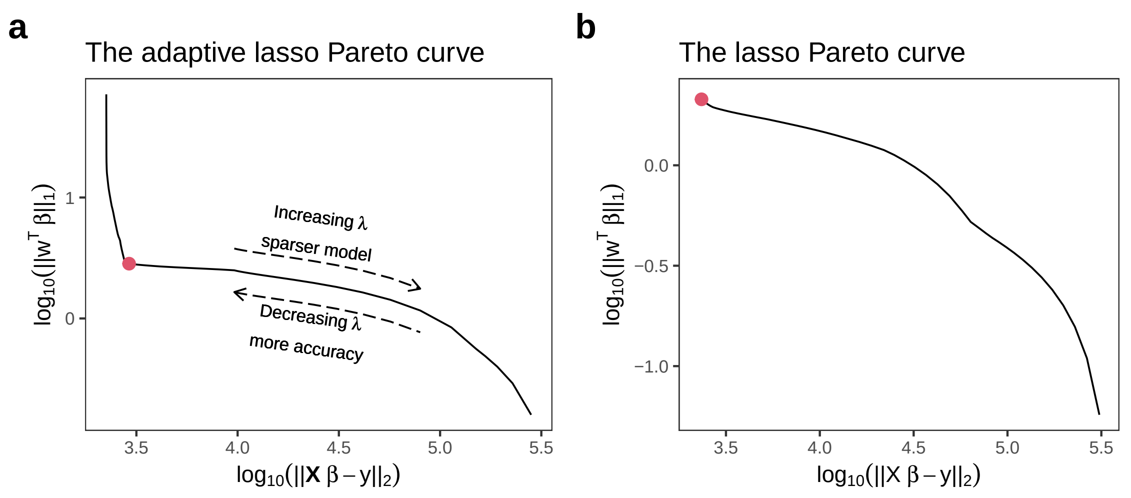


Figure 2.3: Pareto curve of the adaptive lasso and the lasso for a sampled dataset of Navier-Stokes system with SNR = 26 dB (see Section 6.2.2). The Pareto curve balances the trade-off between sparsity and goodness-of-fit. The red points on both curves indicate the optimal value of the regularisation 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.

improvements in one goal require significant sacrifices in another [69]. This optimal point typically offers a good compromise solution, corresponding to the knee (corner) of the L-curve [67, 68, 70]. It should be noticed that in the adaptive ℓ_1 regularization problems, the knee of the Pareto curve might not be distinctly visible, but it still exists [31, 68], indicated by the red point in Fig. 2.3.

The Pareto curve for the ℓ_1-norm is convex, continuously differentiable and non-increasing with $\lambda \in (\lambda_{min}, \lambda_{max})$ [31, 71]. The λ_{max} is $\left\|2\mathbf{X}^T y\right\|_\infty$ for the lasso and is $\left\|2\mathbf{X}^{+T} y\right\|_\infty$, where $\mathbf{X}_j^+ = \mathbf{X}_j / w_j$, $j = 1, 2, \dots, p$, for the adaptive lasso. When the $\lambda_{min} = 0$, it is no longer a regularisation problem, yielding OLS estimates. Hence, in `glmnet` package [61], $\lambda_{min} = \max\left(10^{-7}, \lambda_{max}/10^5\right)$.

In the following analysis (see Chapter 6), λ is treated as a continuous variable [67] that identifies the optimal point as possessing the maximum curvature on the curve. Consequently, the algorithm proposed by [70] can be employed to find the optimal λ. This numerical algorithm determines the λ associated with the maximal positive curvature on the Pareto curve through the usage of the Menger curvature of a circumcircle and the golden section search method, offering an efficient solution.

2.5 Numerical Differentiation and Data Smooth

In most cases, the mathematical formulas of collected data are unknown. Rather than using symbolic differentiation, numerical differentiation methods are needed to approximate the derivative of collected data. Numerical differentiation includes finite difference [72, p.3-11; 73, p.5-18], spectral derivative [73, p.43-65], polynomial interpolation differentiation [73, p.79-108], and Tikhonov differentiation [74].

2.5.1 Noiseless data

In the calculation of derivatives for noiseless data, the finite differences are recommended [4, 75]. Typically, it uses the central approximation to calculate derivatives up to the third-order, Eq. (2.48a) to Eq. (2.48c). Based on this, the finite differences can derive more higher-order derivatives [72, p.9].

$$\frac{d}{dx}f(x) = f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{2.48a}$$

$$\frac{d^2}{dx^2}f(x) = f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{2.48b}$$

$$\frac{d^3}{dx^3}f(x) = f'''(x) \approx \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3} \tag{2.48c}$$

2.5.2 Noisy data smoothing

Data collection invariably results in noisy data, attributed to measurement inaccuracies that encompass both systematic and random errors. The process of numerically differentiating such noisy data presents unexpected difficulties. For example, using the finite differences to deal with noisy data tends to result in the noise dominating the outcomes of numerical differentiation [75, 76]. This noise is an unexpected signal that always has sharp fluctuations. As dynamical systems are typically governed by well-behaved equations and evolve regularly with time, their data are assumed to change slowly. In signal processing, this kind of data is assumed to operate primarily at low-frequency ranges, while additive noise has high frequencies and spreads out over a much wider range than the noiseless data in the frequency domain. To address the noisy data, smoothing methods (also called lowpass filters in signal processing) are necessary to remove these unexpected high frequencies [77, p.164-175].

Typically, the noise in the data is measured by the signal-to-noise ratio (SNR), defined as

$$\mathrm{SNR} = 20 \log_{10}\left(\frac{\sigma_x}{\sigma_z}\right), \tag{2.49}$$

where σ_x is the standard deviation of the data, and σ_z is the standard deviation of the artificial noise. When creating noisy data, the standard deviation of the additive noise is

$$\sigma_z = \sigma_x \cdot 10^{-\mathrm{SNR}/20}, \tag{2.50}$$

where a smaller SNR represents a higher noise level. When SNR closes to the infinity, σ_z tends to zero.

The usage of different smoothing methods depends on the data dimensions. Commonly, data is stored as a one-dimension vector, two-dimension matrix, or higher-dimension tensor. Vector-based smoothing methods can filter data regardless of the number of dimensions. For example, when smoothing a matrix, a vector-based method can filter row by row or column by column. These methods include moving average, Savitzky-Golay filter [78], and total variation regularisation (TVR) [79]. Matrix-based methods, such as singular value decomposition (SVD) and principal component analysis reconstruction (PCAR), can only smooth matrix data, which is not suitable for a vector and leaves an unfiltered dimension for a three-dimensional tensor. Gaussian blur, or lowpass Gaussian filter, is a kernel-based convolution method for smoothing the data contaminated by random noise. Note that the Gaussian function produces the Gaussian kernel. In most cases, noise is assumed to obey a normal distribution with mean zero, making the Gaussian blur an appropriate way to remove Gaussian noise [77, p.161]. Unlike SVD and PCAR, which can only be used in two-dimensional data, Gaussian blur is not limited by dimension and only requires that the kernel has the same dimension as the data.

The default one-dimensional (1D) Gaussian kernel in OpenCV [80] can be generated from a normal distribution probability density function,

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{2.51}$$

with mean zero and the standard deviation $\sigma = 0.3 \cdot ((\text{size} - 1) \cdot 0.5 - 1) + 0.8$. Hence, the simplest Gaussian kernel with size = 3 is

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}, \tag{2.52}$$

where all elements in the kernel are probabilities. The two-dimensional (2D) simplest Gaussian kernel with size 3×3 is

$$\frac{1}{16}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \tag{2.53}$$

and the three-dimensional (3D) version is

$$\frac{1}{64}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \frac{1}{64}\left[\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}\right]. \tag{2.54}$$

The coefficients $\frac{1}{4}$, $\frac{1}{16}$ and $\frac{1}{64}$ ensure kernels are probability densities, and the sum of all elements in each kernel is equal to one.

Kernel-based methods are non-parametric methods that employ convolution between raw data and the kernel to smooth data. Hence, when the kernel is decided, the results are never changed. Figure 2.4 shows how convolution works in two-dimension data. However, parametric methods tune parameters to get different results. For example, the regularisation parameter in TVR controls the smoothness of the filtered data.

The lowpass filter, box filter, and Gaussian blur are two widely used filters. Gonzalez and Woods [77] compared both filters in Example 3.13, Chapter 3.4 of their book *Digital Image Processing*. They conclude that a Gaussian blur with a

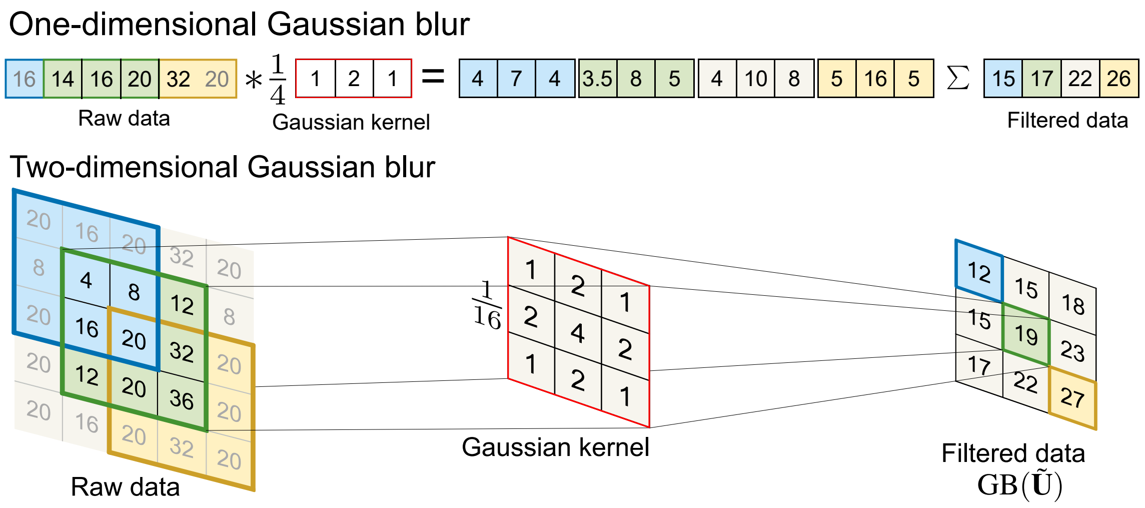


Figure 2.4: Process of Gaussian blur for data with both one and two dimensions. The observed data are in black numbers, while grey numbers are used to calculate boundaries. The red rectangle is the simple Gaussian kernel with a size 3 in one-dimensional data and 3×3 in two-dimensional data. The window in the original data shifts with different colours.

Gaussian kernel can produce smoother results than a box filter with the same kernel size. The box filter is visually acceptable, while the Gaussian kernel is more suitable for data analysis.

When analysing the power spectrum of noisy data, the spectral properties typically show a broad, irregular pattern with numerous peaks and valleys, reflecting the presence of random frequencies superimposed on the original signal. After applying smoothing filters, the signal spectrum becomes more streamlined and focused, with pronounced peaks corresponding to the fundamental frequencies of the signal becoming clearer. Focusing on smoothing in the frequency domain, the Fourier spectral smoothing method, Butterworth filter, and Chebyshev filters can be considered [81]. The Fourier spectral smoothing method applies the Fast Fourier Transform (FFT) to shift data from the time domain to the frequency domain. Hence, noise can be cut off at specific frequencies, and the data can then be returned to the time domain through the inverse FFT. The Butterworth filter [81, p.504-512] has a monotonic response with a maximally flat filter (i.e., has no ripples) in the passband and rolls slowly towards zero in the stopband. Two hyperparameters are necessary for its operation: the order of the filter and the cutoff frequency. Chebyshev filters [81, p.512-525] are derived from the Chebyshev polynomials and have a steeper roll-off than Butterworth filters. In addition to the order number and cutoff frequency, the ripple factor is also required in Chebyshev filters. Although filter hyperparameters can be fixed in a system, they are different when the system changes. Given the substantial expertise required to properly adjust these three methods' hyperparameters, their utilisation is not pursued in the following analysis.

2.5.3 Noisy data differentiation

Numerical differentiation contains the finite difference, Savitzky-Golay filter, spectral method, Tikhonov regularisation, smoothing spline, and total variation regularisation. Among these, finite differences and spectral methods are commonly employed but exhibit heightened sensitivity to noise, which can disproportionately influence the calculation of numerical derivatives [75, 76]. Among the rest of the methods (i.e., total variation regularisation, Tikhonov regularisation [74], and smooth spline) are categorised as global approaches, while the Savitzky-Golay filter and the smoother kernel are local methods. If the data sizes are large and boundary errors can be ignored, both local and global methods perform similar results [82]. However, global methods require more computational resources than local methods [76], which hinders the application of global methods to big datasets. Furthermore, some methods, such as Tikhonov differentiation [74], demonstrate accuracy for first-order derivatives but are not suitable for higher-order derivatives. On the contrary, the Savitzky-Golay filter has been distinguished as the most effective local method in derivative evaluations [4, 82] and is not limited by derivative orders. Therefore, the Savitzky-Golay filter is selected for the differentiation of noisy data in the later analysis.

The Savitzky-Golay filter applies a least squares polynomial fit over a sliding window of data points, thereby achieving simultaneous signal smoothing and differentiation. It can accurately maintain the signal's original contour while significantly diminishing noise and approximate higher-order numerical derivatives with symbolic differentiation [78, 83], and thus, widely used in many fields such as chemistry [84], signal processing [85, 86], and geophysics [87, 88]. In statistics, a similar method, locally estimated scatterplot smoothing (LOESS), focuses on accurately predicting [89, 90], while it cannot calculate derivatives. The Savitzky-Golay filter is characterised by two integer hyperparameters: the polynomial order o and the window length l. These parameters are constrained by the conditions that o must be at least 2, l should be an odd number, and $o + 1 + mod(o) \leq l \leq n - 1$ [83]. Although many methods can find the optimal window length of the Savitzky-Golay filter [84–86, 88, 91, 92], users still need to determine the polynomial order to implement these methods.

Equation (2.55a) to (2.55e) illustrate the processes of the Savitzky-Golay filter at the point i. Firstly, the Savitzky-Golay filter centres data to the local middle with the interval Δx on each sliding window, Eq. (2.55a). Next, it applies the least square estimates to fit the data by using the o-degree polynomials, Eq. (2.55b) and (2.55c). Note that the number of fitted data, the window size $2l + 1$, must be greater than degree o. Finally, the data derivatives are approximated by the symbolic differentiation of the fitted polynomial model, Eq. (2.55e). For example, a dataset with length n needs to calculate the ith first-order derivative. The o-degree polynomial fitting needs the nearest l points to construct a window, $i - l, \cdots, i, \cdots, i + l$, for which the size must be odd. The first-order derivative at the ith point (the centre point of the fitted data) is calculated symbolically by the fitted polynomial. This method gives a reliable and local smooth derivative for noisy data. The following smoothing and derivatives can be operated by shifting the window and taking the convolution with the same processes. However, points close to the data boundaries (i.e., the first l and last l points) may be biased because there are not enough data points at the boundaries.

$$z_i = \frac{x - x_i}{\Delta x}, \quad x = \{x_{i-l}, \ldots, x_i, \ldots, x_{i+l}\} \tag{2.55a}$$

$$y_i = \mathbf{Z}_i \mathbf{a} = a_0 + a_1 z_i + a_2 z_i^2 + \cdots + a_o z_i^o \tag{2.55b}$$

$$\hat{\mathbf{a}} = \left(\mathbf{Z}_i^T \mathbf{Z}_i\right)^{-1} \mathbf{Z}_i^T y_i \tag{2.55c}$$

$$\hat{y}_i = \hat{a}_0 + \hat{a}_1 z_i + \hat{a}_2 z_i^2 + \hat{a}_3 z_i^3 + \cdots + \hat{a}_o z_i^o = \hat{a}_0 \quad \text{at } z = 0,\ x = \bar{x} \tag{2.55d}$$

$$\left.\frac{d^k y}{dx^k}\right|_{y=y_i} = \frac{k!}{\Delta x^k} \hat{a}_k \quad \text{at } z = 0,\ x = \bar{x}; \quad k \in \mathbb{N}^+, k \leq d \tag{2.55e}$$

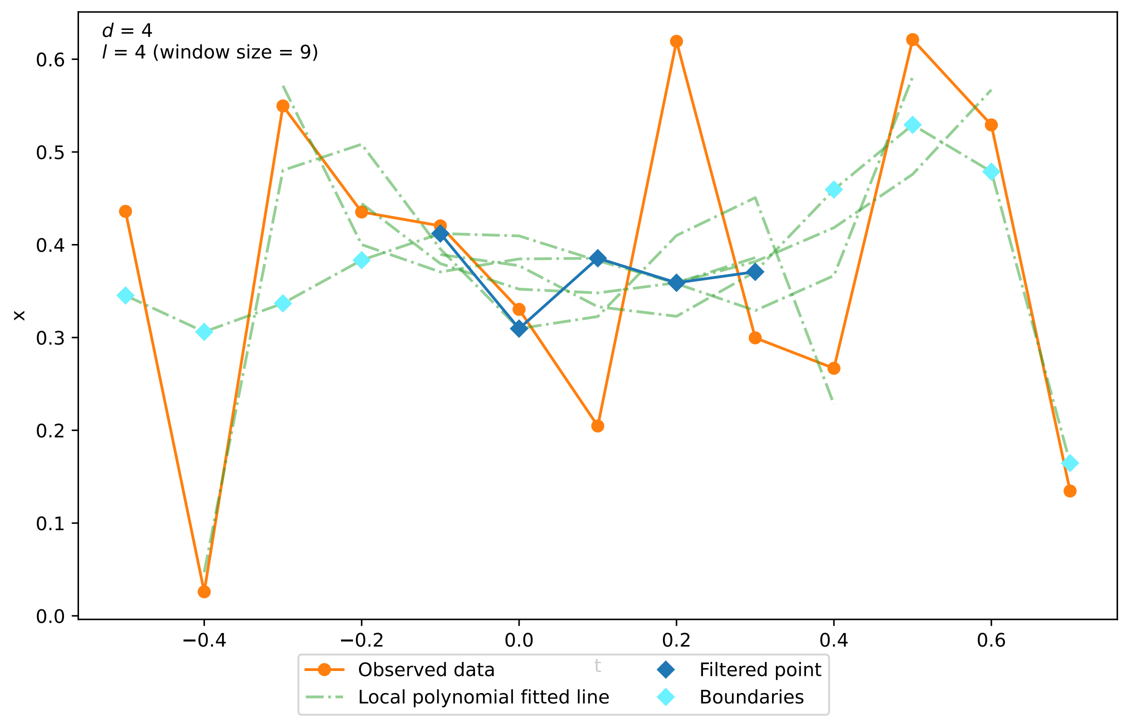


Figure 2.5: Process of applying the Savitzky-Golay filter to a signal. The orange points are the observations, the dark blue rhombus are the filtered data, the cyan rhombus are the boundary points, and the green dash lines represent the local polynomial fits for each window.

In particular, Breugel et al. [75] proposed an optimisation method, inspired by total variation regularisation, which finds the optimal pair of hyperparameters $\{o^*, l^*\}$ in `PyNumDiff` packages. Unlike other smoothing methods, the method in [75] focuses on calculating the first-order derivative $\hat{\dot{x}}$, whereas the smoothed data is calculated by trapezoidal integration of $\hat{\dot{x}}$. To calculate the second-order derivative, users need to put the first-order derivative in the same algorithm, which causes a large bias for higher-order derivatives. Additionally, there is a regularisation parameter γ in their method. Although an empirical equation, Eq. (2.56), is found that γ is related to the signal frequency $(freq)$ and the time interval (dt), this equation is based only on the data used in the paper and will differ from other data.

$$\log(\gamma) = -1.6 \log(freq) - 0.71 \log(dt) - 5.1. \tag{2.56}$$

Despite `PyNumDiff`, Python packages `scipy` and `derivative` contain the Savitzky-Golay filter. However, both `derivative` and `PyNumDiff` packages can only compute the first-order derivative. In R, such as `sgolayfilt` in `signal` package and `savgol` in `pracma` package, implement the Savitzky-Golay filter too.

2.5.4 Automatic derivative method in PyNumDiff

The automatic smoothing and derivative method shown in [75] focuses on calculating the first-order derivative of 1D data. The first-order derivative $\hat{\dot{y}}$ is estimated by

$$\hat{\dot{y}} = \dot{\mathrm{SG}}(\tilde{y}, o, l). \tag{2.57}$$

where $\dot{\mathrm{SG}}(\cdot, o, l)$ represents the first-order derivative calculated from the Savitzky-Golay filter with polynomial order o and window length l. The optimal hyperparameters set $\Phi^* = \{o^*, l^*\}$ are found by

$$\arg\max_{\Phi} \left\{ \mathrm{RMSE}(\hat{y}(\Phi), \tilde{y}) + \gamma \frac{1}{n} \sum_{i=2}^{n} |\hat{\dot{y}}_{i-1} - \hat{\dot{y}}_i| \right\}, \tag{2.58}$$

where γ is the regularisation parameter determined by the empirical equation Eq. (2.56) [75], and $\hat{y}$ is calculated by the trapezoidal integration of $\hat{\dot{y}}$:

$$\hat{y} = \mathrm{trapz}(\hat{\dot{y}}) + \frac{1}{n} \sum_{i=1}^{n} (\hat{y} - \tilde{y}). \tag{2.59}$$

Although Breugel et al. [75] only demonstrates this automatic method for one-dimensional data, it can be expanded to two-dimensional data. With the same processes in two-dimensional data, the optimal hyperparameters $\hat{\Phi}^*$ for two-dimensional data is obtained by

$$\arg\max_{\Phi} \left\{ \mathrm{RMSE}(\hat{u}(\Phi), \tilde{u}) + \gamma \max_{j} \frac{1}{n} \sum_{i=2}^{n} |\hat{u}_x(i-1, j) - \hat{u}_x(i, j)| \right\}, \tag{2.60}$$

where $\hat{u}$ is the smoothed data calculated by

$$\hat{u} = \text{trapz}\left(\begin{bmatrix} \hat{u}_x(\cdot,1) \\ \hat{u}_x(\cdot,2) \\ \vdots \\ \hat{u}_x(\cdot,m) \end{bmatrix}\right) + \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\hat{u}(i,j) - \tilde{u}(i,j)\right), \tag{2.61}$$

and

$$\hat{u}_x = \dot{\text{SG}}\left(\begin{bmatrix} \tilde{u}(\cdot,1) \\ \tilde{u}(\cdot,2) \\ \vdots \\ \tilde{u}(\cdot,m) \end{bmatrix}, o, l\right). \tag{2.62}$$

Therefore, after estimating the optimal $\{o^*, l^*\}$, Eq. (2.62) can calculate the optimal first-order derivative, and Eq. (2.61) can get the optimal smoothed data.

2.6 Summary

The following points summarise the important methods used in this thesis.

- Linear regression is a statistical method used for modelling the relationship between a response variable $\mathbf{y} \in \mathbb{R}^n$ and one or more predictors $\mathbf{X} \in \mathbb{R}^{n\times p}$. The main goal is to find coefficients β of the linear equation that best predicts the response variable based on the predictors. The estimated formula can be written as

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta},$$

- Shrinkage regression refers to a group of regression methods that modify OLS to improve model prediction and interpretation, particularly when dealing with datasets with many predictors or multicollinearity. The key point is employing the regularisation function with ℓ_q-norm $(q > 0)$

to penalise the coefficients:

$$\tilde{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right\}.$$

This penalisation reduces overfitted predictors ($q = 1$, the lasso) and handles high-dimensional data ($q = 2$, ridge).

- The Pareto curve is used for selecting the optimal regularisation parameter λ in the shrinkage regression. The curve visualises the trade-off between the complexity of the model (regularisation part) and the fit of the model (ℓ_2-norm of model residuals) on the log-log scale.

- The Savitzky-Golay filter is a generalised moving average method that smooths noisy signals and computes derivatives. This technique starts by using coefficients obtained from a local polynomial regression, which smooths the data while preserving the natural trend and shape of the signal. This method has two parameters: window length and polynomial order. The window length is an odd number to ensure symmetry around a central point, and the polynomial order must be less than the window length.

CHAPTER 3

Dynamical System Identification Framework

Data-driven methods play an important role in helping scientists discover governing equations from data, leading to a deeper understanding of natural mechanisms. For an unknown dynamic, the governing equation(s) may contain linear and nonlinear terms, and their closed forms need to be found. In recent years, scientists have progressively leveraged statistical and machine-learning methods to uncover governing equations of dynamical systems, especially ordinary and partial differential equations (ODEs and PDEs), from observational data [3, 4, 15–17]. This chapter summarises historical works for identifying dynamical systems in Section 3.1; illustrates the application of a regression-based identification framework, Sparse Identification of Nonlinear Dynamics (SINDy), to ODEs [3] and PDEs [4], in Sections 3.2 and 3.3, respectively; and figures out the drawbacks of the SINDy framework in Section 3.4.

3.1 Previous Works

The data-driven discovery approach traces back to the 1980s when researchers tried to estimate parameters and reconstruct the deterministic portion of dynamical systems [93, 94]. Later, scientists used statistical methods, such as least squares [95–

97], mixed-effects models [98, 99], and Bayesian approaches [16, 100], to estimate parameters in ODEs and PDEs from observational data. However, these works all assume that the underlying dynamics are known and focus on estimating coefficients of linear and nonlinear terms. Additionally, with the widespread use of high-performance computers, which improves computing ability, symbolic regression has enabled the physical and engineering communities to discover governing equations from data [2, 15, 24, 101].

SINDy framework [3, 4] represents a notable development in this arena. Using regression-based methods to identify differential equations from datasets typically involves a combined framework of differentiation (Section 2.5) and sparse regression (Section 2.3 and 2.4). The original SINDy [3] employs sequential threshold least squares regression (STLSQ) to select active terms within a nonlinear, high-dimensional candidate library and to estimate their coefficients. In recent years, the focus has shifted towards SINDy, with research directed at developing new algorithms for identifying the first-order ODEs derived from high-dimensional models. These enhancements span improvements in noise robustness and fast computation [4, 102–106], denoising [107–111], allowing coordinate transformations from complex forms [112–114, 114], solving implicit forms [115, 116], and varying time scales [117, 118]. The breadth of application for these methodologies extends across multiple fields including fluid mechanics [119, 120], civil engineering [119], biological and chemical systems [115, 121–124], epidemiology [19], stochastic processing [125–127], hybrid and cyber-physical systems [128–131], and social science [132].

PDE functional identification of nonlinear dynamics (PDE-FIND) [4] is SINDy's PDE version, i.e., identify PDEs from data by solving an ℓ_0-norm problem using sequential threshold ridge regression (STRidge). The candidate library in PDE-FIND is built by incorporating spatial partial derivative terms. Identifying dynamics governed by PDEs is more challenging than ODEs, so fewer articles cite PDE-FIND than SINDy (see Fig. 3.1). Inspired by PDE-FIND, researchers have investigated and improved approaches to identifying PDEs. For example, the sparse regression algorithm was replaced by group lasso [103], reweighted ℓ_1-regularised least squares [31], and Bayesian regression methods [106, 133]. Other improvements include reducing the PDE to a set of coupled ODEs [114] and converting the PDE to its weak form [134, 135]. Particularly, weak SINDy [109, 136] demonstrates higher noise tolerance than PDE-FIND in identifying PDEs. However, it has more computational steps and introduces more hyperparameters than PDE-FIND, requiring more expertise (e.g., manual tuning of the hyperparameters) to operate the algorithm.

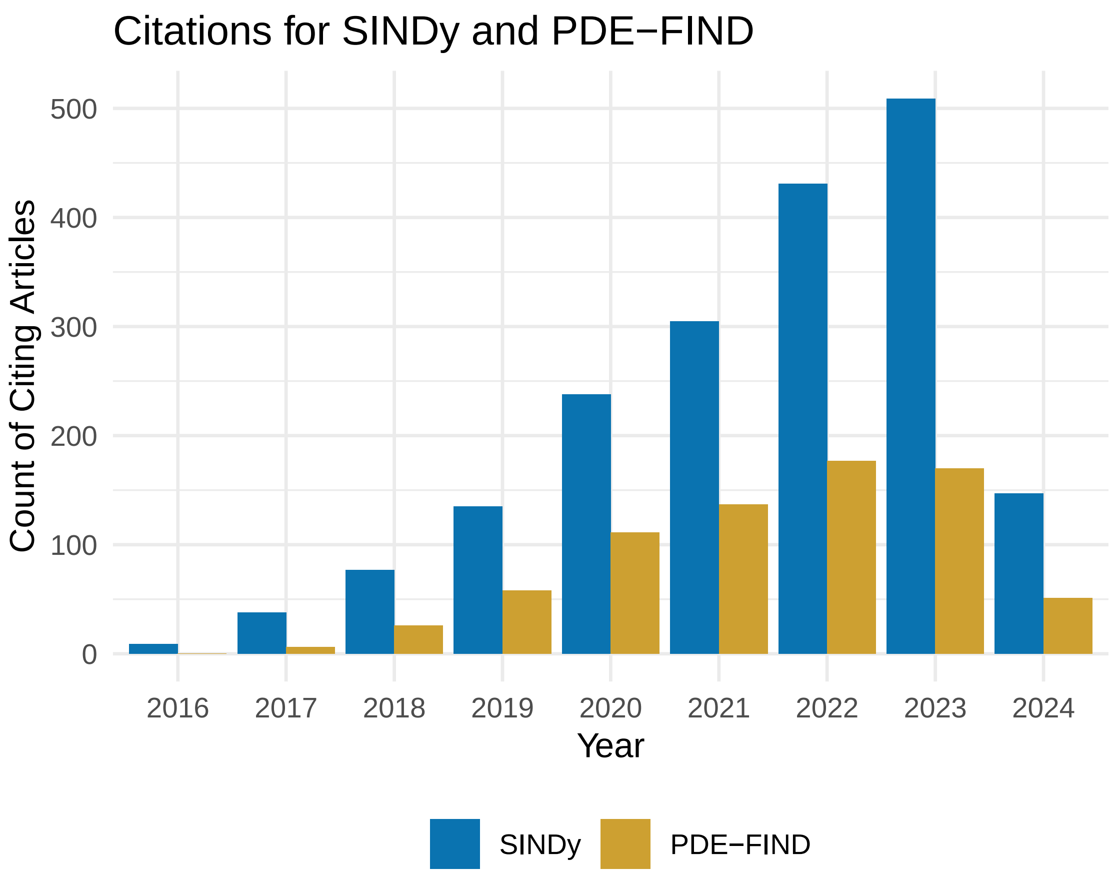


Figure 3.1: The number of articles citing SINDy and PDE-FIND. 715 articles have cited SINDy, and 302 articles cited PDE-FIND. Data was from the Web of Science, accessed on 29-04-2024.

In these SINDy-based methods, there are two groups of hyperparameters: one calculates numerical derivatives, and the other controls the results of sparse regression. As shown in Section 2.5, noise dominates numerical differentiation from the finite differences when the data contains any noise, leading to largely biased derivatives, especially in calculating higher-order derivatives [75, 76]. In this way, when the measurement data has no noise, the finite difference provides accurate derivatives to construct the candidate library, and when there is noise, smoothing filters, e.g., smoothed finite difference and Savitzky-Golay, can smooth the noisy data and calculate accurate derivatives [4, 103]. After smoothing data, sparse regression methods [4, 31, 103, 106, 133] should be used to find the correct terms and compute the accurate coefficients.

However, expert knowledge is required to manually determine these hyperparameters. When calculating derivatives to construct the candidate library, numerical differentiation methods (except finite difference and the spectral method) require parameters. Savitzky-Golay filter, a local polynomial interpolation method, needs the polynomial order and the window length [78]. The sparse regression can only identify the correct governing equations (i.e., no missing or extra terms but slightly different coefficients from the true equation) when the derivatives are calculated with the appropriate parameters. Although a nonconvex optimisation problem in [75] was proposed to find the parameters for numerical differentiation methods, it was inaccurate for higher-order derivatives.

In sparse regression, the STLSQ [3] and threshold sparse Bayesian regression [133] must manually set hard threshold values to trim terms with coefficients close to zero. Although PDE-FIND changes STLSQ to STRidge and updates the hard threshold with their own iterating rule, users still need to decide the initial threshold value and the regularisation parameter in ridge regression [4]. Even for the group lasso, users should classify candidate terms into several groups to implement the sparse regression [103]. Reweighted ℓ_1-regularised least squares [31] is a modified version of the multi-step adaptive lasso [137] to develop a sparser model that more accurately identifies the true equations for ODEs. This is achieved by iteratively adjusting the adaptive weights using previous estimates from the adaptive lasso. A significant advancement made by Cortiella et al. [31] 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. Automatic regression for governing equations (ARGOS) [1] is proposed based on the SINDy framework but without manually setting parameters. It also 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. However, 2000-times bootstrap resampling makes this algorithm computationally expensive.

Recent developments have combined neural network-based techniques and SINDy framework, leading to innovative approaches that enhance noise tolerance in identifying dynamical systems [17, 27, 28, 108, 138–142]. Neural networks can learn complex nonlinear relationships and effectively filter out noise, complementing SINDy's ability to identify parsimonious models. For example, methods shown in [143, 144] smooth data by solving ODEs, while physics-informed neural networks (PINNs) [17, 129, 145–147] focus on solving PDEs. Based on these neural network smoothing methods, Chen et al. [27] employed PINNs with sparse regression (PINN-SR) to denoise corrupted data, following an automatic differentiation to build the candidate library and identify active terms through STRidge. Zhang and Liu [108] applied the neural networks to denoise data; after building the candidate library through automatic differentiation [148, 149], they identified active terms in the frequency domain with a fast Fourier transform.

However, both neural network and SINDy methods require specific hyperparameter tuning, such as setting regularisation parameters or choosing network architectures. For example, STRidge requires setting a threshold to select active terms from the candidate library [4, 27, 28, 150]. Additionally, SINDy-based methods typically approximate numerical derivatives from noisy data using the Savitzky-Golay filter [3, 4]. The parameters of this filter, i.e., the polynomial degree and window size, must be carefully tuned for the optimal performance [1, 4]. Neural network approaches 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 PINNs [17, 27, 28] requires more prior understanding of the equation terms, as well as initial and boundary conditions. Furthermore, neural network methods have other limitations, such as lack of theoretical background [151], high computational cost [152], and poor software support for solving high-order derivatives [140].

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 enhancing its applicability across diverse scientific domains.

3.2 SINDy

SINDy systematically describes processes using regression methods to identify a nonlinear dynamic system [3]. It has three steps: calculating derivatives, constructing the candidate library, and implementing sparse regression. SINDy successfully identifies the chaotic Lorenz system [3], the mean-field model for the cylinder dynamics, the logistic map, and the Hopf normal form. Section 2.5 has reviewed the smoothing and numerical derivative methods. Hence, The following parts focus on constructing the candidate library and implementing sparse regression.

3.2.1 Constructing candidate library

A dynamical system can be described by a set of ODEs following the form

$$\frac{d}{dt}x_k(t) = \dot{x}_k(t) = f_k(x(t)), \qquad k = 1, \ldots, m, \tag{3.1}$$

where m is the dimension of the state space, and $f_k(x(t))$ includes all projections of $x_k(t)$. To identify which projections are active, $f_k(x(t))$ are approximated to be p possible projections:

$$f_k(x(t)) \approx \sum_{j=1}^{p} (\theta_{\mathrm{F}}(x))_{k,j}\beta_{k,j} = \theta_{\mathrm{F}}^T(x)\beta_k, \qquad k = 1, \ldots, m, \tag{3.2}$$

where β_k is the coefficient vector, and $\theta_{\mathrm{F}}(x)$ is the projection function vector. Each element of the projection function can be a basic function, including constant, polynomial, monomial, interaction, and trigonometric terms [3; 153, pg.247]:

$$\theta_{\mathrm{F}}^T(x) = \begin{pmatrix} 1 & x & x^2 & \cdots & \sin(x) & \cos(x) & \cdots \end{pmatrix}. \tag{3.3}$$

To implement system identification, the state vector of a given dynamical system $x(t)$ is expanded from measurements taken at times $t_1, t_2, ..., t_n$ to create a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_n) \end{pmatrix} = \begin{pmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_m(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_m(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_n) & x_2(t_n) & \cdots & x_m(t_n) \end{pmatrix}, \tag{3.4}$$

and the projection function $\theta_{\mathrm{F}}(x)$ becomes the candidate library

$$\mathbf{\Theta}(\mathbf{X}) = \begin{pmatrix} | & | & | & & | & | & | & \\ \mathbf{1} & \mathbf{X} & \mathbf{X}^{[2]} & \cdots & \mathbf{X}^{[d]} & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \cdots \\ | & | & | & & | & | & | & \end{pmatrix}, \tag{3.5}$$

where $\mathbf{X}^{[d]}$ denotes a matrix whose column vectors are all possible time-series of $[d]$th degree polynomial in $x(t)$. According to Eq. (3.1), the time derivative to $\mathbf{X}$, $\frac{d}{dt}\mathbf{X} = \dot{\mathbf{X}}$, is taken to obtain the response variable:

$$\dot{\mathbf{X}} = \begin{pmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_n) \end{pmatrix} = \begin{pmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_m(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_m(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_n) & \dot{x}_2(t_n) & \cdots & \dot{x}_m(t_n) \end{pmatrix}. \tag{3.6}$$

In practice, $\mathbf{X}$ is the only dataset that can be accessed. To get the $\dot{\mathbf{X}}$, one must employ a differentiation method, such as finite differences [3], Savitzky-Golay filter [4], or total variation regularisation [153, p.247]. Alternatively, if the function $\mathbf{x}(t)$ is known, the exact value of $\dot{\mathbf{X}}$ can be calculated from $\mathbf{X}$.

After constructing the candidate library, the regression model can be derived:

$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{B} + \mathbf{E}, \tag{3.7}$$

where $\mathbf{E} \in \mathbb{R}^{n\times m}$ is a matrix of residuals, $\mathbf{B}$ is a $p \times m$ matrix containing all coefficients of the p applied functions and the m state-space dimensions [3; 153, pg.248]. Thus,

$$\mathbf{B} = \begin{pmatrix} \beta_1 & \beta_1 & \cdots & \beta_m \end{pmatrix} = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & \cdots & \beta_{1,m} \\ \beta_{2,1} & \beta_{2,2} & \cdots & \beta_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p,1} & \beta_{p,2} & \cdots & \beta_{p,m} \end{pmatrix}. \tag{3.8}$$

Statistically, due to the m columns in $\dot{\mathbf{X}}$ and $\mathbf{B}$, Eq. (3.7) contains m regression models, and the random noise is irreducible in the regression model. Therefore, by splitting matrix $\mathbf{B}$ and $\dot{\mathbf{X}}$ into vectors and adding random noise, Eq. (3.9) can be obtained. In each equation, $\dot{\mathbf{x}}_k(t)$ is a $n \times 1$ vector, and β_k is a $1 \times p$ vector, $k = 1, 2, \cdots, m$.

$$\begin{cases} \dot{\mathbf{x}}_1(t) = \mathbf{\Theta}(\mathbf{X})\beta_1 + \epsilon_1, \quad \epsilon_1 \sim \mathcal{N}(0, \sigma_1^2) \\ \dot{\mathbf{x}}_2(t) = \mathbf{\Theta}(\mathbf{X})\beta_2 + \epsilon_2, \quad \epsilon_2 \sim \mathcal{N}(0, \sigma_2^2) \\ \vdots \\ \dot{\mathbf{x}}_m(t) = \mathbf{\Theta}(\mathbf{X})\beta_m + \epsilon_m, \quad \epsilon_m \sim \mathcal{N}(0, \sigma_m^2) \end{cases} \tag{3.9}$$

3.2.2 Active terms identification

Instead of using subset selection or lasso, Brunton et al. [3] applied Sequential Threshold Least Squares (STLSQ) to find the active terms. STLSQ starts with the full model and attributes estimated coefficients smaller than a user-determined threshold to zero, equivalent to giving a submodel without the corresponding predictors. Then, it repeats the cutoff process with the last submodel until the iteration stops at the maximum iteration. STLSQ only needs a threshold given by the user and can rapidly converge to a sparse solution within a few iterations, making it computationally efficient [154].

3.3 PDE-FIND

PDE functional identification of nonlinear dynamics (PDE-FIND) follows the SINDy framework but can identify PDEs by calculating partial derivatives (against t and x) and adding possible terms to construct a candidate library [4]. A summarised process of PDE-FIND is shown in Fig. 3.2.

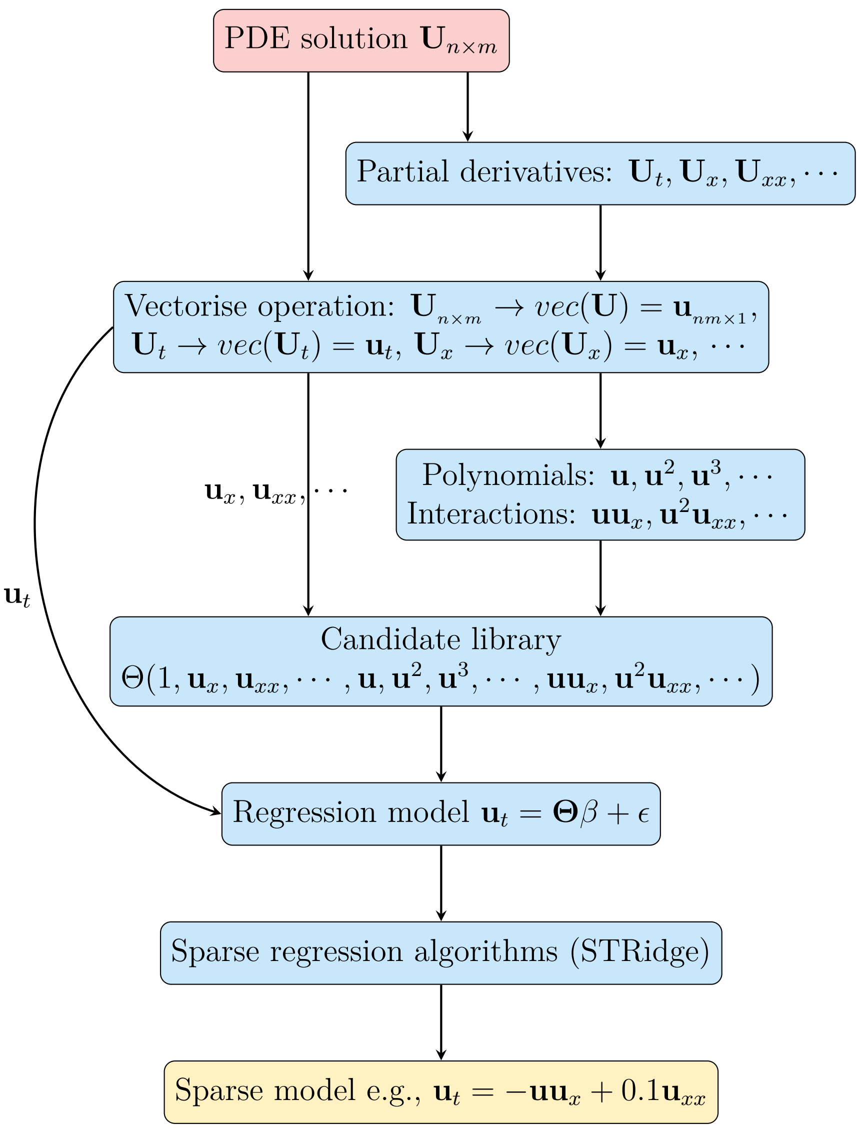

PDE solution $\mathbf{U}_{n\times m}$
Partial derivatives: $\mathbf{U}_t, \mathbf{U}_x, \mathbf{U}_{xx}, \cdots$
Vectorise operation: $\mathbf{U}_{n\times m} \to vec(\mathbf{U}) = \mathbf{u}_{nm\times 1}$, $\mathbf{U}_t \to vec(\mathbf{U}_t) = \mathbf{u}_t$, $\mathbf{U}_x \to vec(\mathbf{U}_x) = \mathbf{u}_x$, $\cdots$
$\mathbf{u}_x, \mathbf{u}_{xx}, \cdots$
Polynomials: $\mathbf{u}, \mathbf{u}^2, \mathbf{u}^3, \cdots$
Interactions: $\mathbf{u}\mathbf{u}_x, \mathbf{u}^2\mathbf{u}_{xx}, \cdots$
$\mathbf{u}_t$
Candidate library
$\Theta(1, \mathbf{u}_x, \mathbf{u}_{xx}, \cdots, \mathbf{u}, \mathbf{u}^2, \mathbf{u}^3, \cdots, \mathbf{u}\mathbf{u}_x, \mathbf{u}^2\mathbf{u}_{xx}, \cdots)$
Regression model $\mathbf{u}_t = \mathbf{\Theta}\beta + \epsilon$
Sparse regression algorithms (STRidge)
Sparse model e.g., $\mathbf{u}_t = -\mathbf{u}\mathbf{u}_x + 0.1\mathbf{u}_{xx}$

Figure 3.2: Diagram of identifying PDE systems for PDE-FIND. The illustrated process is more detailed than that in the original PDE-FIND paper, where the vectorisation is skipped. The vectorisation step is important for readers to understand how to apply regression techniques to a high-dimensional tensor.

For one-dimensional spatial PDEs, such as the Burgers equation, the finite difference can be used to calculate the derivatives of noiseless data, and the Savitzky-

Golay filter can be used for noisy data. For higher-dimensional PDEs, such as Navier-Stokes and reaction-diffusion, the derivative method is Savitzky-Golay with the sampled data [4]. The sparse regression in PDE-FIND is the ℓ_0 regression, which is an *np*-hard problem. Hence, Sequential Threshold Ridge Regression (STRidge) is proposed to estimate potential sparse models and select the one with the lowest ℓ_0 object.

3.3.1 Candidate library construction

The general form of a homogeneous PDE is

$$u_t + F(x, t, u, u_x, u_{xx}, \cdots) = 0 \tag{3.10}$$

where $F(\cdot)$ governs the behaviour of the system, with $u = u(t, x)$ 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 (3.10) serves as a foundational representation of the dynamical system, encapsulating a wide range of phenomena through its generalised form, which can be adapted to include multiple spatial dimensions or to model systems without explicit time dependence.

To focus on data-driven modelling of spatiotemporal dynamical systems, empirical data is directly incorporated into the modelling process:

$$\mathbf{U}_t = \frac{\partial \mathbf{U}}{\partial t} = \mathbf{F}\left(x, \mathbf{U}, \mathbf{U}_x, \mathbf{U}_{xx}, \dots\right), \tag{3.11}$$

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

To estimate the unknown mapping $\mathbf{F}(\cdot)$ with sparse regression, PDE-FIND [4] constructs a comprehensive library of potential terms and assuming that only a few of them are active, i.e., the estimated coefficients should be non-zero [1, 3, 4]. 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 [4]. 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 the Burgers equation, the method should ideally select only these two terms from the candidate library.

All features related to $\mathbf{U}(t, x)$ in Eq. (3.11) are matrices. Implementing this matrix data in sparse regression leads to the creation of n distinct regression models. Each model captures the spatial dynamics of the system at a specific time point t_j, where $j = 1, 2, \ldots, n$. To consolidate the n regression models into a single linear regression problem, the matrix $\mathbf{U}(t, x)$ and its derivative matrices are reshaped 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 vectorised data and candidate terms, a single sparse coefficient vector β can be estimated to represent 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

$$\mathbf{U}(t, x) = \begin{pmatrix} u(t_1, x_1) & u(t_1, x_2) & \cdots & u(t_1, x_m) \\ u(t_2, x_1) & u(t_2, x_2) & \cdots & u(t_2, x_m) \\ \vdots & \vdots & \ddots & \vdots \\ u(t_n, x_1) & u(t_n, x_2) & \cdots & u(t_n, x_m) \end{pmatrix}. \tag{3.12}$$

Vectorising Eq. (3.12) yields:

$$\begin{aligned} \mathbf{u} &= vec(\mathbf{U}) \\ &= \begin{pmatrix} u(t_1, x_1) & \cdots & u(t_n, x_1) & \cdots & u(t_1, x_m) & \cdots & u(t_n, x_m) \end{pmatrix}^T. \end{aligned} \tag{3.13}$$

Similarly, $\mathbf{u}_t = vec(\mathbf{U}_t) = vec(\partial \mathbf{U}/\partial t)$, $\mathbf{u}_x = vec(\mathbf{U}_x) = vec(\partial \mathbf{U}/\partial x)$, $\mathbf{u}_{xx} = vec(\mathbf{U}_{xx}) = vec(\partial^2 \mathbf{U}/\partial x^2)$, $\mathbf{u}^2 = vec(\mathbf{U} \odot \mathbf{U})$, and $\mathbf{uu}_x = vec(\mathbf{U} \odot \mathbf{U}_x)$, where

$\odot$ denotes the Hadamard product. The design matrix is given by

$$\boldsymbol{\Theta}(\mathbf{u}) = \begin{pmatrix} | & | & & | & & | & | & & | & & | & \\ \mathbf{1} & \mathbf{u} & \cdots & \mathbf{u}^d & \cdots & \mathbf{u}_x & \mathbf{u}_{xx} & \cdots & \mathbf{u}\mathbf{u}_x & \cdots & \mathbf{u}^d\mathbf{u}_{xx} & \cdots \\ | & | & & | & & | & | & & | & & | & \end{pmatrix}, \tag{3.14}$$

where $\mathbf{u}^d$ is a vector where all elements denote a d-th degree monomial. The interaction terms, such as $\mathbf{u}^d\mathbf{u}_x$, represent the element-wise product of the two vectors. If $\boldsymbol{\Theta}$ has a sufficient column space including all needed dynamics, Eq. (3.15) is a well-represented model. For example, if the data $\mathbf{U}(t,x)$ 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 vectorisation, $\mathbf{F}(\cdot)$ is estimated by transforming Eq. (3.11) to a linear regression model

$$\mathbf{u}_t = \boldsymbol{\Theta}(\mathbf{u})\beta + \epsilon, \tag{3.15}$$

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. Some dynamical systems, such as reaction-diffusion, contain two or more equations. When identifying such a system, the following equations are needed:

$$\begin{aligned} \mathbf{u}_t &= \boldsymbol{\Theta}(u,v)\beta_u + \epsilon_u, \quad \epsilon_u \sim \mathcal{N}(0,\sigma_u^2), \\ \mathbf{v}_t &= \boldsymbol{\Theta}(u,v)\beta_v + \epsilon_v, \quad \epsilon_v \sim \mathcal{N}(0,\sigma_v^2). \end{aligned}$$

Both equations share the same candidate library $\boldsymbol{\Theta}(u,v)$, but the sparse estimates $\hat{\beta}_u$ and $\hat{\beta}_v$ are different. Computationally, after calculating $\mathbf{u}_t$, $\mathbf{v}_t$, and $\boldsymbol{\Theta}(u,v)$, a sparse regression method is implemented one-by-one to identify the governing equations.

3.3.2 Active terms identification

Rudy et al. [4] improved STLS to STRidge because the OLS estimation is unreliable when the candidate library has a high correlation, while ridge regression is an alternative to solve multicollinearity (see Section 2.4.1). Additionally, as the lasso fails in multicollinearity for sparsity, Rudy et al. [4] used the ℓ_0 regularised regression's object function as the criterion, which is equivalent to finding several candidate models via STRidge and use Eq. (3.16) to select the best one. Instead of using a fixed tuning parameter in STLSQ, STRidge applies a changeable threshold to adjust penalisation when the algorithm gets a new estimation. The STRidge optimises the following problem[1]:

$$\text{error} = \|\mathbf{y} - \mathbf{\Theta}\hat{\beta}_{\text{STR}}\|_2 + \eta\|\hat{\beta}_{\text{STR}}\|_0, \tag{3.16}$$

where $\hat{\beta}_{\text{STR}}$ is the coefficients estimated by STRidge. By considering highly-correlated and ill-conditioned data, Rudy et al. [4] set an empirical tuning parameter, $\eta = 10^{-3}\kappa(\mathbf{\Theta})$ where $\kappa(\mathbf{\Theta})$ is the condition number of the design matrix $\mathbf{\Theta}$. After the algorithm converges, they apply the OLS to the found terms and obtain unbiased estimations.

3.4 Limitations of SINDy framework

Based on the previous review, the SINDy framework (the original SINDy [3] and PDE-FIND [4]) depends on users to determine the algorithm parameters. However, if users lack prior information about the dynamical system, the parameters are difficult to determine, leading to a doubted model. Hence, an automatic framework is required to automate setting parameters and identify the governing equations. The limitations of the SINDy framework are grouped into two parts: constructing the candidate library and finding active terms for PDEs. These are summarised from papers, supporting documents, and Python codes, available on `https://github.com/dynamicslab/pysindy` and `https://github.com/snagcliffs/PDE-FIND` [3, 4].

3.4.1 Limitations on constructing candidate library

Two limitations are found in constructing the candidate library. First, in most cases, users do not know whether the collected data is noisy or noiseless. SINDy lacks a

[1]There is a typo on the paper [4]. I have contacted the authors and confirmed that Eq. (3.16) is correct.

rule to determine whether the data is noisy, which involves using finite differences or the Savitzky-Golay filter.

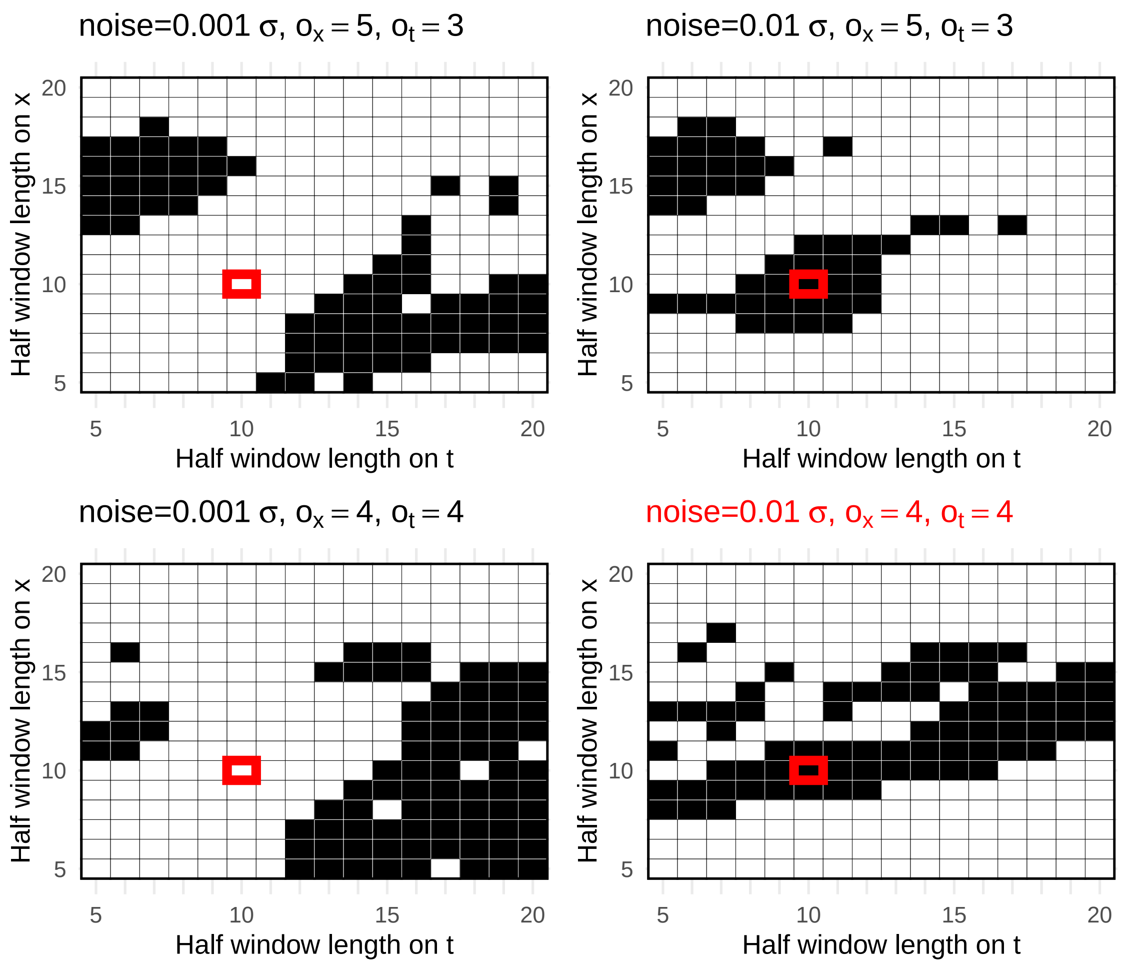


Figure 3.3: Identifying the Burgers equation using PDE-FIND with different noise levels and sliding window sizes on the x (o_x) and t (o_t) axis. Black cells mean PDE-FIND can identify the correct PDEs, while white cells are those that fail. Here, a correct PDE means that the identified PDE has no missing or extra terms. The graph with the red title represents the noise level and sliding window sizes used in PDE-FIND, and the red rectangle at each graph is PDE-FIND's sliding window size.

Second, when constructing the candidate library for noisy data, users need prior knowledge to decide the polynomial order and the sliding window sizes for the Savitzky-Golay filter. The algorithm cannot identify the correct equation if the polynomial order and sliding window sizes are inappropriate. Constructing the library in PDE-FIND for noisy data in the spatiotemporal grid, $\mathbf{u}(t, x)$, is investigated by varying noise levels and sliding window sizes along the t and x axes. In Rudy et al. [4]'s investigation, the additive noise level is limited at 1% of the data's standard deviation, i.e., $0.01 \cdot \mathrm{sd}(\mathbf{u})$. Figure 3.3 shows the exploration using PDE-FIND codes. The polynomial orders and sliding window sizes used in [4] are denoted by red titles and rectangles. Black cells indicate cases where the correct PDEs (i.e., no missing or extra terms but slightly different coefficients from the true equation) were identified, while white cells represent failures in identification. Tuning any of these parameters may fail to identify the correct PDEs, indicating that researchers need to enumerate all parameter combinations and spend a long time finding the best one. This hinders applying the SINDy framework to identify differential equations from different datasets. Rudy et al. [4] employed settings specific to particular cases, suggesting a need for improvement in automatically setting Savitzky-Golay parameters.

3.4.2 Limitations on identifying active terms

Three limitations are found in identifying active terms.

1. When using STRidge in PDE-FIND, users should manually tune the penalise parameter (λ) for ridge regression and the initial threshold (d_{tol}) for the sequential threshold process. If both tuning values are inappropriate, the algorithm cannot find the correct PDEs. Both parameters can be automatically determined by, for example, cross-validation and BIC.
2. After active terms are found, the statistical inference with OLS for the estimated model is not stable if the variables are collinear. Rudy et al. [4] do not consider the multicollinearity after the active terms are found in STRidge. Within the Python codes of STRidge, they employ the OLS estimate after the sequential threshold iteration, which can be seen in the 612th row on `PDE_FIND.py`.
3. STRidge finds redundant terms when identifying PDEs referred to the complex number, such as quantum harmonic oscillator and nonlinear Schrodinger equation. The coefficients in both equations only have the imaginary part, while the results from STRidge contained the real part.

The Automatic regression for governing equations (ARGOS) [1] framework aims to automatically identify ODEs from a dynamical system by integrating machine learning with statistical inference, but it cannot identify PDEs. The next two chapters will address these limitations from signal processing and statistical learning perspectives and extend the ARGOS framework to identify PDEs. The improvement of smoothing data and calculating derivatives is in Section 5, and a new sparse regression method, the recurrent adaptive lasso, will be illustrated in Section 6.

3.5 Summary

This chapter has reviewed the SINDy framework in identifying dynamical systems described by ODEs $\frac{d}{dt}x(t) = \dot{x}(t) = f(x(t))$ and PDEs $u_t + F(x, t, u, u_x, u_{xx}, \cdots) = 0$.

- The aim of the system identification methods is to find these differential equations from data.
- SINDy provides an innovative framework that allows sparse regression methods to identify dynamical systems.
- However, this framework can be improved based on the following parts:
  - the derivatives should be calculated automatically;
  - the sparse regression solving algorithms STLSQ and STRidge are replaced with other methods that can determine the regularisation parameters without manual tuning.

CHAPTER 4

Analysis of Other SINDy-based Methods

After the original SINDy algorithm was presented in 2016 [3], researchers aim to improve the algorithms for discovering governing equations under various circumstances, such as handling noisy and incomplete datasets, estimating noise, and measuring uncertainty in the final prediction model. Hence, different SINDy versions have been developed to address these problems [155–158]. This chapter will analyse the other three SINDy methods: **Ensemble SINDy (ESINDy)** [155], **Uncertainty Quantification SINDy (UQ SINDy)** [156], and **Modified SINDy** [157]. The analysis is a part of the work in the published paper *Automatically discovering ordinary differential equations from data with sparse regression* [1] on Communications Physics. The success rate is a key measurement in the following analysis. Here, "success" means the identified equations of the algorithm have no missing or extra terms. After selecting the correct terms, ordinary least squares estimates coefficients that are slightly different from the ones in the true equation.

4.1 ESINDy

ESINDy employs ensemble learning techniques, specifically bagging and bragging, to the SINDy framework [155]. While bagging [159] averages model coefficients from multiple bootstrap samples (see Section 2.2.4), bragging [160] uses the median for aggregation, thereby being more robust to outliers. These methods aim to increase the robustness and accuracy of SINDy, particularly for noisy data.

Bagging creates an ensemble of models robust to noise by reducing the variance of the base learning algorithm without increasing bias. This happens because the base learning algorithms are fit to various bootstrap samples and thus de-correlated, reducing variance when averaged. While this is highly effective for improving predictions, especially in algorithms like decision trees that often have high variance, its utility in sparse regression for inference is limited. Sparse regression techniques like the lasso are generally characterised by low flexibility but high interpretability, contrasting with bagging's high flexibility and lower interpretability [33, p.340]. This distinction is also highlighted in Fig. 4.1 (taken from Fig. 2.7 of [33, p.25]), which shows that the lasso and bagging are fundamentally different techniques [33].

One of ESINDy's most significant contributions is the introduction of the "inclusion probability", a user-defined threshold that quantifies the frequency at which a candidate term appears in an ensemble of q SINDy models. These models are generated from different bootstrap samples and can feature different subsets of candidate terms. For instance, a term appearing in 80 out of 100 models would have an inclusion probability of 0.8. This probability serves multiple purposes: it can be used for automatic term thresholding, uncertainty quantification, and improving model interpretability. Low probabilities are indicators of term irrelevance, whereas high probabilities suggest that a term is robust and critical for model accuracy.

It is worth noting that the inclusion probability is influenced by the number of bootstrap samples, introducing additional hyperparameters into the SINDy algorithm. ESINDy recommends a default of 100 bootstrap samples, which may not be optimal for smaller datasets like those used in [1].

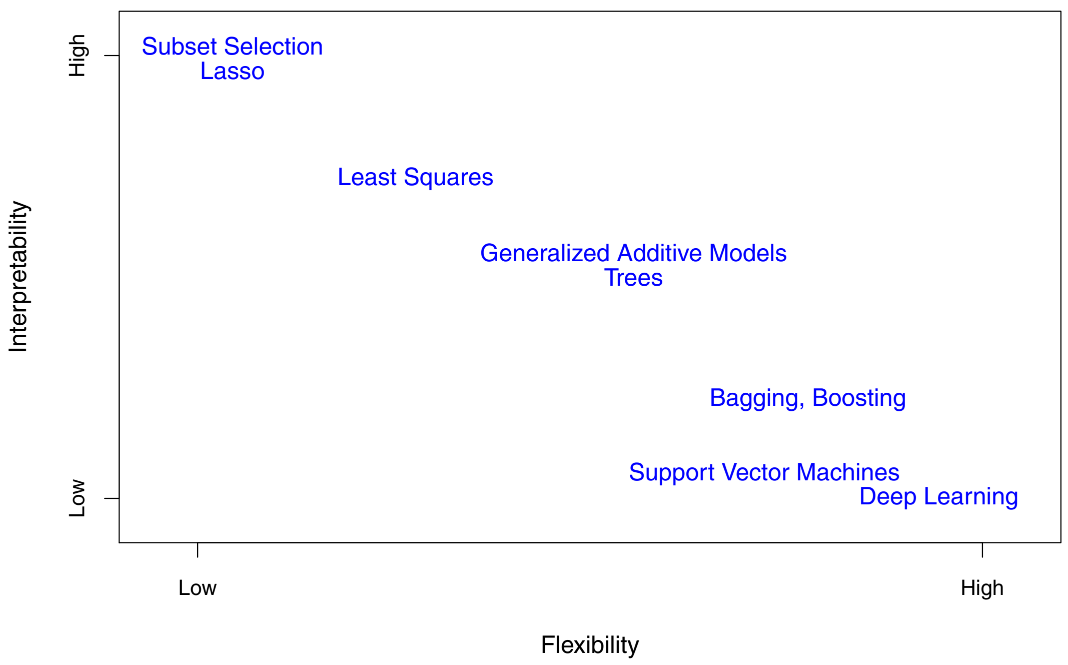


Figure 4.1: A representation of the trade-off between flexibility and interoperability, different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases (taken from Fig. 2.7 of [33, p.25]).

4.1.1 Hyperparameter tuning with ESINDy

One noteworthy challenge ESINDy introduced is the complexity of tuning hyperparameters, which are all human-determined. The original SINDy approach [3] requires tuning a single threshold. However, the standard ESINDy involves four hyperparameters, while library ESINDy requires five, which include:

- λ_{SINDy} (a SINDy threshold),
- Polynomial order o_{SG} and window length l_{SG} for the Savitzky-Golay filter,
- tol (the bagging tolerance, which is a threshold set manually, $tol = 0.6$ in ESINDy with b(r)agging and at $tol = 0.4$ in library ESINDy),
- l (the number of library columns sampled in library ESINDy set at $l = 0.6D$, where D is the number of candidate library functions).

These additional parameters can increase the chance of retaining irrelevant predictors and compromise the sparse model representation.

4.1.2 Results of ESINDy tests

To investigate the influence of hyperparameters on model performance, the benchmark tests as outlined in [1] are replicated, i.e., changing the initial conditions to the Lorenz system in both sample size and SNR tests, and randomly sampling noise in SNR test. Here, the time step Δt of the simulated data is fixed at 0.001. As the data is simulated with noise, Δt will influence the smoothed results. A smaller Δt will provide a higher resolution of the simulated data, providing more system details. After applying the Savitzky-Golay filter, the system's trajectory can be restored. However, with Δt increasing, e.g., $\Delta t > 1$, the simulated data will be scarce, i.e., details of the dynamics will be lost during the simulation. Due to this, the dynamics of the smoothed data will be lost because the Savitzky-Golay filter uses neighbour points to estimate the central point.

The findings indicate that bagging with ESINDy enhances identification performance as the time series n length increases, Fig. 4.2(a). Conversely, it is also noted that the performance of other ESINDy variants decreases with an increase in n. This observation implies that hyperparameter tuning should consider the variable n, which is impractical with real-world data due to the unpredictability of n's impact on the identification algorithm. Furthermore, when $n < 1000$, i.e., the Lorenz system evolves less than a time unit, the success rates of all methods are close to zero, indicating that a sufficient system evolution time is required in all state-space variables. The SNR plot in Fig. 4.2(b) suggests that library ESINDy bagging and bragging are superior methods, as they have similar performance and begin to identify correct model when SNR > 25 dB. Although ESINDy bagging and library ESINDy with bagging and bragging are more frequent in identifying correct equations with SNR increasing, ESINDy bagging starts identification when SNR > 37 dB. It is worth noting that achieving this level of performance necessitates meticulous tuning of the 4-5 hyperparameters, highlighted by the method's inconsistent performance as the number of observations increases.

The absence of specific tuning guidelines in the original ESINDy paper limits its applicability and raises questions regarding its recommended hyperparameters,

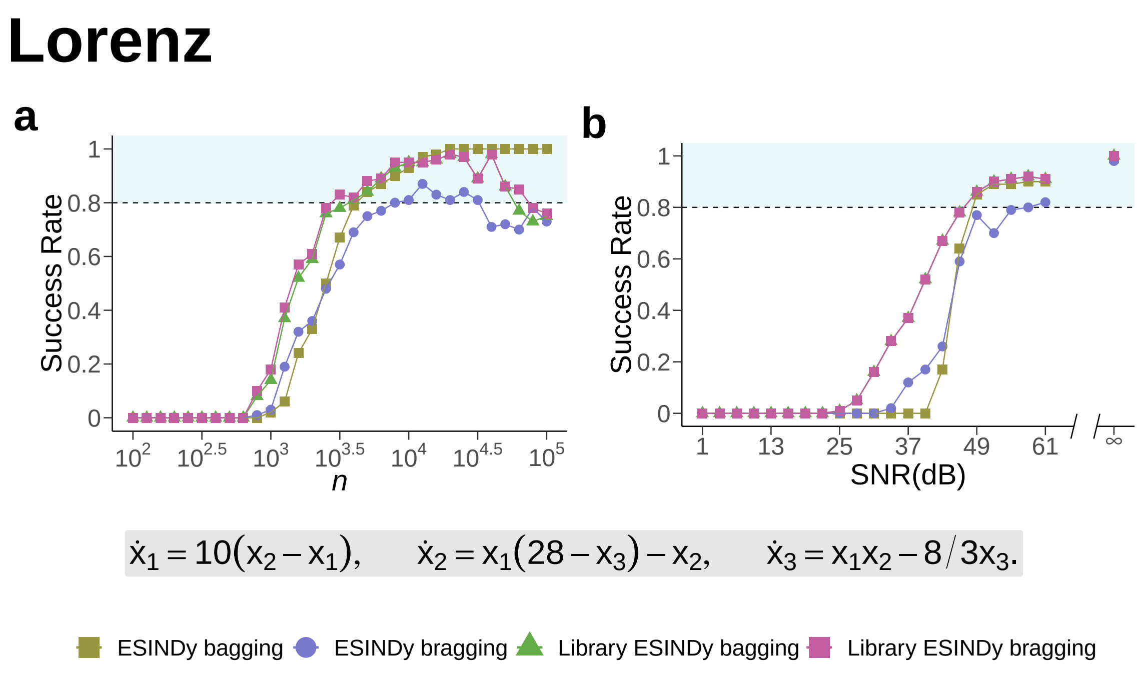


Figure 4.2: Success rate of the ESINDy for identifying the Lorenz system. The test implements 100 random initial conditions and examines the success rate of Ensemble SINDy in discovering the correct terms of the governing equations from each system at each value of n and SNR. When the time-series length n increases, SNR is fixed at 49 dB (left panel); when SNR increases $n = 5000$ (right panel). Shaded regions represent model discovery above 80%.

potentially leading to inappropriate threshold settings when identifying different systems. For instance, in the case of identifying the Rossler attractor, where $\dot{x}_2 = x_1 + 0.2x_2$, employing $\lambda_{\text{SINDy}} = 0.2$ consistently excludes the x_2 predictor from the final model because its coefficient is smaller than the threshold. This results in inaccurate model discovery and introduces domain-specific challenges that often require system knowledge for proper identification.

Following the heat plot in [155], Fig. 4.3 demonstrates the evaluation of the original SINDy, ESINDy with bagging, bragging, and library bagging. Each heat plot in Fig 4.3 shows the success rate when noise level and data length change. Generally, the bottom left of the graph represents the higher success rate with more data points and lower noise levels. The first row depicts results using the original conditions outlined in the ESINDy paper. The second row shows that performance deteriorates when the initial conditions are changed while retaining ESINDy's original Savitzky-Golay parameters ($o_{SG} = 3, l_{SG} = 5$), highlighting the necessity of

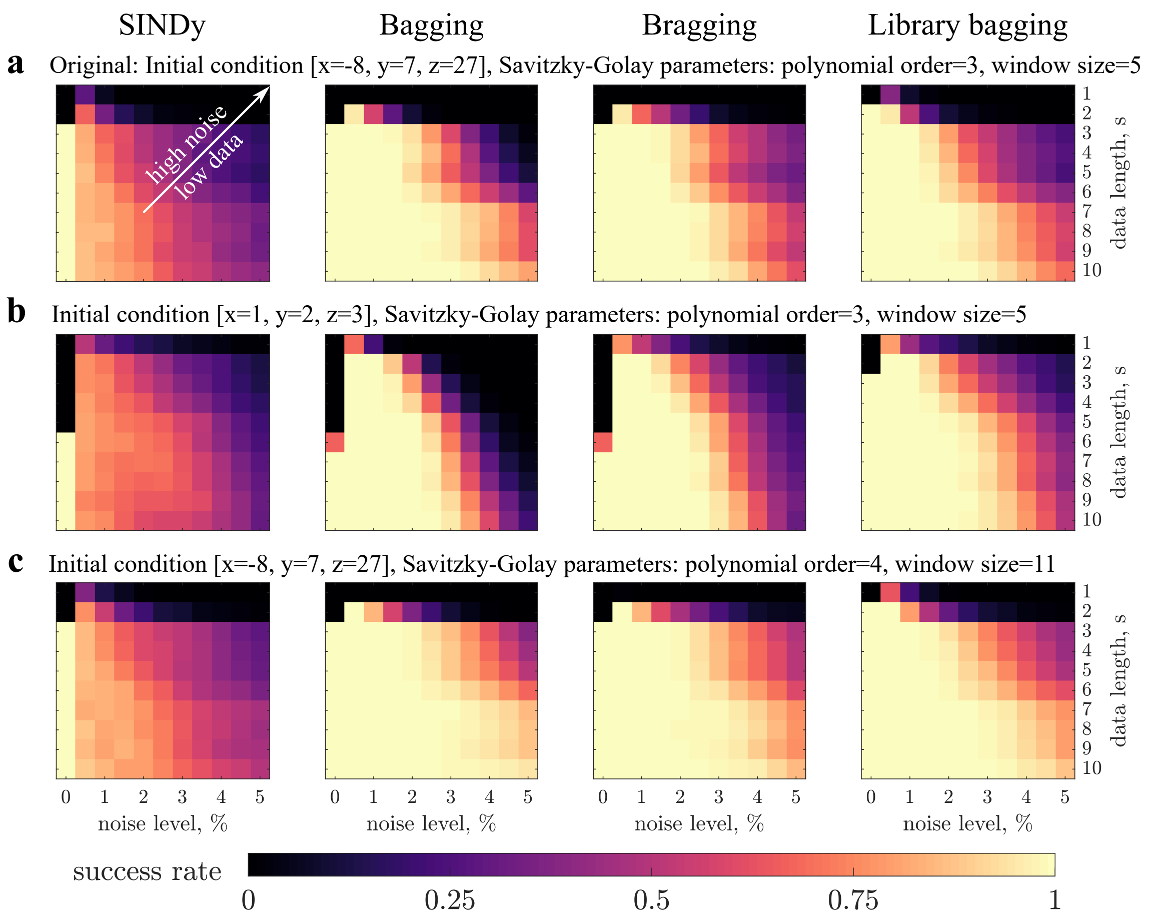


Figure 4.3: Multi-faceted evaluation of ESINDy's performance under varied conditions. Each SINDy method's success rates are determined across varying noise levels and time series lengths using 1000 noise realisations. For each graph, the x-axis shows the noise level, the percentage of the data's standard deviation, and the y-axis is the data length shown in seconds with $\Delta t = 0.01$. (a) Original results based on the hyperparameters and initial conditions recommended in the ESINDy paper. (b) Deteriorated performance when using altered initial conditions but retaining the original Savitzky-Golay filter parameters. (c) Improved performance when original initial conditions are paired with different Savitzky-Golay filter settings.

evaluating each method under varied initial conditions for a thorough effectiveness assessment. The third row demonstrates improved results upon reverting to the original initial conditions but applying different values for the Savitzky-Golay filter, collectively emphasising ESINDy's significant reliance on manual hyperparameter tuning.

4.2 UQ-SINDy

The UQ-SINDy method integrates Bayesian inference with the SINDy to identify dynamical systems [156]. It aims to quantify uncertainties in model parameters and identify the most probable model from data. This framework introduces sparsity-promoting priors – the spike-and-slab and regularised horseshoe priors – to handle the challenges of limited and noisy data effectively, enabling the identified model to be robust to noise.

Like ESINDy, UQ SINDy employs “inclusion probabilities” to quantify the probability that a given candidate term belongs in the final model. In the spike-and-slab priors, Eq. (4.1), these probabilities are calculated based on the posterior mean of a binary inclusion variable λ_j, which takes a value of 1 for inclusion and 0 for exclusion.

$$\beta_j|\lambda_j \sim \lambda_j \mathcal{N}(0, c^2) + (1-\lambda_j)\mathcal{N}(0, \epsilon^2) \tag{4.1}$$

Here, c^2 and ϵ^2 are included and excluded variances of β_j respectively. Shrinkage factors serve as pseudo-probabilities when using the regularised horseshoe prior, Eq. (4.2). These probabilities indicate the significance of each term, with values near zero suggesting potential irrelevance and values closer to one implying importance.

$$\begin{aligned}
&\beta_i \mid \tilde{\lambda}_i, \tau, c \sim \mathcal{N}\left(0, \tilde{\lambda}_i^2 \tau^2\right), \\
&\tilde{\lambda}_i = \frac{c\lambda_i}{\sqrt{c^2+\tau^2\lambda_i^2}}, \\
&\lambda_i \sim C^+(0,1), \\
&c^2 \sim IG\left(\frac{\nu}{2}, \frac{v}{2}s^2\right) \\
&\tau \sim C^+\left(0, \tau_0\right),
\end{aligned} \tag{4.2}$$

Here, C^+ denotes half-Cauchy distribution with τ_0 controls the scale. IG denotes the inverse gamma distribution with two parameters, ν and s, controlling the shape and scale of the distribution. It should be noted that UQ SINDy with both priors relies on manual selection for specific experiments and does not specify universal threshold values for inclusion probabilities. Hence, domain knowledge remains a key component throughout the model discovery process.

Another consideration in UQ SINDy is the role of Markov Chain Monte Carlo (MCMC) sampling in estimating uncertainty. UQ SINDy uses 2000 MCMC samples for initial model discovery. Although example cases often involve thousands of MCMC samples, it is essential to conduct convergence diagnostics to ensure that the sample size is sufficient for the problem.

4.2.1 Results of UQ SINDy tests

The effectiveness examination of UQ SINDy follows the tests outlined in the corresponding journal article [156]. A significant challenge encountered is the computational overhead resulting from the MCMC sampling process, especially in dealing with high-dimensional posterior distributions. The computational demands increase with the number of variables in the design matrix, challenging the method's scalability. The issue is depicted in Fig. 4.4, showing the rising computational burden as the design matrix grows in dimensionality. For example, to identify the Lotka-Volterra system (having two state variables) with a monomial order of five for a system size of $n = 100$, UQ SINDy necessitates approximately seven hours. Although the Lorenz system is initially considered in this test, the computing time is too long and exceeds 20 hours for the system size of $n = 100$ and monomial order of five. Hence, Lotka-Volterra equations are an alternative used in the current test.

Figure 4.5 illustrates the method's success rate over ten instances of the Lotka-Volterra system as the polynomial order of the design matrix increases. Initial tests, guided by suggestions from the original article [156], use a pseudo-probability threshold of 1 for the regularised horseshoe prior. However, this threshold fails to discover the underlying system. To overcome this issue, the threshold is adjusted to 0.8, resulting in an improved identification rate. For using the spike-and-slab prior, in the absence of a specified threshold for inclusion probabilities from the original article, a decision is made to employ an inclusion probability of 0.5, which facilitates model discovery. The results reveal that UQ SINDy achieves optimal performance with a design matrix incorporating terms up to a polynomial order of $d = 2$ with six predictor variables. However, the success rate declines to zero for polynomial orders

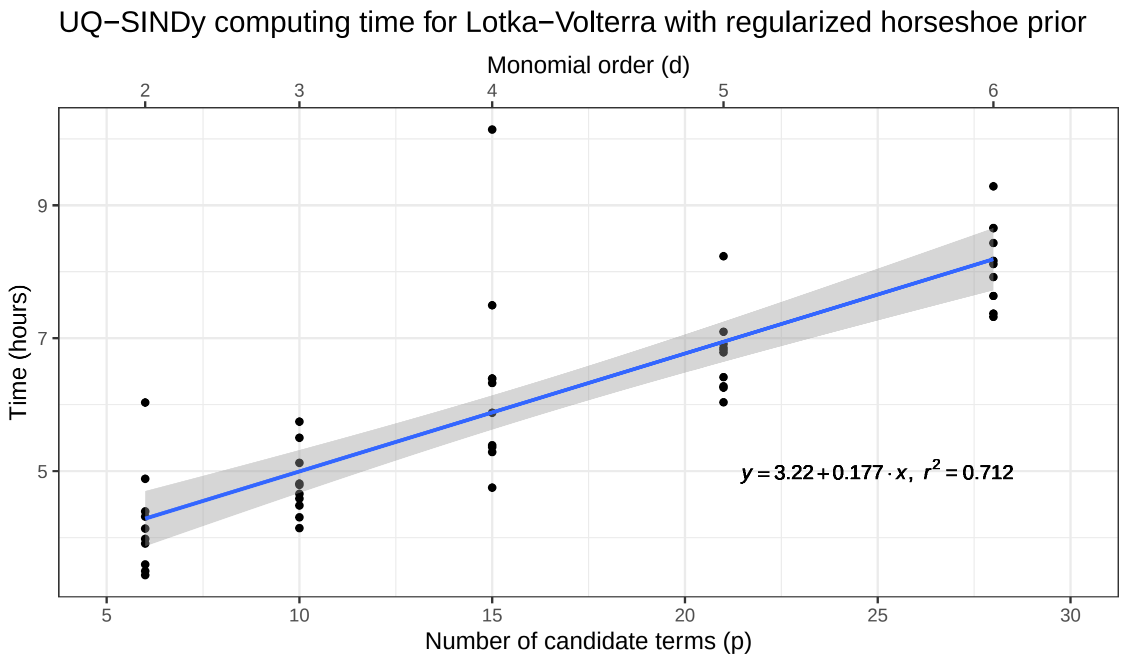


Figure 4.4: Running time of UQ SINDy. It is a function of both polynomial order and the corresponding number of variables in the candidate library, using the regularised horseshoe prior. Ten instances of the Lotka-Volterra system were identified with parameters $n = 100$, SNR $= 49$, and $x(t = 0) = [10, 5]$. The equation accompanying the blue line indicates the fitted mean computational time for UQ SINDy as the number of candidate terms increases.

beyond this, underscoring an increasing unreliability with the growing complexity of the design matrix.

These findings highlight that the method struggles with computational efficiency and model discovery as the number of observations and predictors in the design matrix increases. As a result, its practicality for real-world applications is significantly limited, as opposed to the ARGOS framework, which is comparatively efficient as these conditions increase.

4.3 Modified SINDy

Modified SINDy [157] incorporates neural networks within the SINDy framework to address the noise-related challenges in identifying dynamical systems. Central to this method is a coupled cost function, which is optimised for two core components:

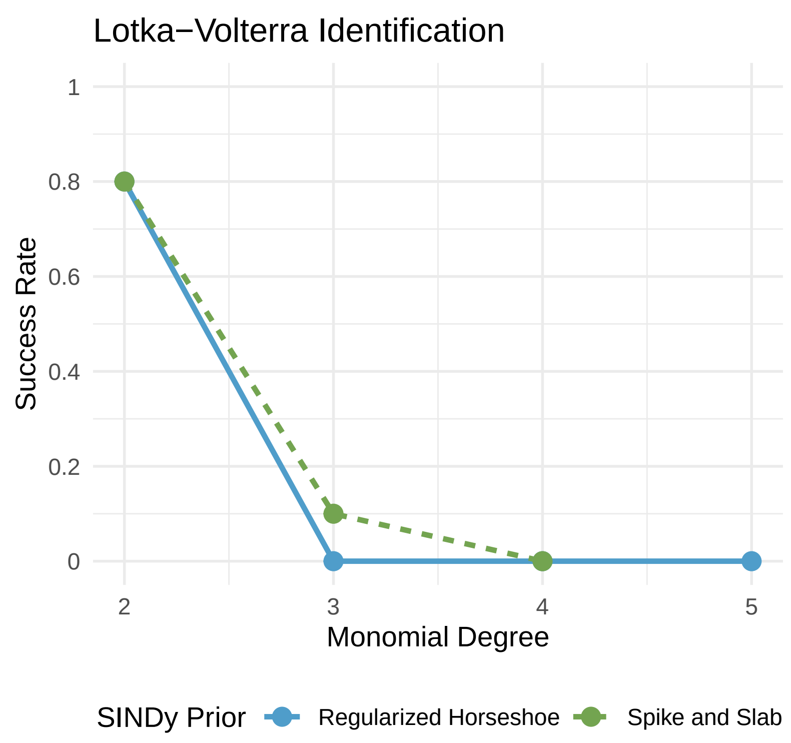


Figure 4.5: Success Rate of UQ SINDy when applied to 10 instances of the Lotka-Volterra system with $n = 100$, SNR $= 49$, and $x(t = 0) = [10, 5]$. Both spike and slab and regularised horseshoe priors were tested.

e_s, the error of estimated noise $\hat{\mathbf{N}}$, and e_d, the error system parameters $\mathbf{\Xi}$.

$$\begin{aligned} \mathbf{\Xi}, \hat{\mathbf{N}} &= \underset{\mathbf{\Xi}, \hat{\mathbf{N}}}{\text{argmin}} \mathcal{L}(\mathbf{\Xi}, \hat{\mathbf{N}}), \\ \text{s.t.} \quad & (|\mathbf{\Xi}| < \lambda) = 0. \end{aligned} \tag{4.3}$$

$$\mathcal{L}(\mathbf{\Xi}, \hat{\mathbf{N}}) = e_s + e_d = \omega \|\mathbf{Y} - \hat{\mathbf{N}} - \hat{\mathbf{F}}(\hat{\mathbf{X}})\|_2^2 + \|\dot{\hat{\mathbf{X}}} - \mathbf{\Theta}(\hat{\mathbf{X}})\mathbf{\Xi}\|_2^2 \tag{4.4}$$

$$\mathbf{Y}_{j+q} - \hat{\mathbf{N}}_{j+q} = \hat{\mathbf{X}}_{j+q} = \hat{\mathbf{F}}^q(\hat{\mathbf{X}}_j) = \hat{\mathbf{X}}_j + \int_{t_j}^{t_{j+q}} \mathbf{\Theta}(\hat{\mathbf{X}}(\tau))\mathbf{\Xi} d\tau \tag{4.5}$$

where $\mathbf{Y}$ is the observed noisy data, $\hat{\mathbf{N}}$ is the denoised signal, $\hat{\mathbf{X}}$ is the estimated signal, and λ is a threshold. Hence, this dual optimisation problem has three main objectives: 1) denoising time-series data, 2) learning and parameterising the noise probability distribution, and 3) discovering the underlying dynamical systems.

Modified SINDy incorporates $\hat{\mathbf{N}}$ as a set of parameters in the cost function. By minimising the cost function e_s in Eq. (4.4), the algorithm derives the denoised signal $\hat{\mathbf{X}} = \mathbf{Y} - \hat{\mathbf{N}}$. The optimised $\hat{\mathbf{N}}$ aims to capture the inherent noise in the observed data $\mathbf{Y}$, providing empirical insights into the data's noise behaviour. Concurrently, the system parameters Ξ are fine-tuned within the same cost function by applying a sparsity-promoting threshold to Ξ and yielding a sparse representation of the underlying dynamical system.

More detailed, modified SINDy employs the Adam optimiser to minimise the cost function, allowing it to work efficiently across different noise conditions. This optimiser requires several hyperparameters, including two exponential decay rates β_1 and β_2, a numerical stability constant ϵ, and the learning rate η_t [161]. While modified SINDy sets default values for the Adam optimiser parameters, it introduces three additional user-defined hyperparameters: the cut-off λ in the problem (4.3), the iteration times N_{loop} for this optimisation problem, and the prediction step q in Eq. (4.5). Appendices B to D in [157] detail the effects of these hyperparameters on the method's success in model selection. However, manually adjusting λ indicates that the algorithm is not automated. Furthermore, Fig. 12 in [157] suggests that the computational time increases linearly as the value of q increases. Therefore, although the modified SINDy performs better than the original SINDy under noisy conditions, unlike the convex optimisation method in the ARGOS framework, the demand for additional computational resources to train parameters limits the method's scalability in complicated applications such as identifying high-dimensional systems.

4.3.1 Results of modified SINDy tests

The following part will explore the modified SINDy method as delineated by Kaheman et al. [157]. This process involved evaluating its identification results according to the criteria established in the paper [157]. Upon in-depth examination, several notable limitations of the modified SINDy method are found.

First, modified SINDy strongly depends on manually tuning hyperparameters. As previously mentioned, modified SINDy requires tuning up to seven hyperparameters: four from the Adam optimiser and three additional ones defined by the user. However, manual tuning of hyperparameters is incompatible with the automated tests, which cover a wide range of initial conditions, time series length, and

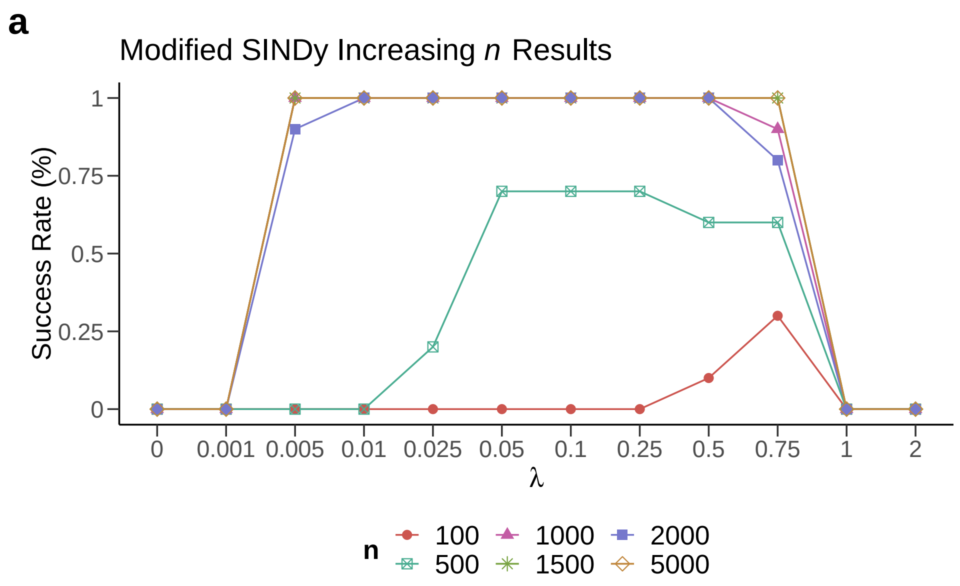
SNRs.

Properly tuning λ is critical for the success of modified SINDy. The authors advise starting with low values, such as 0.1, and increasing it until a good model is identified. Figure 4.6 shows modified SINDy's effectiveness in identifying the Lorenz system, even with limited observations and moderate to low SNRs. Figure 4.6(a) illustrates the success rate with changes in sample size and threshold under SNR = 49 dB. When $n \geq 1000$, modified SINDy successfully identify model with $\lambda \in (0.005, 1)$. However, when $n \leq 500$, modified SINDy struggles in the success rate. Figure 4.6(b) shows the success rate varying with SNR and threshold when there are 2500 observations. As SNR increases, more λ values can be used to identify the correct equations. When SNR = 20 dB, the maximum success rate for modified SINDy is 75% at $\lambda = 0.75$. However, when SNR $\geq$ 49 dB and $\lambda \in [0.005, 0.75]$, the success rate of modified SINDy is 100%. Both n and SNR results suggest an intricate relationship between these factors, ensuring the success rate can rise to 100%.

Since all parameter values in the Lorenz system are larger than one, the hyperparameter λ can be conveniently chosen between zero and one. This highlights that modified SINDy can only be used when prior knowledge of the underlying system exists. However, some systems' parameters are smaller than one, such as the two-dimensional linear, cubic, and three-dimensional linear systems used in evaluating ARGOS [1]. If λ starts from 0.1, the correct equation will never be found for these systems because some coefficients are smaller than 0.1. In contrast, the ARGOS framework offers a distinct advantage by automating tuning parameters, effectively reducing the dependency on domain knowledge during model discovery.

The other limitation is the degree of the library. In the context of modified SINDy, library degree d plays a pivotal role. Appendix J in [157] advises users without expert knowledge of the system to first employ a simple second-order polynomial library to assess the performance of the identified model. If the initial results are unsatisfactory, the recommendation is to increase d incrementally. However, as the library degree grows, two main challenges surface.

1. Modified SINDy will produce `NaN` when degrees $d > 3$. This is confirmed



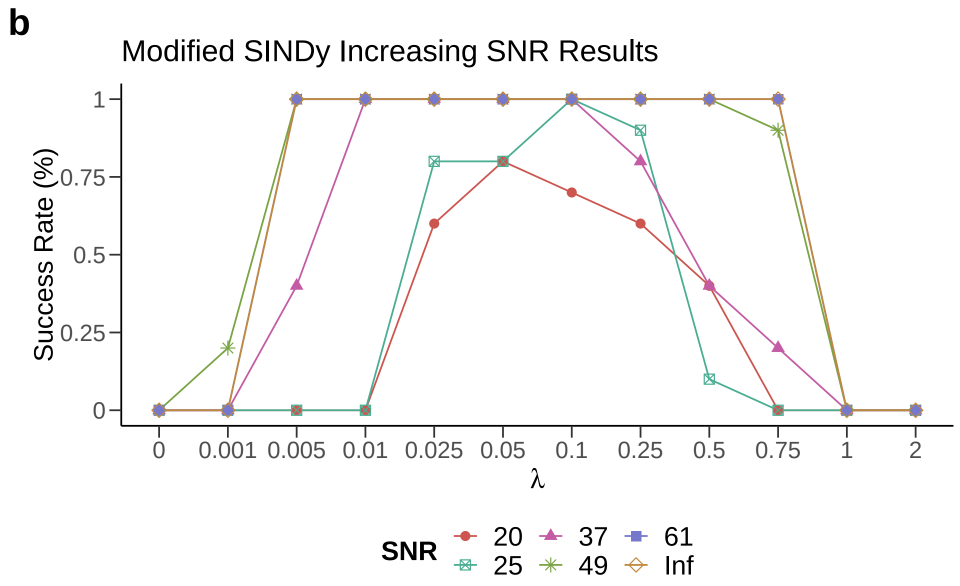


Figure 4.6: Modified-SINDy model discovery results for the Lorenz system, influenced by ten instances of random noise with $x(t) = [5, 5, 25]$. Success rate as a function of λ for several values of (a) n, holding SNR $= 49$ constant; (b) SNR, fixing $n = 2500$. Terms of the monomial degree of the design matrix $\mathbf{\Theta}(\mathbf{X})$ are limited to three due to modified SINDy's failure at handling degrees $d > 3$, resulting in forward and backward simulation producing `NaN` costs.

in Appendix J, which suggests that when the design matrix contains higher polynomial orders, the forward and backward simulation may fail [157].

2. Although Fig. 17 in [157] shows results for polynomial degree $d = 4$, the design matrix omits the constant term $\mathbf{1}$. However, the code running crashes when keeping the constant in the design matrix, rendering the method ineffective. This constraint impedes the automated tests using a library degree $d = 5$ in the candidate library.

Despite the challenges that have been discovered hindering the systematical evaluation of modified SINDy, it is important to acknowledge this method's advantages. Appendix E of [157] shows it is twice as robust to noise as its original SINDy, with an ability to handle noise levels exceeding 40%. This robustness is highlighted by its performance in model discovery under conditions of high noise contamination. Moreover, the method's default hyperparameter settings have been shown to facilitate successful model discovery across various examples, including complex systems like the Lorenz and Rossler systems. This indicates Modified SINDy's potential in applications where noise resilience and effective model identification are paramount despite the need for computational resources to explore a range of hyperparameters.

4.4 Summary

In summary, all methods require a deep understanding of the system to determine parameters in identification. Additionally, UQ-SINDy and modified SINDy are computationally expensive, while ESINDy uses 100 bootstrap samples, costing less time than UQ-SINDy and modified SINDy.

- **Ensemble SINDy (ESINDy)** [155] employs bagging to average model coefficients, selecting terms that exceed a predetermined inclusion probability. It relies on several hyperparameters without a comprehensive guideline for establishing their optimal values, including manually setting both the sparsity-promoting threshold and the inclusion probability.
- **Uncertainty Quantification SINDy (UQ SINDy)** [156] uses Bayesian sparse regression to quantify uncertainty, employing inclusion and pseudo probabilities for variable selection. It lacks a universal threshold for inclusion probabilities and requires 2000 MCMC samples, making it computationally intensive.
- **Modified SINDy** [157] incorporates neural networks to address noise-related issues in system identification. Limited by the design matrix's terms and necessitates tuning of hyperparameters, especially the sparsity-promoting threshold. High computational demand hampers its inclusion in automated tests.

CHAPTER 5

Automatically Finding the Optimal Parameters for Savitzky-Golay Filter

Smoothing data is important in data analysis and signal processing, serving as a powerful tool to reveal underlying patterns obscured by noise or fluctuations. By applying smoothing techniques, researchers can effectively reduce noise and outliers, revealing underlying trends and patterns that might otherwise remain obscured. The smoothing methods improve the quality of data visualisation and serve as a foundational step in predictive modelling, allowing for more accurate forecasts and insights, and thus, support informed decision-making and contribute to the advancement of scientific knowledge.

The derivative (gradient) is also important as it describes the data's rate of change, allowing researchers to analyse concavity and find the maximum or minimum points. Generally, there are three categories of differentiation methods: symbolic, numerical, and automatic differentiation [162]. Symbolic and automatic differentiation methods require users to know the mathematical expression of the function. When the expression of the data is unknown, numerical differentiation is more efficient than the other two.

As described in Section 2.5, the Savitzky-Golay filter has been distinguished as the most effective local method in derivative evaluations [4, 82], and Breugel et al. [75] proposed a semi-automatic Savitzky-Golay filter in the `PyNumDiff` package. This chapter will present the automatic Savitzky-Golay filter with Gaussian blur, which fully automates the selection of hyperparameters, smooths data, and calculates derivatives simultaneously. To explore the performance, the `PyNumDiff`'s Savitzky-Golay filter (shown in Section 2.5.4) is set as the benchmark to compare with the newly proposed method.

5.1 Savitzky-Golay filter with Gaussian blur

The notation in Table 5.1 is used to explain the differences between methods. The expression $y = f(x)$ represents a series of one-dimensional data that depend only on one variable, x, whereas $u = f(x, t)$ is for two-dimensional data that depend on two variables, x and t. Furthermore, given that two-dimensional data are stored in a matrix format, it is defined that each row represents the value at varying x, while each column corresponds to different t.

Table 5.1: Notation of symbols used in this chapter. One-dimensional (1D) data $y = f(x)$ means the data only depends on one variable x. Two-dimensional (2D) data $u = f(x, t)$ means the data depends on two independent variables x and t.

1D data	2D data	
y	u	The noiseless signal
$\tilde{y}$	$\tilde{u}$	The noisy signal
$\hat{y}$	$\hat{u}$	The smoothed signal
$\dot{y}$	u_x	The true first-order derivative
$\ddot{y}$	u_{xx}	The true second-order derivative
$\hat{\dot{y}}$	$\hat{u}_x$	The estimated first-order derivative
$\hat{\ddot{y}}$	$\hat{u}_{xx}$	The estimated second-order derivative

Although the two-dimensional Savitzky-Golay filter has been proposed [163], when calculating the derivative $u_x = \frac{\partial u(t,x)}{\partial x}$, the variable t is considered known, and the calculation reduces to the original Savitzky-Golay filter. Therefore, when dealing with two-dimensional data, the Savitzky-Golay filter is implemented with one pair of $\{d, l\}$ columns by columns (or rows by rows) to calculate the partial derivatives.

As the ground truth is unknown, it is difficult to find the best hyperparameters of the Savitzky-Golay filter. Here, the Gaussian blur with kernel κ is regarded as the ground truth because Gaussian blur can smooth data regardless of dimensions. The optimal hyperparameters $\{o^*, l^*\}$ correspond to the lowest MSE between the Savitzky-Golay filtered and Gaussian blurred data. For one-dimensional data, the process is

$$\{o^*, l^*\} = \arg\min_{o,l} \{\mathrm{MSE}\left(\mathrm{SG}(\tilde{y}, o, l), \mathrm{GB}_\kappa(\tilde{y})\right)\}, \tag{5.1}$$

where

$$\kappa = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}.$$

Here, the simplest Gaussian kernel is always selected in the following analysis. Although the Gaussian kernel can be wider, such as $[1 \quad 2 \quad 4 \quad 2 \quad 1]$ for one-dimensional data, more details will be lost due to Gaussian blur's weighted average calculation, leading to over-smoothed data [77, p.168]. Furthermore, as discussed in Section 4.1.2, when the time step of the simulated data increases, it will contain fewer details, making it challenging for Eq. (5.1) to find the appropriate parameters.

The two-dimensional data solves the following problem:

$$\{o^*, l^*\} = \arg\min_{o,l} \left\{ \mathrm{MSE} \left(\mathrm{SG} \left(\begin{bmatrix} \tilde{u}(\cdot, 1) \\ \tilde{u}(\cdot, 2) \\ \vdots \\ \tilde{u}(\cdot, m) \end{bmatrix}, o, l \right), \mathrm{GB}_{\kappa^2}(\tilde{u}) \right) \right\}, \tag{5.2}$$

where

$$\kappa^2 = \kappa \otimes \kappa = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

After finding the optimal parameters $\{o^*, l^*\}$, the smoothed data and its derivatives (at most to the o^{th} order) are calculated by

$$\hat{y} = \mathrm{SG}(\mathrm{GB}_\kappa(\tilde{y}), o, l) \tag{5.3}$$

for one-dimensional data, and

$$\hat{u} = \mathrm{SG}\left(\mathrm{GB}_{\kappa^2}(\tilde{u}), o, l\right), \tag{5.4}$$

for two-dimensional data. Algorithm 2 shows the pseudocode for the automatic Savitzky-Golay filter with Gaussian blur.

Algorithm 2: Automatic Savitzky-Golay Filter

```
**Input:** $\mathbf{U} \in \mathbb{R}^{n\times m}$ or $\mathbb{C}^{n\times m}$, $dt$, $dx$.
**Output:** partial derivatives $\mathbf{U}_t$, $\mathbf{U}_x$, $\mathbf{U}_{xx}$, $\cdots$.
1 $\mathbf{U}_{\mathrm{GB}}$ = Gaussian_Blur($\mathbf{U}$); // use Gaussian blurred data as the
    ground truth;
2 $(o_t^*, l_t^*) = \arg\min_{o,l}$ MSE (Savitzky-Golay($\tilde{\mathbf{U}}(t), o, l$), $\mathbf{U}_{\mathrm{GB}}$);
3 $(o_x^*, l_x^*) = \arg\min_{o,l}$ MSE (Savitzky-Golay($\tilde{\mathbf{U}}(x), o, l$), $\mathbf{U}_{\mathrm{GB}}$);
4 $\mathbf{U}_t$ = Savitzky-Golay($\mathbf{U}_{GB}, o_t^*, l_t^*$, derivative=1);
5 $\mathbf{U}_x$ = Savitzky-Golay($\mathbf{U}_{GB}, o_x^*, l_x^*$, derivative=1);
6 $\mathbf{U}_{xx}$ = Savitzky-Golay($\mathbf{U}_{GB}, o_x^*, l_x^*$, derivative=2);
7 $\vdots$
```

The Gaussian kernel in this work is generated by OpenCV's `getGaussianKernel`. For one-dimensional data, `numpy`'s `convolve` is used to take the convolution of the whole series data to implement the Gaussian blur. For two-dimensional data, `GaussianBlur` in OpenCV is applied to the entire dataset.

5.2 Results

To evaluate the performance of the automatic Savitzky-Golay filter with Gaussian blur (ASG) and `PyNumDiff`'s Savitzky-Golay filter (PSG), the data sizes are fixed, and the SNRs of the data are varied. The noisy data are created by adding Gaussian noise z to the data x:

$$\tilde{x} = x + z, \quad z \sim \mathcal{N}(0, \sigma_z^2), \tag{5.5}$$

which is the most common way to mimic naturally occurring random processes [111, 164, 165]. White noise serves as a meaningful baseline for assessing the algorithm's performance and robustness due to its uniformly distributed power spectral density.

In signal processing, the Additive white Gaussian noise (AWGN) [166] is a basic noise assumption in communication channels. Due to the Central Limit Theorem, independent random variables added together tend to form a Gaussian distribution, regardless of their original distributions. Gaussian distributions are tractable in the spatial and frequency domains, simplifying analysis and theoretical derivations, making them convenient for practical use [164]. For example, adding Gaussian noise to the data can simulate the data collected by a sensor with background thermal noise. Here, the noise mean is zero ($\mu_Z = 0$), and the noise standard deviation σ_Z only depends on the SNR because the data's standard deviation is fixed. To measure the uncertainty at each SNR value, 100 noisy datasets are generated, and MSE is employed to assess the smoothing method's performance. Section 2.5.3 has discussed that the Savitzky-Golay filter may poorly estimate values at boundaries (i.e., the first and last l points). The following MSE plots will display the ASG with and without boundaries in green and blue boxes, respectively.

5.2.1 One-variable functions

One-dimensional Gaussian function

The Gaussian function is a smooth curve with continuous derivatives of all orders without sharply changing points. The mathematical expression is

$$f(x) = a \exp\left(-\frac{(x-b)^2}{2c^2}\right). \tag{5.6}$$

It has a symmetric "bell curve" shape, see Fig. 5.1(a), with arbitrary real constants a, b, and a non-zero c. The parameter a determines the amplitude of the curve, b is the centre of the curve, and c controls the altitude and width of the "bell". Here, the simplest example is used, $a = 1$, $b = 0$ and $c^2 = 0.5$, i.e., $f(x) = exp(-x^2)$. The x domain is set from -2 to 2 with $\Delta x = 0.01$.

Figure 5.1 illustrates that a higher SNR leads to more accurate smoothed data and first-order derivatives for both methods. Particularly, when data contain noise, the MSE distribution for the ASG is always lower than that of the PSG. While Fig. 5.1(b) shows similar results in terms of smoothed data for these methods, the

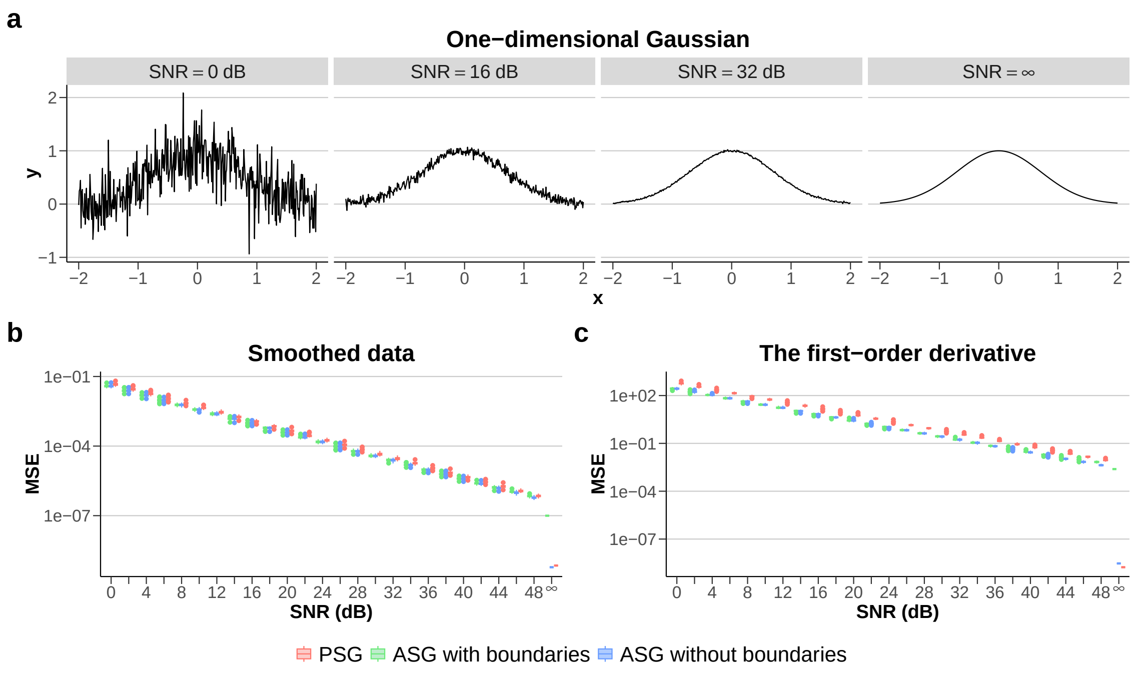


Figure 5.1: Evaluation results for the one-dimensional Gaussian function. With SNR increases, (a) shows the function curve, (b) and (c) illustrate MSE distributions with box plots of smoothed data and the first-order derivative using 100 generated datasets at each SNR value, respectively. The red boxes represent the `PyNumDiff`'s Savitzky-Golay filter (PSG). The green and blue boxes are the results of automatic Savitzky-Golay (ASG) with and without boundary points, respectively.

ASG outperforms PSG in calculating the first-order derivatives (see Fig. 5.1(c)). Furthermore, when the data are noiseless, MSEs of both the ASG without boundary and PSG are lower than 10^{-7}, representing the results from both methods are close to the ground truth. Overall, the ASG without boundary performs the best for smoothing and calculating first-order derivatives of one-dimensional Gaussian functions.

Linear chirp function

A chirp function is a sinusoidal waveform whose frequency increases or decreases over time. Due to the frequency changes, the chirp function is important in signal processing, and communication systems. For example, in radar and sonar systems, the variation in range and velocity of targets can be described as a chirp function.

Here, the linear chirp function is used:

$$f(t) = \cos\left(2\pi\left(f_0 t + \frac{f_1 - f_0}{2t_1}t^2\right)\right), \tag{5.7}$$

where f_0 is the frequency at $t = 0$, f_1 is the final frequency, and t_1 is the time it takes to sweep from f_0 to f_1. The time domain is from 0 to 5 with $\Delta t = 0.01$.

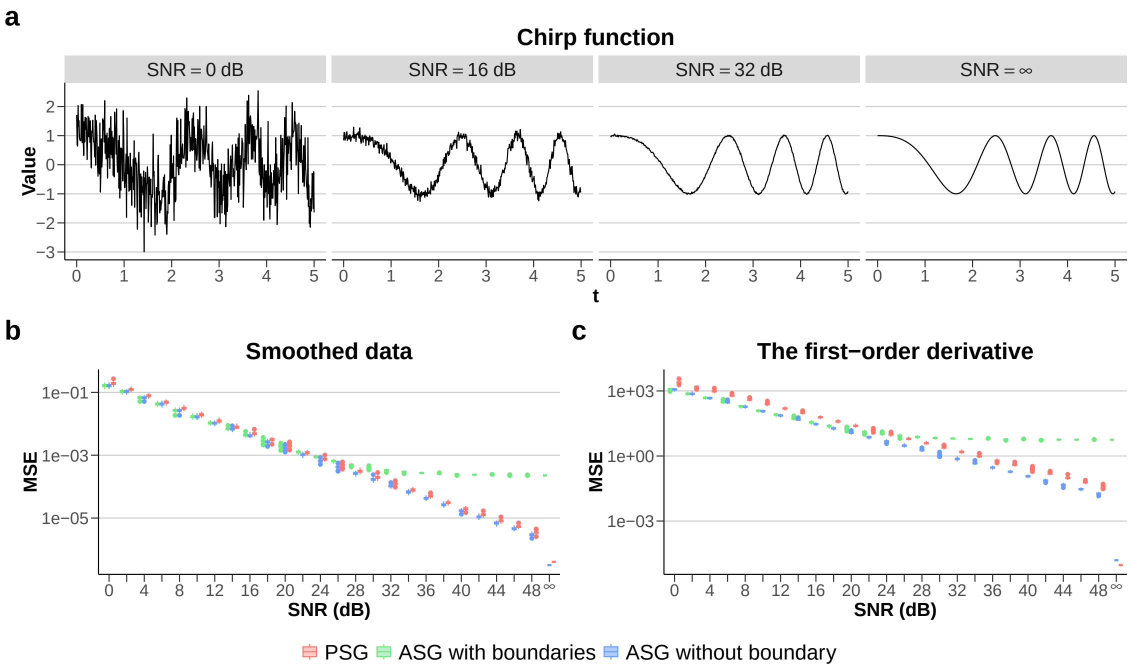


Figure 5.2: Evaluation results for the linear chirp function. With SNR increases, (a) shows the function curve, (b) and (c) illustrate MSE distributions with box plots of smoothed data and the first-order derivative using 100 generated datasets at each SNR value, respectively. The red boxes represent the `PyNumDiff`'s Savitzky-Golay filter (PSG). The green and blue boxes are the results of automatic Savitzky-Golay (ASG) with and without boundary points, respectively.

The chirp function is employed to evaluate the efficacy of the smoothing method in handling varying frequencies. As depicted in Fig. 5.2(a), the test signal frequency increases over time. In the presence of noise, both Fig. 5.2(b) and (c) demonstrate a linear decrease in MSEs for PSG and ASG without boundary. Notably, the ASG without boundary consistently yields more precise estimations with lower MSE distributions than that of the PSG. However, due to challenges associated with boundaries estimation, the MSE for ASG with boundaries is always higher than 10^{-4} and 5 in Fig. 5.2(b) and (c), respectively. Although the ASG and PSG use the same Savitzky-Golay filter function in Python, `scipy.signal.savgol_filter`, PSG applies trapezoidal integration to the first-order derivatives to estimate data, see Section 2.5.4. Due to this, the boundaries of PSG are smoother than those in ASG, leading to the decreasing MSE of PSG when the SNR is larger than 32 dB. Overall, the ASG without boundary performs the best among the tested smoothing methods for the linear chirp function.

Lorenz system

The Lorenz system is a set of three nonlinear ordinary differential equations, Eq. (5.8), describing the behaviour of a simplified model of atmospheric convection [167].

$$\begin{aligned}\frac{\mathrm{d}x_1}{\mathrm{d}t} &= 10(x_2 - x_1),\\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= x_1(28 - x_3) - x_2,\\ \frac{\mathrm{d}x_3}{\mathrm{d}t} &= x_1x_2 - \frac{8}{3}x_3,\end{aligned} \tag{5.8}$$

It is one of the simplest chaotic systems, sensitive to its initial conditions, causing the "butterfly effect", i.e., slight differences in initial conditions lead to drastically different outcomes over time. When solving the system using the Lorenz attractor and drawing in phase space, the shape may be seen to resemble a butterfly (see Fig. 5.3(a), SNR $= \infty$).

In this work, the `odeint` in Python's `scipy` is used to solve the Lorenz system. The evolution time is from -5 to 5 with $\Delta t = 0.01$, and the initial point is $x = -8$, $y = 8$ and $z = 27$.

The MSE distribution of the ASG without boundary and PSG in both Fig. 5.3(b) and (c) exhibits a linear decline as SNR increases. Particularly, the ASG without boundary demonstrates the lowest MSE in the boxplots. Additionally, the ASG with boundaries shows similar MSE distributions to the ASG without boundary when the SNR is less than 24 dB. However, for SNR greater than 24 dB, the MSE of the ASG with boundaries gradually decreases and stabilises at approximately 0.033 in Fig. 5.3(b) and 800 in Fig. 5.3(c), due to errors in estimating boundaries points. Overall, the ASG without boundary proves to be the most effective method for smoothing

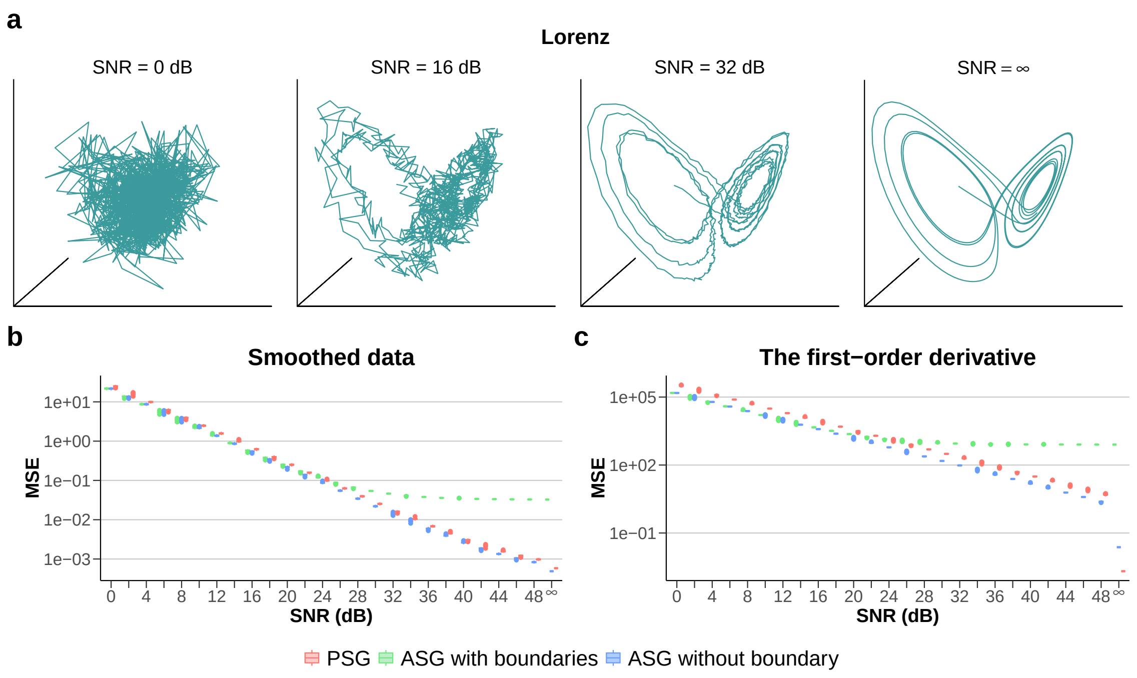


Figure 5.3: Evaluation results for the Lorenz system. With SNR increases, (a) shows the function curve, (b) and (c) illustrate MSE distributions with box plots of smoothed data and the first-order derivative using 100 generated datasets at each SNR value, respectively. The red boxes represent the `PyNumDiff`'s Savitzky-Golay filter (PSG). The green and blue boxes are the results of automatic Savitzky-Golay (ASG) with and without boundary points, respectively.

data and calculating first-order derivatives.

5.2.2 Two-variable functions

Two-dimensional Gaussian function

As the one-dimensional Gaussian function has been used in Section 5.2.1, the two-dimensional Gaussian function will also be employed to evaluate the performance of the automatic Savitzky-Golay filter with Gaussian blur. The two-dimensional Gaussian function is defined as

$$f(x, y) = A \exp\left(-\left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0)^2}{2\sigma_y^2}\right)\right) \tag{5.9}$$

where A represents the height of the curve's peak, x_0 and y_0 determine the centre of the curve, σ_x and σ_y controls the spreads of the shape. Here, $A = 1$, $x_0 = y_0 = 0$, and $\sigma_x^2 = \sigma_y^2 = 0.5$.

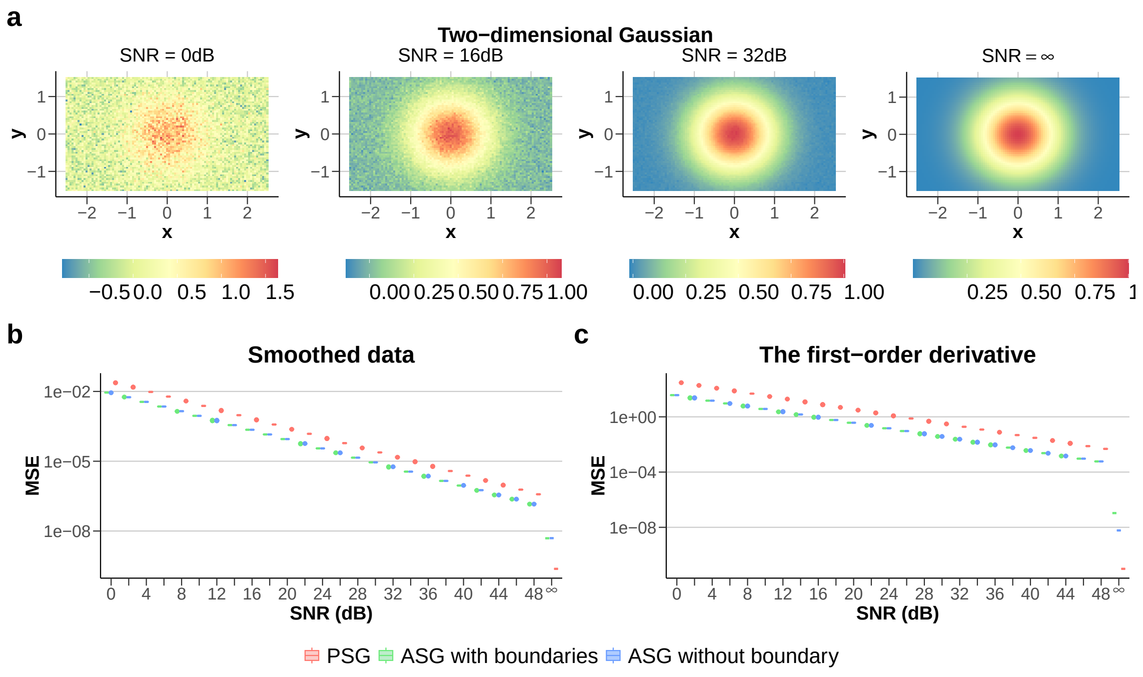


Figure 5.4: Evaluation results for the two-dimensional Gaussian function. With SNR increases, (a) shows the function curve, (b) and (c) illustrate MSE distributions with box plots of smoothed data and the first-order derivative using 100 generated datasets at each SNR value, respectively. The red boxes represent the `PyNumDiff`'s Savitzky-Golay filter (PSG). The green and blue boxes are the results of automatic Savitzky-Golay (ASG) with and without boundary points, respectively.

Figure 5.4 illustrates that a higher SNR leads to smaller MSE in smoothed data and first-order derivatives for both methods. Particularly, when data contain noise, the MSE distribution for the ASG is always lower than that of the PSG. Figure 5.4(b) and (c) illustrate that both ASG methods always have lower MSE distributions than the PSG, indicating that the ASG methods outperform the PSG. However, when data is noiseless, PSG has a smaller MSE than the other two.

Heat equation

The heat equation is a second-order PDE modelling a wide range of phenomena related to heat transfer. It is defined as

$$u_t = a u_{xx}, \quad 0 < x < L, \quad t > 0 \tag{5.10}$$

where a is a positive constant. The solution of the heat equation provides the heat behaviour in the given systems, such as the diffusion of temperature in solids and the cooling of objects in fluids. The heat equation has been widely applied. For example, the Schrödinger equation in quantum mechanics can be regarded as the heat equation in imaginary time. Combined with Fourier theory, the heat equation can describe the thermal diffusivity in polymers. The heat equation is linked by the Fokker-Planck equation and is connected with random walks and Brownian motion.

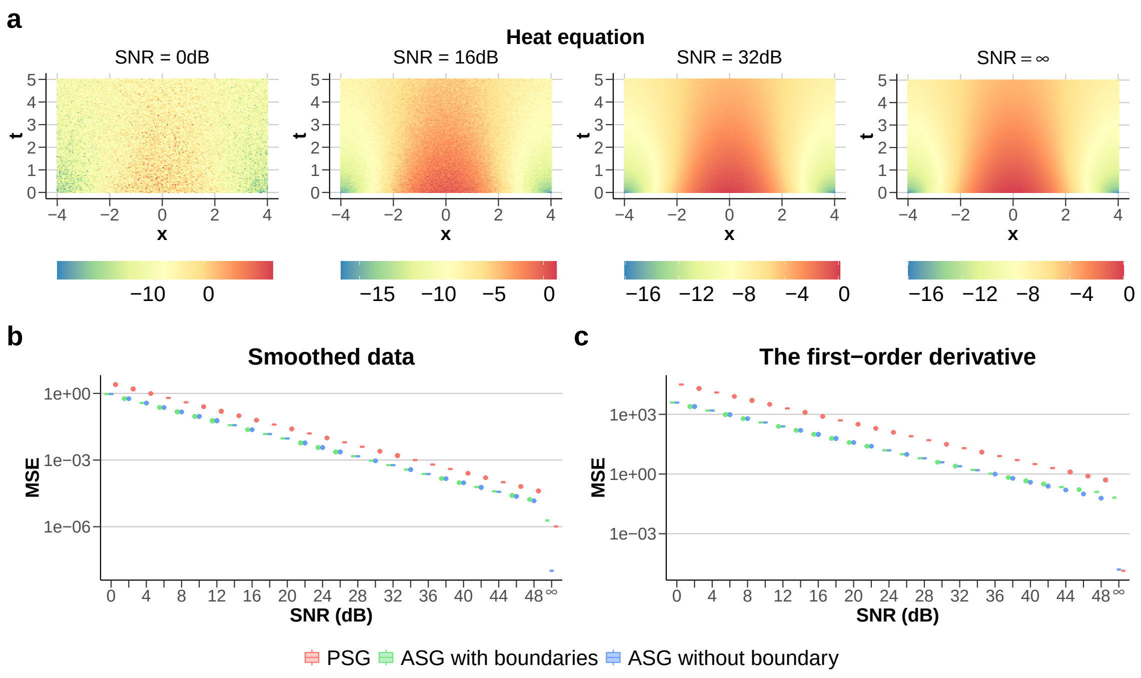


Figure 5.5: Evaluation results for the heat equation. With SNR increases, (a) shows the function curve, (b) and (c) illustrate MSE distributions with box plots of smoothed data and the first-order derivative using 100 generated datasets at each SNR value, respectively. The red boxes represent the `PyNumDiff`'s Savitzky-Golay filter (PSG). The green and blue boxes are the results of automatic Savitzky-Golay (ASG) with and without boundary points, respectively.

Here, the `odeint` in `scipy` with the spectral method is used to solve the heat equation. The solution is shown in Fig. 5.5(a) with $x \in [-4, 4]$, $\Delta x = 0.01$ and $t \in [0, 5]$, $\Delta t = 0.01$, and the initial function is x^2. The periodic boundary condition is applied when solving this PDE.

The box plots in Fig. 5.5(b) and (c) illustrate that all methods have a declining trend when SNR increases. Both ASG methods have similar accuracy and outperform PSG when data contain noise. When data is noiseless, the ASG without boundary is the best for smoothing data, while the PSG performs the best at calculating the first-order derivative.

Burgers equation

Burgers equation is a nonlinear PDE and can be derived from the Navier-Stokes equation for the velocity field by dropping the pressure gradient term. Unlike the Navier-Stokes equation, the Burgers equation does not behave with turbulence and can transform to a linear form via the Cole-Hopf transformation [168]. It is one of the simplest PDEs for understanding fluid flow behaviour. The equation has a nonlinear convection term coupled with a linear diffusion term:

$$u_t = -uu_x + \nu u_{xx} \tag{5.11}$$

where ν is the diffusion coefficient and is set to 0.1 here. Moreover, $x \in [-8, 8]$ with $\Delta x = 0.0625$, $t \in [0, 10]$ with $\Delta t = 0.1$, and the initial function is $\exp(-(x-2)^2)$. The numerical solver is `odeint` in Python's `scipy` with the spectral method Fourier spectral method, and the periodic boundary condition is employed [169, p.63-69].

Figure 5.6(b) and (c) show that the MSE values for the ASG without boundary and PSG decrease as SNR increases. The MSE of the ASG with boundaries slowly declines when SNR $>$ 34 dB in Fig. 5.6(b) and 24 dB in Fig. 5.6(c). Furthermore, when data contain noise, the ASG without boundary outperforms others. However, when data is noiseless, the PSG has the lowest MSE.

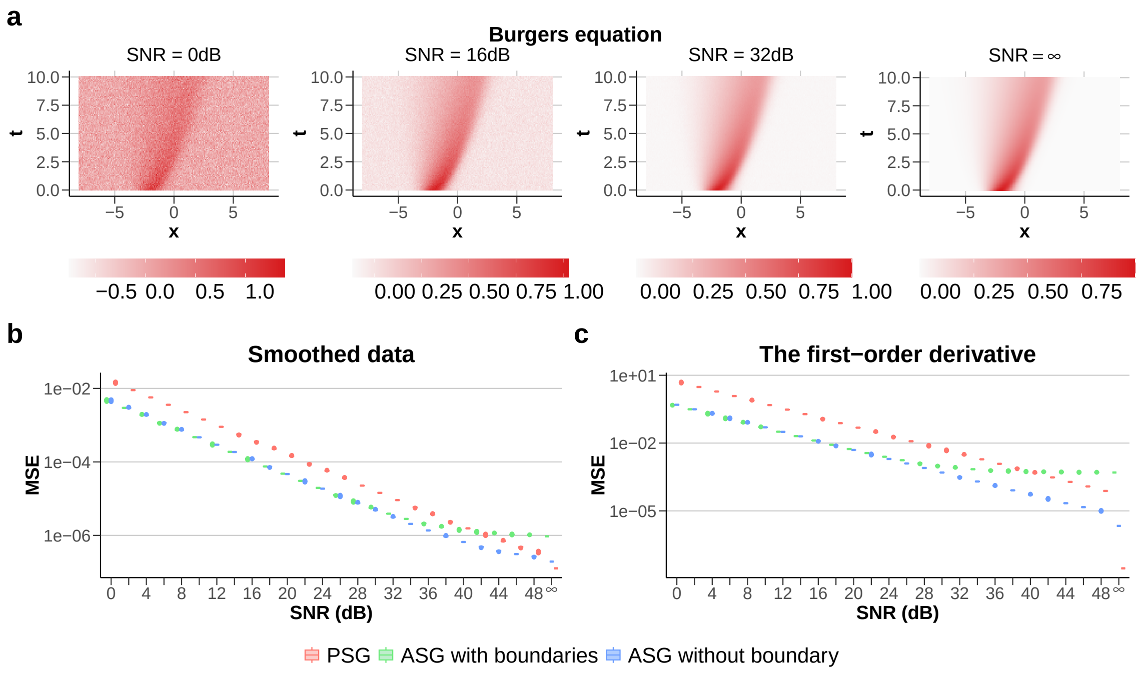


Figure 5.6: Evaluation results for the Burgers equation. With SNR increases, (a) shows the function curve, (b) and (c) illustrate MSE distributions with box plots of smoothed data and the first-order derivative using 100 generated datasets at each SNR value, respectively. The red boxes represent the `PyNumDiff`'s Savitzky-Golay filter (PSG). The green and blue boxes are the results of automatic Savitzky-Golay (ASG) with and without boundary points, respectively.

5.3 Summary

The previous tests evaluate the estimation accuracy of the PSG and ASG with and without boundaries. The MSE boxplots are employed to measure the differences between estimated data (including smoothing data and the first-order derivatives) and their ground truth at different SNRs.

- The ASG without boundary illustrates the best performance from Fig. 5.1 to 5.6.
- Interestingly, both ASG methods have similar MSE distributions in smoothing Gaussian functions. However, in the cases of the chirp (Fig. 5.2), Lorenz (Fig. 5.3) and Burgers (Fig. 5.6), the MSE values of the ASG with boundaries maintain at a certain value when SNR is larger than 28 dB. This is due to the lack of data on the Savitzky-Golay filter's implementation of polynomial estimates at boundary points.
- In the noiseless two-dimensional Gaussian function (Fig. 5.4), the PSG has the lowest MSE distribution. However, the other two methods also illustrate accurate results because the MSE values are smaller than 10^{-7}, which is the default floating comparing values in some numerical computing packages, such as `glmnet` in R and `numpy` in Python.
- The biggest advantage of the ASG is that it can accurately and automatically smooth data and calculate the derivatives. As users do not know the ground truth, the automatic method helps them overcome the parameter-tuning process. Although the estimation may have problems at boundary points, the ASG without boundary can be used, when data sizes are sufficient.

In the next chapter, this automatic method will be used to calculate the partial derivatives of the PDE data.

CHAPTER 6

Automating the Discovery of Partial Differential Equations in Dynamical Systems

The review of the SINDy framework in Chapter 3 figures out a challenge in identifying dynamical systems: developing an automated algorithm to identify PDEs with minimal manual intervention, streamlining the process, and improving its applicability across diverse scientific domains. This chapter demonstrates the performance of a novel method developed from Automatic Regression for Governing Equations (ARGOS) framework [1] to identify canonical PDEs. ARGOS assumes the underlying system is unknown, automates the fine-tuning of parameters for numerical differentiation (the Savitzky-Golay filter), and leverages sparse regression with bootstrap confidence intervals to select active terms from the candidate library. To automatically identify PDEs, ARGOS with the Recurrent Adaptive Lasso (ARGOS-RAL) is proposed. This extension of the ARGOS framework employs only sparse regression to identify equations rather than engaging in large-scale bootstrapping.

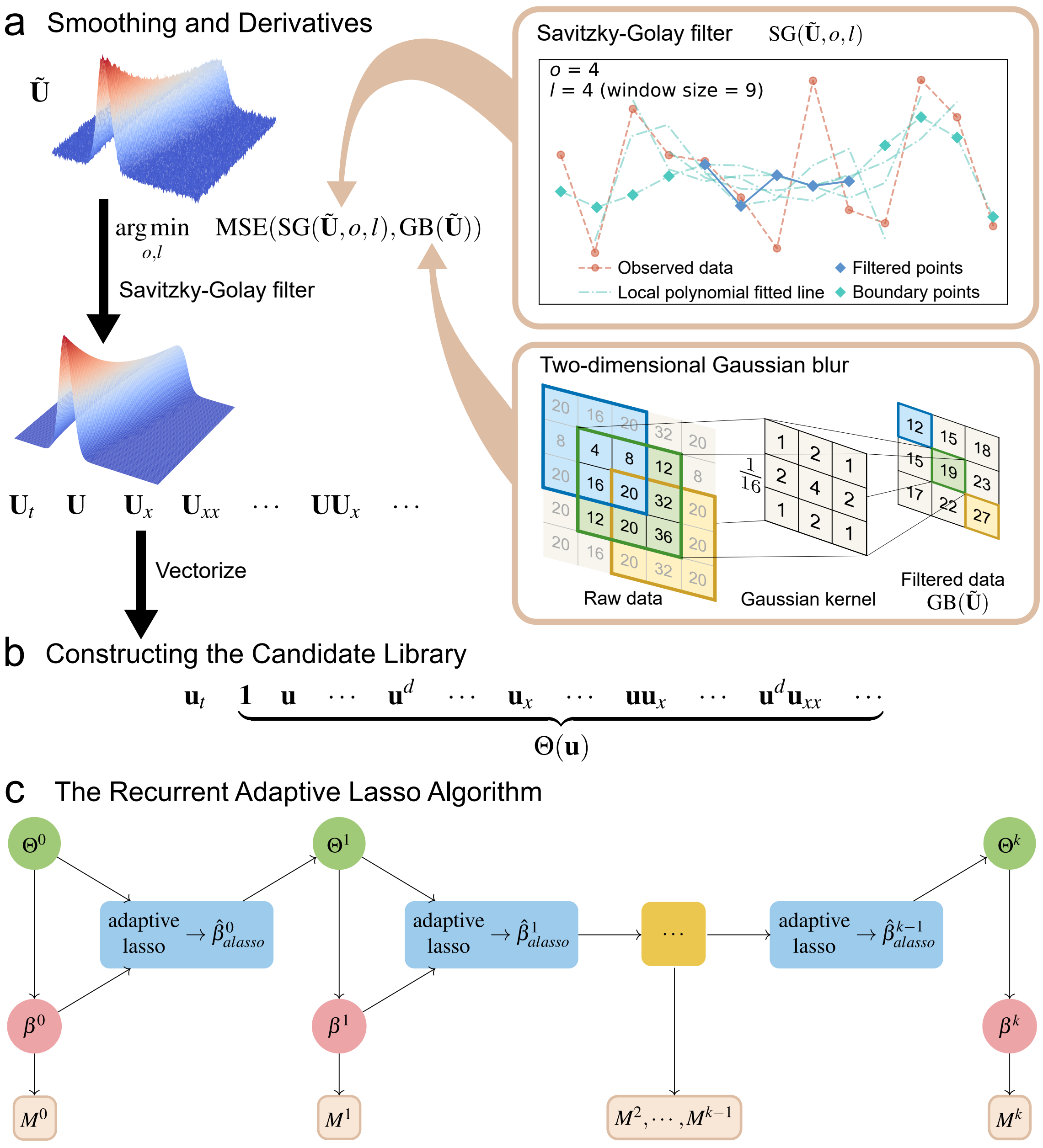
To compare the success rates and robustness of different methods, i.e., ARGOS-RAL, Sequential Threshold Ridge Regression (STRidge) [4], and backward stepwise selection with knowing the number of active terms (shortly "backward stepwise selection"), the performances are evaluated through a series of three numerical tests, each designed to assess its ability to identify canonical PDEs across diverse fields, including biology, neuroscience, fluid mechanics, and quantum mechanics. The first test explores the algorithm's resilience against varying noise levels by altering the SNR in Gaussian random noise integrated into the PDE solutions, which is crucial for understanding the robustness of these methods 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 accurate identification of the underlying equation. The final evaluation assesses the algorithm's ability to process datasets characterised by significant noise, challenging its limits and practical applicability in scenarios where data quality is compromised.

6.1 Method

6.1.1 Overview of the ARGOS-RAL Framework

Section 3.3.1 has interpreted details about how PDE-FIND constructs the candidate library. ARGOS framework employs the same way to create the library for identifying PDEs. Hence, the aim is to estimate the unknown mapping $\mathbf{F}(\cdot)$ in Eq. (3.11) with sparse regression by constructing a comprehensive library of potential terms and assuming that only a few of them are active [1, 3, 4]. Figure 6.1 illustrates the process of ARGOS-RAL, which combines automatic numerical differentiation (Chapter 5), building candidate library (Section 3.3.1), and the new proposed recurrent adaptive lasso.

A key step in constructing the candidate library in Eq. (3.14) is the numerical calculation of derivatives, see Fig. 6.1(a) and (b). The Savitzky-Golay filter [78] has become a favoured solution in system identification for signal smoothing and differentiation [4, 75]. 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 [83]. Chapter 5 illustrates more details and the evaluation performances of this automatic method.



a Smoothing and Derivatives
$\tilde{\mathbf{U}}$
$\arg\min_{o,l} \mathrm{MSE}(\mathrm{SG}(\tilde{\mathbf{U}},o,l),\mathrm{GB}(\tilde{\mathbf{U}}))$
Savitzky-Golay filter
$\mathbf{U}_t$ $\mathbf{U}$ $\mathbf{U}_x$ $\mathbf{U}_{xx}$ $\cdots$ $\mathbf{U}\mathbf{U}_x$ $\cdots$
Vectorize
Savitzky-Golay filter $\mathrm{SG}(\tilde{\mathbf{U}},o,l)$
o = 4
l = 4 (window size = 9)
Observed data
Filtered points
Local polynomial fitted line
Boundary points
Two-dimensional Gaussian blur
Raw data
Gaussian kernel
$\frac{1}{16}$
Filtered data $\mathrm{GB}(\tilde{\mathbf{U}})$
b Constructing the Candidate Library
$\mathbf{u}_t \quad \underbrace{\mathbf{1} \quad \mathbf{u} \quad \cdots \quad \mathbf{u}^d \quad \cdots \quad \mathbf{u}_x \quad \cdots \quad \mathbf{u}\mathbf{u}_x \quad \cdots \quad \mathbf{u}^d\mathbf{u}_{xx} \quad \cdots}_{\Theta(\mathbf{u})}$
c The Recurrent Adaptive Lasso Algorithm
Θ^0
adaptive lasso $\to \hat{\beta}^0_{alasso}$
Θ^1
adaptive lasso $\to \hat{\beta}^1_{alasso}$
$\cdots$
adaptive lasso $\to \hat{\beta}^{k-1}_{alasso}$
Θ^k
β^0
β^1
β^k
M^0
M^1
$M^2,\cdots,M^{k-1}$
M^k

Figure 6.1: 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. The data $\tilde{\mathbf{U}}$ is collected and applied to automatic Savitzky-Golay filtering with Gaussian blur to calculate the smoothed $\mathbf{U}$ and its partial derivatives. Following this, the smoothed data, all partial derivatives, and other related terms are vectorised to construct the candidate library. The recurrent adaptive lasso is then employed to identify the active features in the library, and the unbiased coefficients of the identified model are estimated using ordinary least squares regression.

6.1.2 Sparse Regression with the Recurrent Adaptive Lasso

The adaptive lasso is a two-step method [1, 64], which has been introduced in Section 2.4.4. The first step uses the ordinary least squares (OLS) to obtain unbiased estimates and derive the weights $w = |\hat{\beta}_{ols}|^{-\gamma}$, where $\gamma > 0$. The second step employs optimisation solver, such as `glmnet` package [61] in R, to obtain the estimated coefficients $\hat{\beta}_{\text{alasso}}$ by solving the problem

$$\hat{\beta}_{\text{alasso}} = \arg\min_{\beta} \|\mathbf{u}_t - \mathbf{\Theta}\beta\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j| , \tag{6.1}$$

where λ is a nonnegative regularisation parameter controlling the amount of shrinkage applied to the coefficients of the predictors. 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, the regularisation parameter λ is determined by employing the Pareto curve, which illustrates the optimal trade-off between the regularisation penalty and the model residuals [67, 68, 70]. More details about the Pareto curve can be found in Section 2.4.6. Although cross-validation is an alternative method, Cortiella *et al.* [31] have shown that it finds a λ optimised for prediction, potentially finding extra features for the equation.

The adaptive lasso regression often detects more terms than those in the true system. To improve parsimony, Egan *et al.* [1] suggested combining the adaptive lasso with bootstrap techniques to identify ODEs. Similarly, Cortiella *et al.* [31] adopted a modified version of the multi-step adaptive lasso [137] 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.* [31] 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.

Algorithm 3: The recurrent adaptive lasso with Pareto curve and AIC

```
   Input: $\boldsymbol{\Theta}(u) \in \mathbb{R}^{(n \cdot m) \times p}$ 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: $\hat{\beta}$
 1 for $\gamma$ in 1:5 do
 2     $\mathcal{J}^{(\gamma,0)} = \text{NULL}$; // initialize $\mathcal{J}$;
 3     k = 1; // iteration counter;
 4     $\mathcal{J}^{(\gamma,k)} = \{1, 2, \cdots, p\}$; // selected columns from $\boldsymbol{\Theta}$;
 5     while $\mathcal{J}^{(\gamma,k)} \neq \mathcal{J}^{(\gamma,k-1)}$ do
 6         $w^{(\gamma,k)} = \left(\arg\min_{\beta_{\mathcal{J}^{(\gamma,k)}}} \|u_t - \boldsymbol{\Theta}(u)_{\mathcal{J}^{(\gamma,k)}} \beta_{\mathcal{J}^{(\gamma,k)}}\|_2^2\right)^{-\gamma}$; // ols
              weights;
 7         $\hat{\beta}^{(\gamma,k)} =$
            $\arg\min_{\beta_{\mathcal{J}^{(\gamma,k)}}} \|u_t - \boldsymbol{\Theta}(u)_{\mathcal{J}^{(\gamma,k)}} \beta_{\mathcal{J}^{(\gamma,k)}}\|_2^2 + \lambda^* \sum_{j=1}^{p} w_j^{(\gamma,k)} |\beta_{j \cdot \mathcal{J}^{(\gamma,k)}}|$;
           // $\lambda^*$ is the optimal point on the Pareto curve;
 8         $\mathcal{A}^{(\gamma,k)} = \text{AIC}(\hat{\beta}^{(\gamma,k)})$;
 9         $\mathcal{J}^{(\gamma,k)} = \left\{j : \hat{\beta}_j^{(\gamma,k)} \neq 0\right\}$; // select active terms;
10         $k = k + 1$;
       end
   end
11 $\mathcal{J}^* = \mathcal{J}^{(\gamma^*,k^*)}$ where $(\gamma^*, k^*)$ is the index of the minimum $\mathcal{A}$;
12 $\hat{\beta} = \arg\min_{\beta_{\mathcal{J}^*}} \|u_t - \boldsymbol{\Theta}(u)_{\mathcal{J}^*} \beta_{\mathcal{J}^*}\|_2^2$;
```

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 Fig. 6.1(c). At each iteration, it removes terms whose coefficients the adaptive lasso penalised to zero (see Algorithm 3 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 on the regularisation of 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 zero [59, 64]. Meanwhile, it relaxes the regularisation 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 $\boldsymbol{\Theta}$, all candidate models are recorded, and their AIC values are calculated 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 [49, 51]. The pseudocode of the recurrent adaptive lasso is shown in Algorithm 3.

6.1.3 Enabling regression analysis for complex number

In the following tested PDEs, quantum harmonic oscillator and nonlinear Schrodinger equations are complex-valued PDEs, which refer to real and imaginary numbers. Most statistical analysis focuses on the real number, while the complex number is not considered. Although the least squares estimate allows one to calculate the complex number, most functions in R programming require real-number data. When performing a sparse regression on complex numbers, the regression is transformed from complex to real numbers. The complex-number regression can be written as a pair of models for each observation, $y_i = y_i^R + iy_i^I$, where the superscript y^R represents the real-number part of the complex number y, the superscript y^I means the imaginary part, the normal i is the imaginary number, the subscript y_i represents the ith observation. Due to this, the equation can be reformed as

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

This equation can be divided into two equations with real and imaginary parts

$$y_i^R = \beta_0^R + \sum_{j=1}^{p} x_{ij}^R \beta_j^R - \sum_{j=1}^{p} x_{ij}^I \beta_j^I + \epsilon^R, \tag{6.2a}$$

$$y_i^I = \beta_0^I + \sum_{j=1}^{p} x_{ij}^I \beta_j^R + \sum_{j=1}^{p} x_{ij}^R \beta_j^I + \epsilon^I. \tag{6.2b}$$

Based on Eq. (6.2), the dataset can be organised 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, the recurrent adaptive lasso, STRidge, and backward stepwise selection can be implemented on the re-posited Y and $\mathbf{X}$.

6.2 Results

6.2.1 Varying Noise Levels and Sample Sizes

The following noisy and noiseless tests compare the performance of ARGOS-RAL, STRidge [4], and backward stepwise selection in identifying canonical PDEs under various SNRs and sample sizes (N). Figure 6.2 demonstrates the impact of introducing increasing levels of Gaussian random noise into the solution of the Burgers equation, effectively decreasing the SNR values.

In the evaluation of noise-contaminated data, similar to the tests in Chapter 5, σ_Z in SNR's expression, Eq. (2.50), is varied to span a broad range of noise levels, facilitating a comprehensive evaluation of the efficacy of ARGOS-RAL, STRidge, and backward stepwise selection in identifying various PDEs under different noise conditions. For this purpose, datasets with SNRs set at $\{0, 2, \cdots, 58, 60, \infty\}$ [1] are generated, each comprising paired elements $\{\mathbf{u}_t, \mathbf{\Theta}(\mathbf{u})\}$. This approach can examine the robustness of each identification method as it copes with varying noise levels.

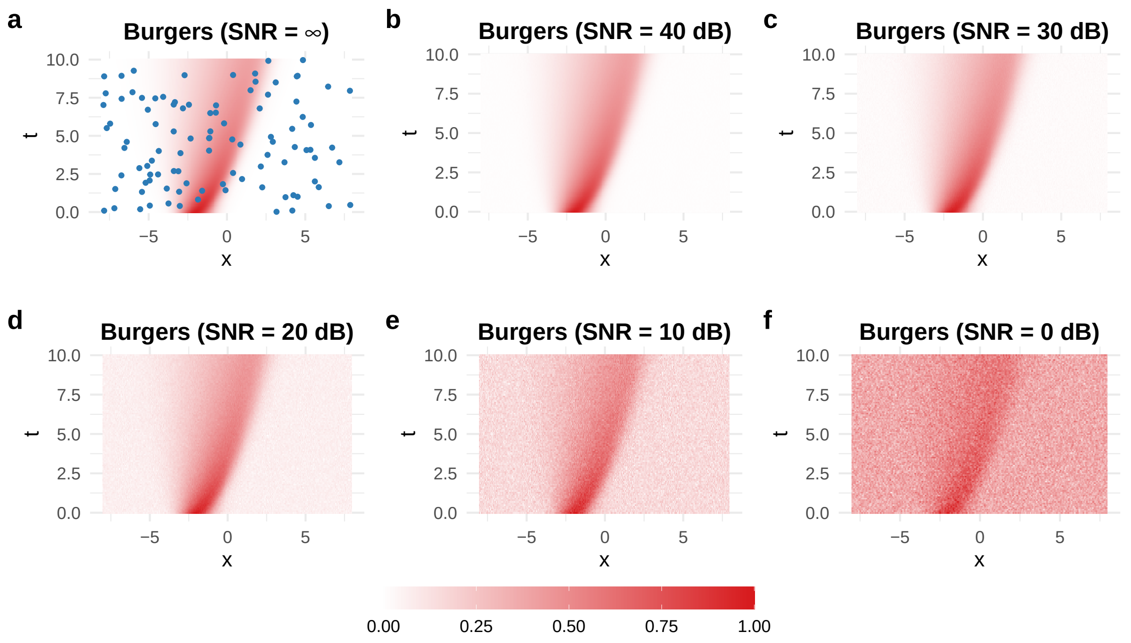


Figure 6.2: Influence of SNR on the Burgers equation dataset. (a) Noiseless data points (blue) serve as a reference for evaluating the impact of sample size on PDE identification accuracy. (c-f) Noisy datasets are generated by adding Gaussian noise at SNR levels of 40 dB, 30 dB, 20 dB, 10 dB and 0 dB, respectively, to comprehensively characterise the system's behaviour under varying noise conditions.

In investigating sample size, N, the objective is to determine the smallest number of samples needed to reliably identify PDEs with a success rate exceeding 80%. To achieve this, a full dataset for each PDE is first generated by calculating partial derivatives and assembling a candidate library as described in Eq. (3.14). The size of this dataset, denoted as N, varies depending on the specific PDE under consideration. Specifically, $\mathrm{N} = 10^4$ for the advection-diffusion, Burgers, and cable equations, $\mathrm{N} = 10^{4.8}$ for the heat and Korteweg-De Vries (KdV) equations, $\mathrm{N} = 10^5$ for the transport, Navier-Stokes, and reaction-diffusion equations, and $\mathrm{N} = 10^{5.2}$ for the Kuramoto-Sivashinsky (KS), quantum harmonic oscillator (QHO), and nonlinear Schrodinger (NLS) equations. Next, smaller subsets of size N are randomly sampled from the full dataset, i.e., points are randomly sampled on x and t regular grids, where N is chosen from a $\log_{10}$ spaced grid: $N = 10^2, 10^{2.2}, \cdots, \mathrm{N}$ [1], see the blue points in Fig. 6.2(a). By applying the PDE identification methods to these subsets and evaluating their success rates, the smallest sample size required for reliable identification of each PDE can be determined.

6.2.2 Success Rates in Identifying Canonical PDEs

To evaluate the impact of different SNRs and data sizes on the method, the uncertainty of model identification caused by random sampling is measured. To accomplish this, 100 unique datasets corresponding to different SNRs and N values are created at each point on the grid. For each dataset, the identification accuracy is quantified with the success rate, $\eta = \#\text{correct}/100$, where #correct represents the number of times the model correctly identifies all active terms, i.e., the identified equation has no missing or extra terms. This 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. Figures from 6.3 to 6.16 illustrate these results for the Burgers, Cable, Navier-Stokes, reaction-diffusion, transport, heat, advection-diffusion, KdV, KS, QHO, and NLS equations.

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 Fig. 6.3, 6.4, 6.5, 6.6 and 6.9). However, accurately detecting specific equations requires a high SNR, particularly for the QHO, KdV, transport, and diffusion equations. The KdV and KS equations, which involve third-order and fourth-order partial derivatives, respectively, present challenges due to the significant biases in numerical approximations of these derivatives [142], resulting in datasets unsuitable for system identification with sparse regression. Sparse regression within the real number domain for the QHO and NLS is implemented by applying the transformation shown in Section 6.1.3. This transformation expands the candidate library $\mathbf{\Theta}$ from $nm \times p$ to $2nm \times 2p$, effectively quadrupling its size and potentially leading to high correlations between the variables in $\mathbf{\Theta}$. 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 [64].

Figures from 6.3 to 6.16 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 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. ARGOS-RAL is thus demonstrated 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 [4] 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 Fig. 6.5(b) and 6.6(b). Moreover, STRidge with $d_{tol} = 2$ is more proficient in identifying the quantum harmonic oscillator and the transport equation with an SNR lower than 52 dB, see Fig. 6.13(b) and 6.7(b), 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 Fig. 6.7 and 6.8.

The following parts will show each example in detail, including how to solve these PDEs and how the success rate changes. The Fourier spectral method is employed to solve the Burgers, Cable, reaction-diffusion, advection-diffusion, transport, diffusion, and Kuramoto-Sivashinsky (KS) systems; thus, their boundary conditions are the periodic boundary conditions.

Burgers equation

Burgers equation can be derived from the Navier-Stokes equation for the velocity field by dropping the pressure gradient term. Hence, the Burgers equation ignores the effects of turbulence in the system and can be converted to linear form by the Cole-Hopf transformation [168]. Here, the Fourier spectral method [169, pg.63-69] with *ode45* in MATLAB is used to solve the Burgers equation. The spatial points $x \in [-8, 8]$ with $m = 256$, the time-steps $t \in [0, 10]$ with $n = 101$, and the initial condition is a Gaussian function: $\exp\left(-(x+2)^2\right)$. Burgers equation is:

$$u_t = -uu_x + 0.1u_{xx}. \tag{6.3}$$

In analysing noisy effects, Fig. 6.3(b), backward stepwise selection illustrates the best results when identifying the Burgers equation for each SNR value. It begins to identify the correct equation when SNR is 2 dB, although the success rate is 3%. As SNR increases, the backward stepwise selection is more likely to find the correct equation and reach a 100% success rate at a 24 dB SNR. Furthermore, ARGOS-RAL starts identifying the correct equation when SNR is larger than 10 dB, and the success rate increases to 100% at SNR = 22 dB. However, STRidge with different d_{tol} has a fluctuating performance.

In sampling data analysis, Fig. 6.3(c), backward stepwise selection outperforms others. The performance of ARGOS-RAL ranks second, which has a 73% success rate at N = 10^2 and increases to 100% at N = $10^{3.2}$. The success rates of STRidge fluctuate below 25%.

Cable equation

The cable equation describes the electrical behaviour of a nerve axon or any other cable-like structure in biological systems, especially the electrical circuit of current flow and voltage change both within and between neurons. The equation is derived

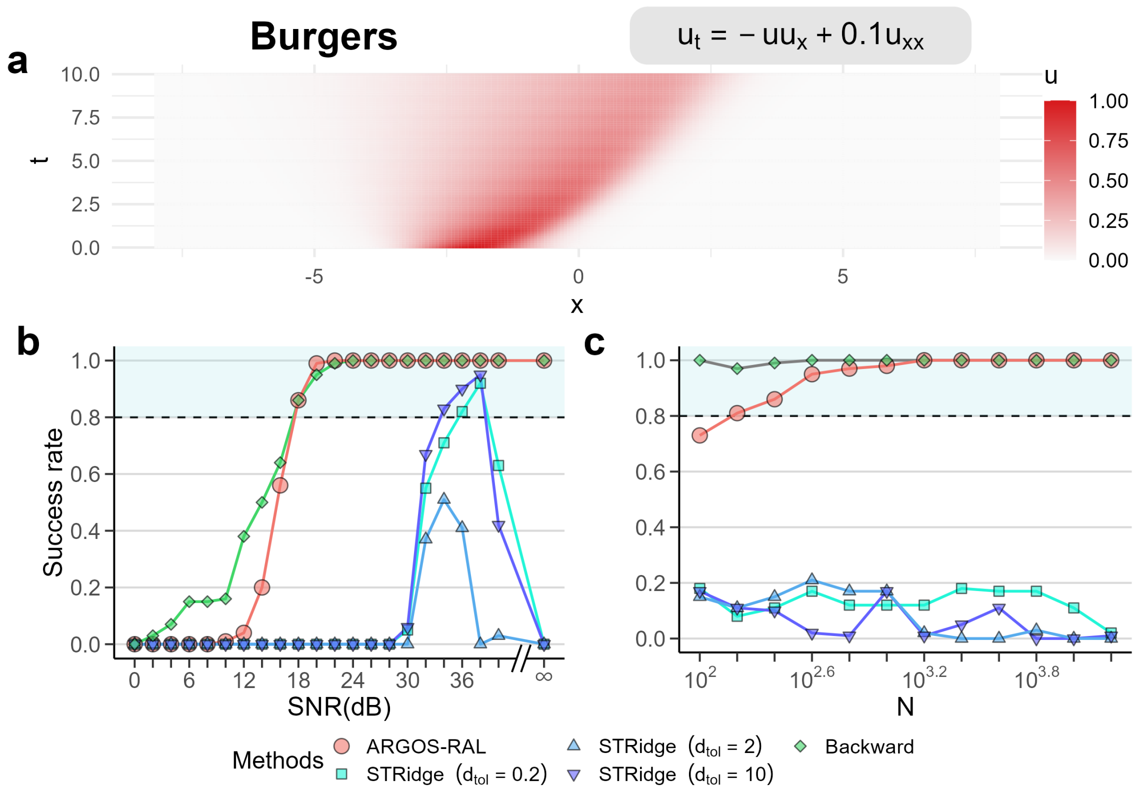


Figure 6.3: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying Burgers equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

from a circuit model of the membrane and its intracellular and extracellular space:

$$\lambda^2 \frac{\partial^2 V}{\partial x^2} = \tau \frac{\partial V}{\partial t} + V \quad \text{where} \quad \lambda = \sqrt{\frac{r_m}{r_e + r_a}} \quad \text{and} \quad \tau = r_m c_m, \tag{6.4}$$

where λ and τ are all set to be one in this test. It is an important PDE in biophysical studies, understanding how electrical signals are affected in diseases and disorders. Identifying the cable equation helps researchers diagnose these negative conditions by checking changes in the capacitances c_m, resistances r_m, and axial resistance r_a, r_e. The cable equation is solved using *odeint* in Python by the Fourier spectral method with $x \in [-4, 4]$ with $\Delta x = 0.1$, $t \in [0, 5]$ with $\Delta t = 0.01$, and the initial condition is a Gaussian function: $\exp(-x^2)$.

In SNR and N tests, Fig. 6.4(b) and (c), backward stepwise selection performs

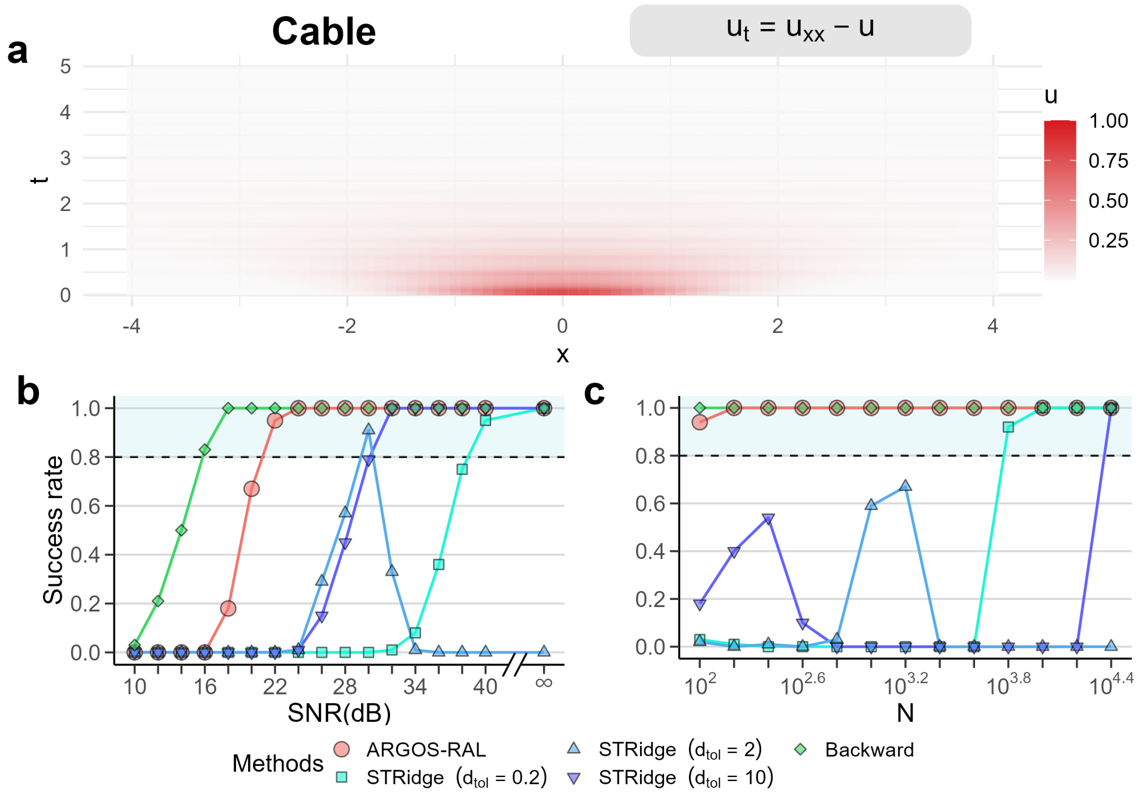


Figure 6.4: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the cable equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 10 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

the best, following ARGOS-RAL at the second. As the SNR increases, Fig. 6.4(b), methods (except STRidge with $d_{tol} = 2$) more frequently identify the correct cable equation. Particularly, backward stepwise selection maintains a 100% success rate when SNR is larger than 16 dB. The success rate of ARGOS-RAL exceeds 80% when SNR is larger than 22 dB. All STRidge methods require SNR greater than 24 dB to identify the correct equation. Figure 6.4(c) shows that backward stepwise selection and ARGOS-RAL have consistent success rates. When there are only 100 sample points, backward stepwise selection can always (100%) identify the cable equation, and ARGOS-RAL's success rate is 94%. The success rates of STRidge with $d_{tol} = 2$ and 10 vary dramatically, while the success rate of STRidge with $d_{tol} = 0.2$ maintains at 100% when the sample size is larger than 10^4.

Navier-Stokes equation

The Navier-Stokes equations describe the two-dimensional fluid flow past a circular cylinder. The Immersed Boundary Projection Method (IBMP) is employed to simulate the Navier-Stokes equations [170, 171]:

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \triangledown)\omega = \frac{1}{\mathrm{Re}} \triangledown^2 \omega, \quad \mathbf{u} = (u, v), \quad \mathrm{Re} = 100 \tag{6.5}$$

$$\omega_t = 0.01\omega_{xx} + 0.01\omega_{yy} - u\omega_x - v\omega_y \tag{6.6}$$

Here are some parameter settings: the Reynolds number is 100, $x \in [0, 9]$ with 449 space points, $y \in [0, 4]$ with 199 space points, and $t \in [300, 330]$ with 151 time points. Hence, the simulated data contains approximately 13.5 million points. The boundary condition for the IBPM is the no-slip boundary condition.

Constructing the candidate library $\mathbf{\Theta}$ poses computational challenges for this large dataset because it costs too much RAM space and computing time (exceeding Durham Hamilton HPC usage limits) to calculate the derivatives (∂t, ∂x and ∂y) and find the Savitzky-Golay parameters. Thus, 300,000 points are randomly sampled, consisting of 60 different snapshots and 5,000 spatial points in each snapshot within the red rectangular area, see Fig. 6.5(a). Algorithm 2 is then applied to calculate the derivatives for these sampled points. Therefore, the derivatives and candidate library are calculated within the subset of approximately 1.67% of all data.

Figure 6.5 shows backward stepwise selection is the best identification method, which always has the highest success rate. However, if users do not know the active terms of the equation, ARGOS-RAL will be the best alternative because it consistently identifies the correct equation with high frequencies at each SNR value, Fig. 6.5(b), and sample size, Fig. 6.5(c). Although STRidge with $d_{tol} = 10$ has robust results and outperforms ARGOS-RAL at SNR = 20 dB, it requires more than $10^{4.6}$ data points to begin the identification. Furthermore, STRidge with other d_{tol} values shown in Fig. 6.5 cannot consistently or fail to find the correct equation. Interestingly, when SNR is smaller than 16 dB, none of the shown algorithms can identify the Navier-Stokes equation.

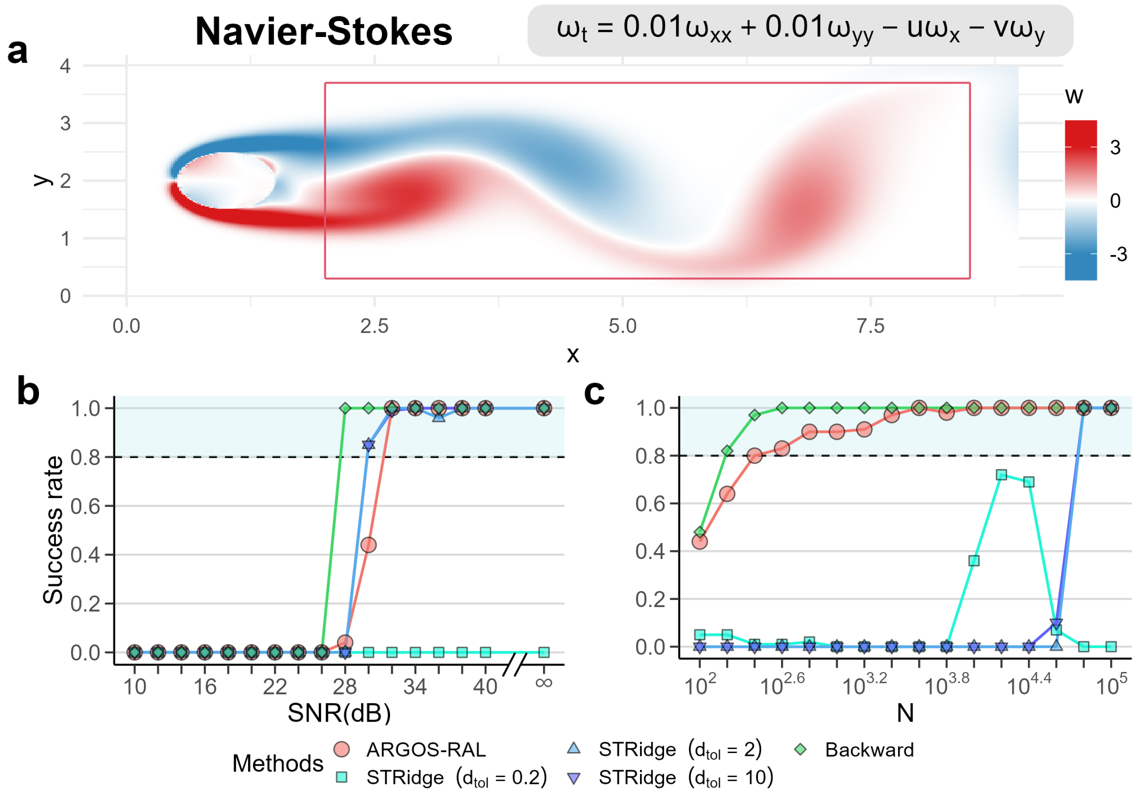


Figure 6.5: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the Navier-Stokes equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a), and 300,000 data points are randomly sampled for the analysis. Results with SNR smaller than 10 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

Reaction-diffusion equation

Reaction-diffusion systems describe pattern-forming systems widely used in many natural phenomena. This dynamic model was initially applied in chemistry and extended to biology, geology, physics, and ecology. Reaction-diffusion systems, such as spots on the skins, zigzags, spiral waves, and rolls, drive many periodic patterns. The following analysis considers the $\lambda - \omega$ systems:

$$u_t = 0.1u_{xx} + 0.1u_{yy} + u - uv^2 - u^3 + v^3 + u^2v \tag{6.7a}$$

$$v_t = 0.1v_{xx} + 0.1v_{yy} + v - uv^2 - u^3 - v^3 - u^2v \tag{6.7b}$$

This reaction-diffusion equation produces spiral waves with periodic boundaries in a two-dimensional space, following the simulation method in [4] to generate the data of the reaction-diffusion equation. The space grid $x, y \in [-10, 10]$ with 512×512, and the time-steps $t \in [0, 10]$ with 201 points. Therefore, the simulated data have a grid with $512 \times 512 \times 201 = 52690944$ points. To speed up the computation time, 15000 (about 0.28%) points are randomly sampled from the whole dataset with 5000 spatial points and 30 time points for further analysis. Figure 6.6(a) illustrates the simulated Reaction-Diffusion equation at t=0.45. The candidate library contains derivatives terms (at most 2nd order and calculated via Algorithm 2), polynomial terms (at most 3rd order) and all interactions. Therefore, the candidate library has 110 candidates.

When analysing the noisy effect, 100 sampled datasets are used to calculate the probability of correct identification (success rate) at each SNR. However, using the automated Savitzky-Golay filter to calculate partial derivatives and build one candidate library with 150,000 points requires about 18 hours with 18 CPU cores under parallel computing, so it is computationally expensive to build 100 candidate libraries. Therefore, the sample sizes are enlarged to 300,000 (5000 spatial points and 60 time points) and then subsampled 150,000 points from the parent sampled dataset.

For the noisy data analysis, Fig. 6.6(b), backward stepwise selection tolerates the largest noise, which can 100% identify the system when SNR $\geq$ 26 dB. ARGOS-RAL starts to identify the system at SNR = 30 dB and can 100% identify the system when SNR $\geq$ 34dB. Furthermore, STRidge with $d_{tol} = 2$ and 10 has the same results. They begin identifying the correct equation at SNR = 30 dB and keep it at 100% after that SNR value. However, the success rate for STRidge with $d_{tol} = 0.2$ waves when SNR $\geq$ 32 dB.

For data size analysis, Fig. 6.6(c), backward stepwise selection outperforms others. It requires only 100 points to reach a 61% success rate and keep at 100% when data have more than $10^{2.6}$ (about 400) points. ARGOS-RAL reaches and keeps a 100% success rate when sampled data sizes are larger than 10,000. STRidge with $d_{tol} = 0.2$ performs the best among other d_{tol} values, which has an 82% success rate

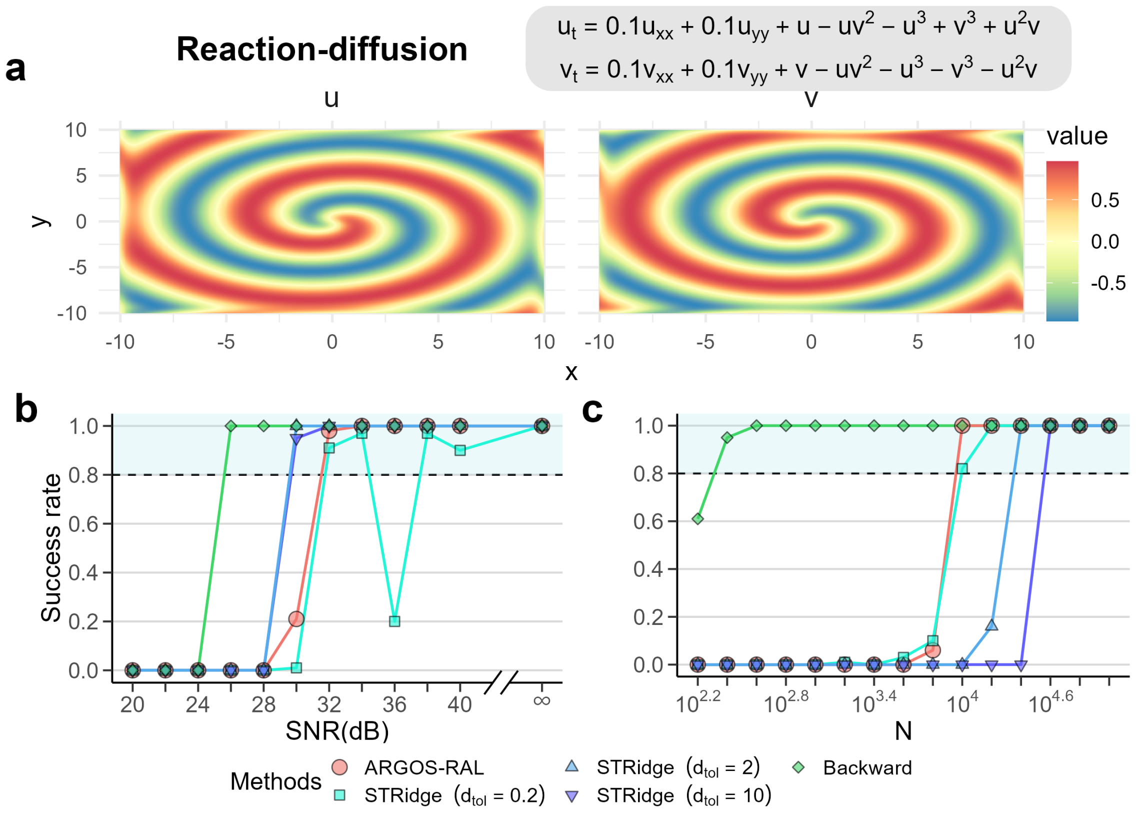


Figure 6.6: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the reaction-diffusion equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a), and 150,000 data points are randomly sampled for the analysis. Results with SNR smaller than 20 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

at N = 10,000 and reaches 100% when more than $10^{4.2}$ data points are sampled.

Transport equation

The transport equation, Eq. (6.8), is a fundamental PDE in science and engineering, describing the evolution of a scalar quantity or a vector field in space and time. The analytic solution, Eq. (6.9), is employed with $c = 3$ to solve the equation and get the data.

$$u_t = cu_x, \quad c > 0 \tag{6.8}$$

$$u(t, x) = \exp(-(x + ct)^2) \tag{6.9}$$

The identification results are shown in Fig. 6.7. All methods have 100% success rates in identifying noiseless data. In noisy data analysis, backward stepwise selection outperforms other methods. Although all STRidge methods have higher noise tolerance than ARGOS-RAL, their success rates are inconsistent, fluctuating with SNR increases. ARGOS-RAL has a robust success rate, but it starts identifying the correct equation when SNR is larger than 56 dB.

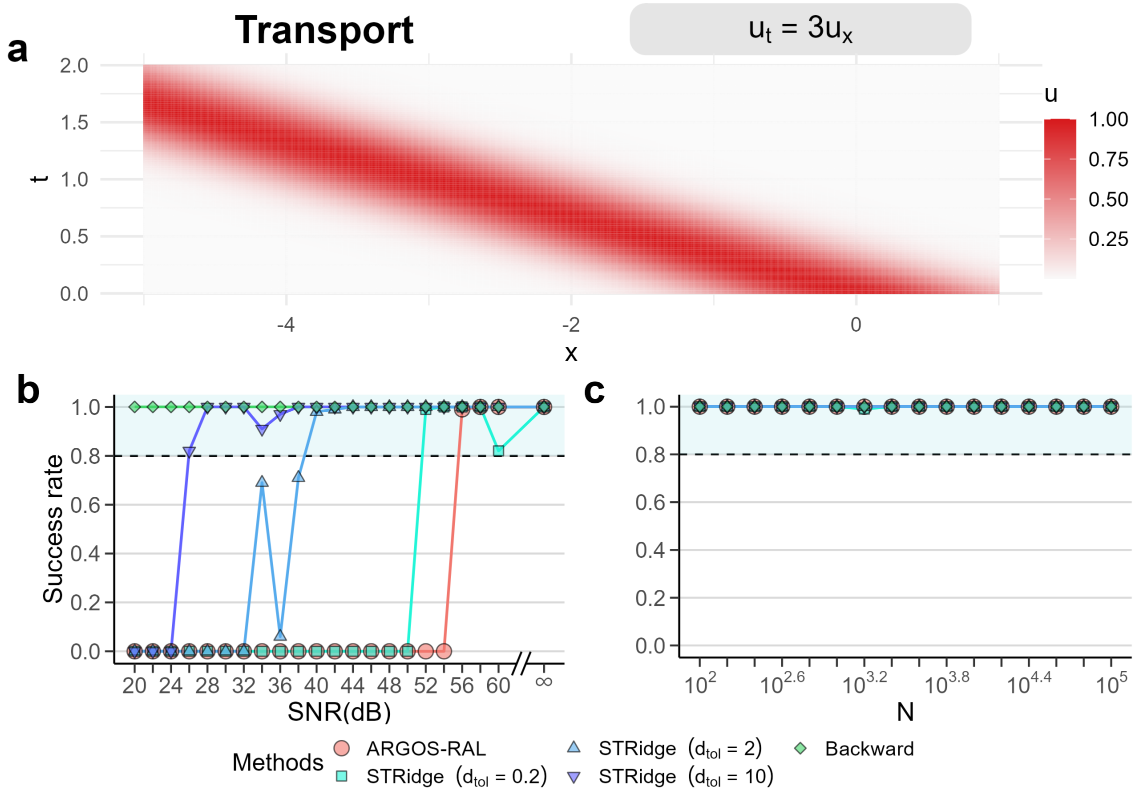


Figure 6.7: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the transport equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 20 dB are removed because the success rates are zero for ARGOs-RAL and STRidge. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

Diffusion equation

The diffusion (heat) equation, Eq. (6.10), holds paramount significance in diverse scientific and engineering domains, such as solid-state physics, materials science, and environmental science, due to its ability to elucidate the fundamental process of heat diffusion. The equation is given by

$$u_t = ku_{xx}, \tag{6.10}$$

where k is a positive number. Identifying this equation enables engineers to gain profound insights into heat conduction, thermal conductivity, and temperature-dependent phenomena in solids and other materials. Here, the analytic solution is used to get the data.

$$u(t, x) = 6 \sin\left(\frac{\pi x}{L}\right) \mathbf{e}^{-k\left(\frac{\pi}{L}\right)^2 t}, \quad k = 10 \tag{6.11}$$

The success rate plots in Fig. 6.8(b) show that all methods can identify noisy data with high SNR. Particularly, backward stepwise selection tolerates more noise than other methods. When SNR is larger than 42 dB, backward stepwise selection starts identifying the correct equation, while STRidge with $d_{tol} = 10$ requires SNR greater than 46 dB to identify the equation. In the sample size test, Fig. 6.8(c), only STRidge with $d_{tol} = 0.2$ performs a decreasing trend as the sample sizes increase, while other methods have a nearly 100% success rate.

Interestingly, ARGOS-RAL archives a 100% success rate in the sample size test of both transport and diffusion equations, while the success rate is almost zero in the SNR tests. However, a small noise corruption, i.e., SNR $>$ 56 dB in the transport equation and SNR $>$ 60 dB in the diffusion equation, makes it difficult for the recurrent adaptive lasso in ARGOS-RAL to penalise extra terms and select the correct ones. Furthermore, the PDE solutions (i.e., the input data) also influence the identification results. Here, the transport and diffusion equations are solved analytically with specific initial and boundary conditions. If these conditions are changed or numerical solvers are used, the solutions of these equations will differ from the ones currently used. As the input datasets change, ARGOS-RAL may

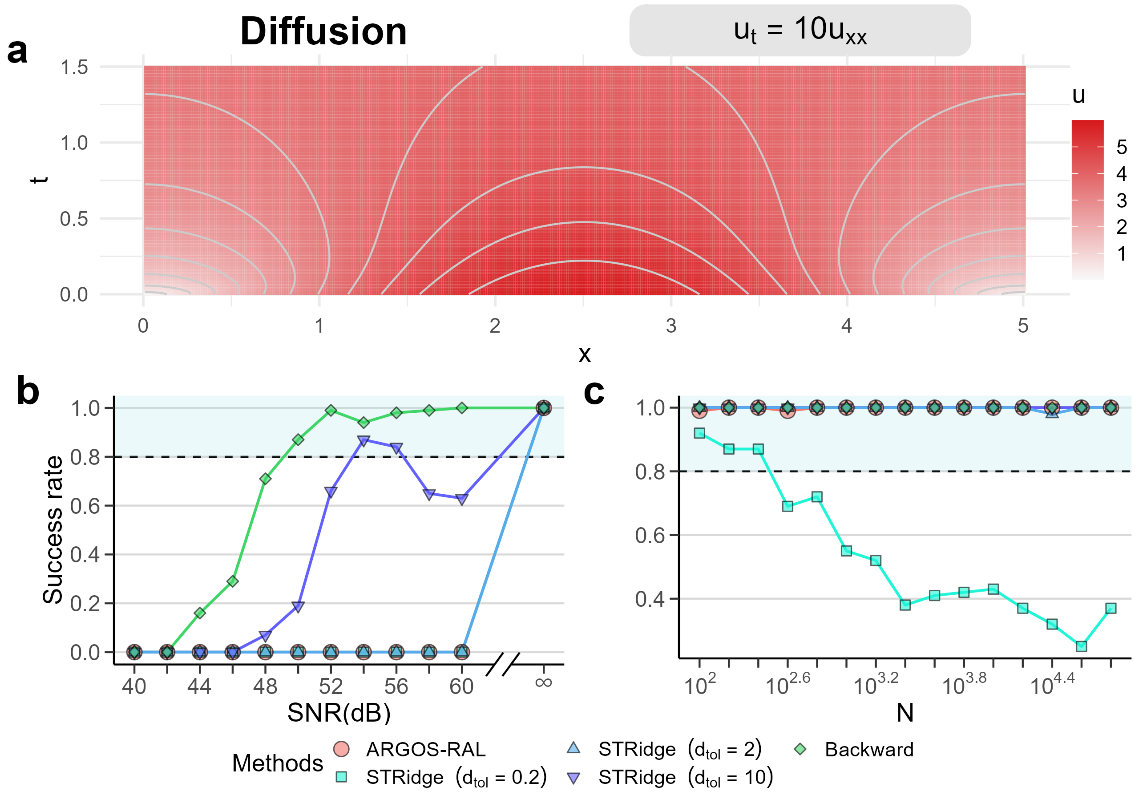


Figure 6.8: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the diffusion equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 40 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

tolerate more noise in the SNR test.

Advection-diffusion equation

The advection-diffusion equation (also known as convection-diffusion) combines both advection and diffusion terms in the system. It describes the transport and dispersion of quantities, such as temperature, the concentration of a substance, or fluid velocity in science and engineering. The equation has the following form:

$$c_t = Dc_{xx} - uc_x. \tag{6.12}$$

This equation is solved by `odeint` in Python with the Fourier spectral method. For the solution, $D = 1$, $u = 1$, $x \in [-10, 10]$ with $\Delta x = 0.1$, $t \in [0, 10]$ with $\Delta t = 0.01$, and the initial condition is a Gaussian function: $\exp\left(-(x+2)^2\right)$.

The identification results are shown in Fig. 6.9 with backward stepwise selection performs the best in both SNR and N tests. ARGOS-RAL has a consistent success rate with the increase of SNR and sample size, outperforming the STRidge. In noisy data, it starts identifying the correct equation from SNR is 14 dB and reach 100% success at SNR = 26 dB. In the sampled data test, it has an 83% success rate when only 100 points are sampled and can 100% identify the correct equation when there are $10^{2.6}$ points.

STRidge with $d_{tol} = 0.2$ has a consistent success rate in the SNR test, while the rates for the other two fluctuate. Although STRidge with $d_{tol} = 2$ is better than ARGOS-RAL at SNR = 24 and 26 dB, the success rate drops to zero at SNR = 34 dB. The success rate of STRidge with $d_{tol} = 10$ reaches to 100% at SNR = 30 dB, but it dramatically drops from SNR = 36 dB. In the sampled data test, all STRidge fluctuate irregularly. Overall, backward stepwise selection and ARGOS-RAL show robust results in identifying the advection-diffusion equation with SNR and sample sizes varying.

Korteweg–De Vries equation

The Korteweg–De Vries (KdV) equation is a nonlinear PDE, describing the travel of waves on shallow water surfaces:

$$u_t = -6uu_x - u_{xxx}. \tag{6.13}$$

In the solution of the KdV equation, the travelling wave is linear when waves are isolated, while they exhibit nonlinearity when interacted. However, because of the dependence of wave velocity on wave amplitude, any solution with multiple amplitudes will exhibit nonlinear behaviour regardless of the interaction.

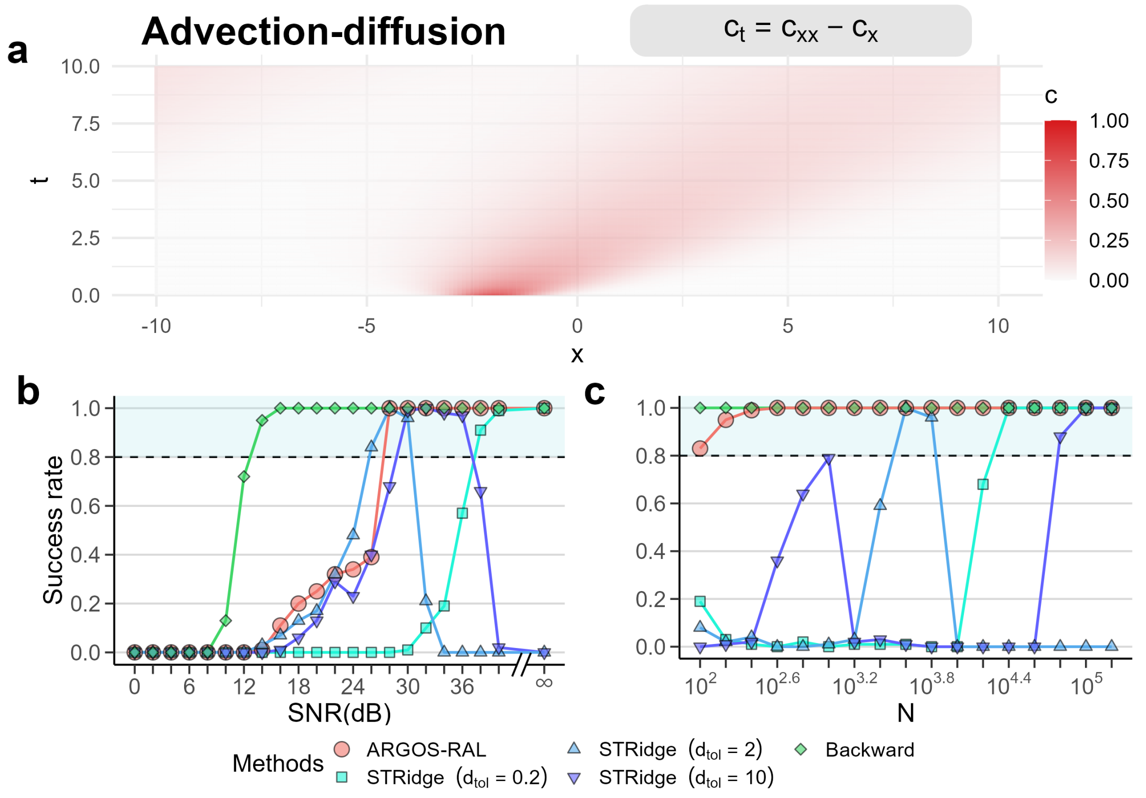


Figure 6.9: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the advection-diffusion equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

The noiseless data is solved through a two-soliton solution [172, pg.858]:

$$\begin{aligned} w(x,t) &= -2\frac{\partial^2}{\partial x^2}\ln\left(1 + B_1 e^{\theta_1} + B_2 e^{\theta_2} + AB_1B_2e^{\theta_1+\theta_2}\right) \\ \theta_1 &= a_1 x - a_1^3 t, \quad \theta_2 = a_2 x - a_2^3 t, \quad A = \left(\frac{a_1 - a_2}{a_1 + a_2}\right)^2, \end{aligned} \tag{6.14}$$

where a_1, a_2, B_1, and B_2 are arbitrary constants, but they are set $a_1 = 0.5$, $a_2 = 1$, $B_1 = 1$, $B_2 = 5$. The timesteps is set to 201 ($n = 201$) $t \in [0, 20]$, and spatial points is set 512 ($m = 512$), $x \in [-30, -30]$. The Dirichlet boundary condition, $u(x,t) = 0$ at $x = \pm\infty$, is applied.

The identification results in Fig. 6.10(b) show that ARGOS-RAL has a more consistent success rate than STRidge. ARGOS-RAL can identify the equation when

SNR $\geq$ 50 dB for noisy data or $N \geq 10^{3.8}$ for noiseless data. Although STRidge with $d_{tol} = 0.2$ performs better than ARGOS-RAL at SNR = 48 dB, the success rate drops at SNR = 54 and 56 dB. STRidge with $d_{tol} = 2$ has a 100% success rate only at SNR = 50, 52, 54 dB, whereas the success rates for $d_{tol} = 10$ are zero.

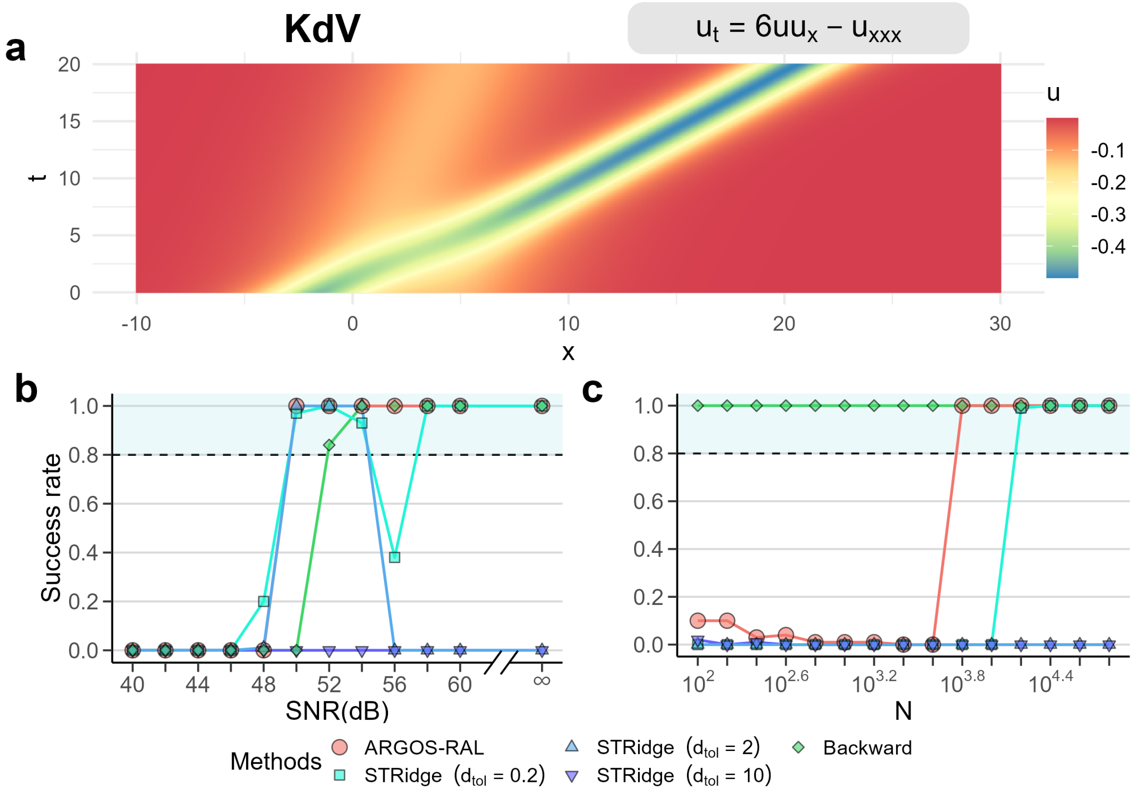


Figure 6.10: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the KdV equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 40 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

Figure 6.10(c) illustrates that backward stepwise selection, ARGOS-RAL, and STRidge with $d_{tol} = 0.2$ can identify the noiseless data of the KdV equation. Backward stepwise selection finds the correct equation when the dataset contains only 100 data points. Although ARGOS-RAL has a 10% success rate when N = 100, it drops and suddenly jumps to 100% at N = $10^{3.8}$. Furthermore, only STRidge with $d_{tol} = 0.2$ has a 100% success rate when data sizes are larger than $10^{4.2}$.

Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky (KS) equation is used to describe the complex spatial-temporal dynamics in extended systems driven far from equilibrium by intrinsic instabilities [173]. It has been applied to model phenomenons, including instabilities of dissipative trapped ion modes in plasmas, instabilities in laminar flame fronts, phase dynamics in reaction-diffusion systems, and fluctuations in fluid films on inclines [174].

The KS equation is a nonlinear PDE with nonlinear wave-breaking dynamics $u_t + uu_x = 0$. The second-order partial differentiate term u_{xx} describes the long-wavelength instabilities, whereas the fourth-order provides the stabilising regularisation. Equation (6.15) shows the KS equation. Within the generated data, the space $x \in [0, 100]$ with $m = 1024$, and the time-steps $t \in [0, 100]$ with $n = 251$.

$$u_t = -uu_x - u_{xx} - u_{xxxx} \tag{6.15}$$

Figure 6.11 shows that backward stepwise selection has the most consistent and accurate results. In noisy data tests, Fig. 6.11(b), only backward stepwise selection has 100% success rates when SNR $\geq$ 34 dB. Although STRidge with $d_{tol} = 10$ has a 45% success rate at SNR = 34 dB, others all fail to identify the KS equation. In identifying noiseless data, Fig. 6.11(c), backward stepwise selection has a 99% success rate when N = 100 and keeps 100% after that value. The performances of other methods fluctuate without consistent success rates.

Figure 6.12 shows that the adaptive lasso can find the correct model with specific λ. Under the given case, i.e., a random sampled noisy dataset with SNR = 36 dB, the correct λ value should be located between 26.96 (about $e^{3.29}$) and 29.59 (about $e^{3.39}$). However, the current regularisation parameter's finding methods (10-folder cross-validation and the Pareto curve) cannot give the appropriate λ value. The λ values from both methods are lower than the correct λ range. The best λ value from 10-folder cross-validation is 0.019 (about $e^{-3.96}$), while the best λ for the Pareto curve is 0.453 (about $e^{-0.79}$). Therefore, using the adaptive lasso with 10-folder cross-validation and the Pareto curve cannot identify the correct KS equation.

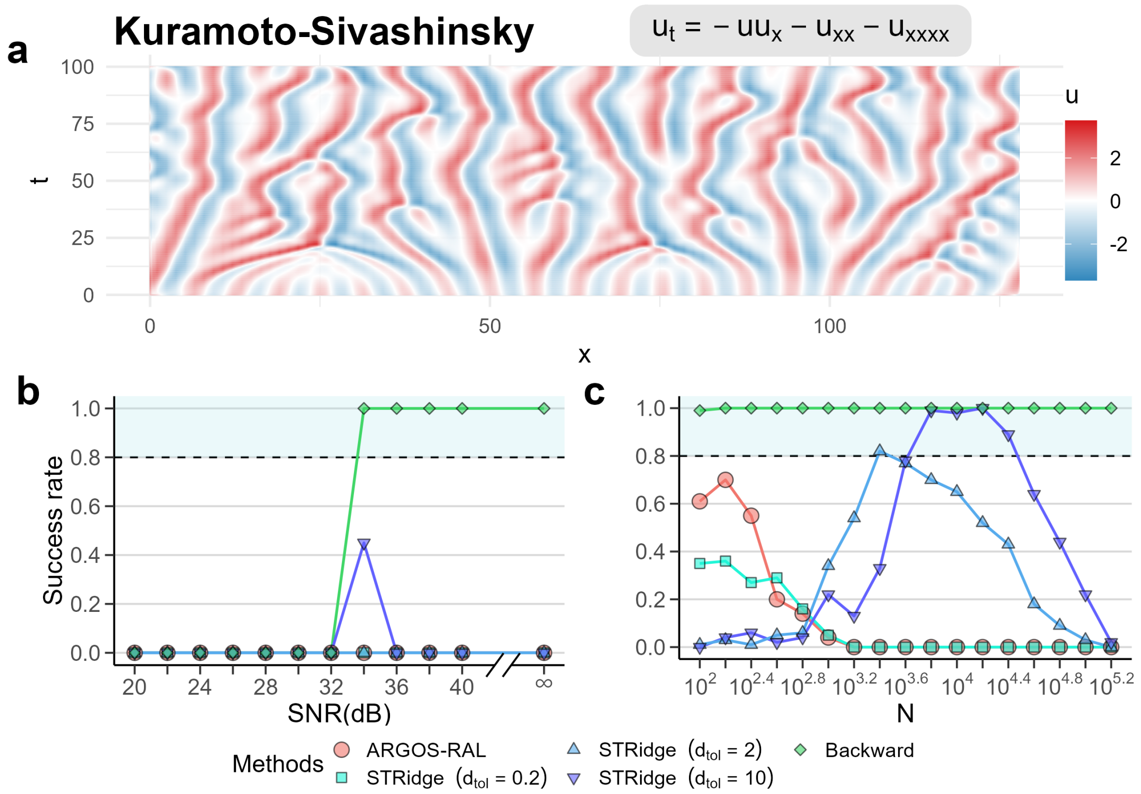


Figure 6.11: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the KS equation. (a) shows the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 20 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

Quantum Harmonic Oscillator

The quantum harmonic oscillator (QHO) describes the parabolic potential of a harmonic oscillator in quantum mechanics. This equation simulates the temporal development of the wave function related to a particle in the parabolic potential. It provides the distribution function of the particle's position at any moment by taking the squared magnitude. The energy levels of a quantum harmonic oscillator are quantised, meaning that they can only take on specific, discrete values. Additionally, even if a statistical distribution is formed from several experiments, it will be devoid of details about the intricate phase of the wave function. The equation used

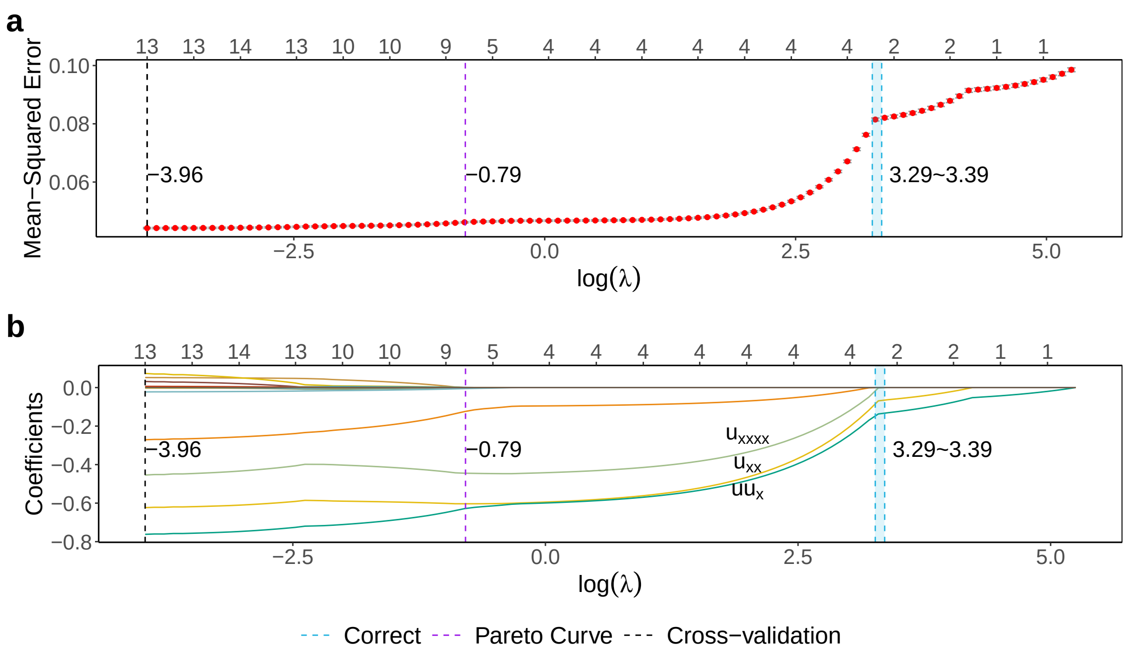


Figure 6.12: The coefficients with different regularisation parameters λ of using the adaptive lasso to identify the KS equation. The used case is a noisy dataset with SNR = 36 dB. By varying λ, the mean squared errors from 10-folder cross-validation are shown in (a), and the estimated coefficients are shown in (b). The top axis shows the number of non-zero terms.

in this work is:

$$u_t = \frac{1}{2}iu_{xx} - iuV = \frac{1}{2}iu_{xx} - \frac{x^2}{2}iu. \tag{6.16}$$

The operator splitting method with Fourier transform is used here to solve the QHO equation numerically. The time domain is $t \in [0, 10]$ with $\Delta t = 0.025$, and the space domain is $x \in [-7.5, 7.5]$ with $\Delta x = \frac{15}{512}$ with a Gaussian initial condition.

Figure 6.13 shows that backward stepwise selection is the best method for finding the QHO equation, while ARGOS-RAL succeeds in dealing with noiseless data. In noisy data, Fig. 6.13(b), only backward stepwise selection consistently and 100% identifies the correct equation when SNR is greater than 16 dB. However, the success rates for STRidge with all d_{tol} values are waving from SNR larger than 18 dB. For noiseless data, Fig. 6.13(c), both backward stepwise selection and ARGOS-RAL have 100% success rates. STRidge with $d_{tol} = 0.2$ shows a decreasing trend, $d_{tol} = 10$ illustrates a fluctuation performance, and $d_{tol} = 2$ has zero success rate.

Figure 6.14 shows that the correct model is contained in the searching path of

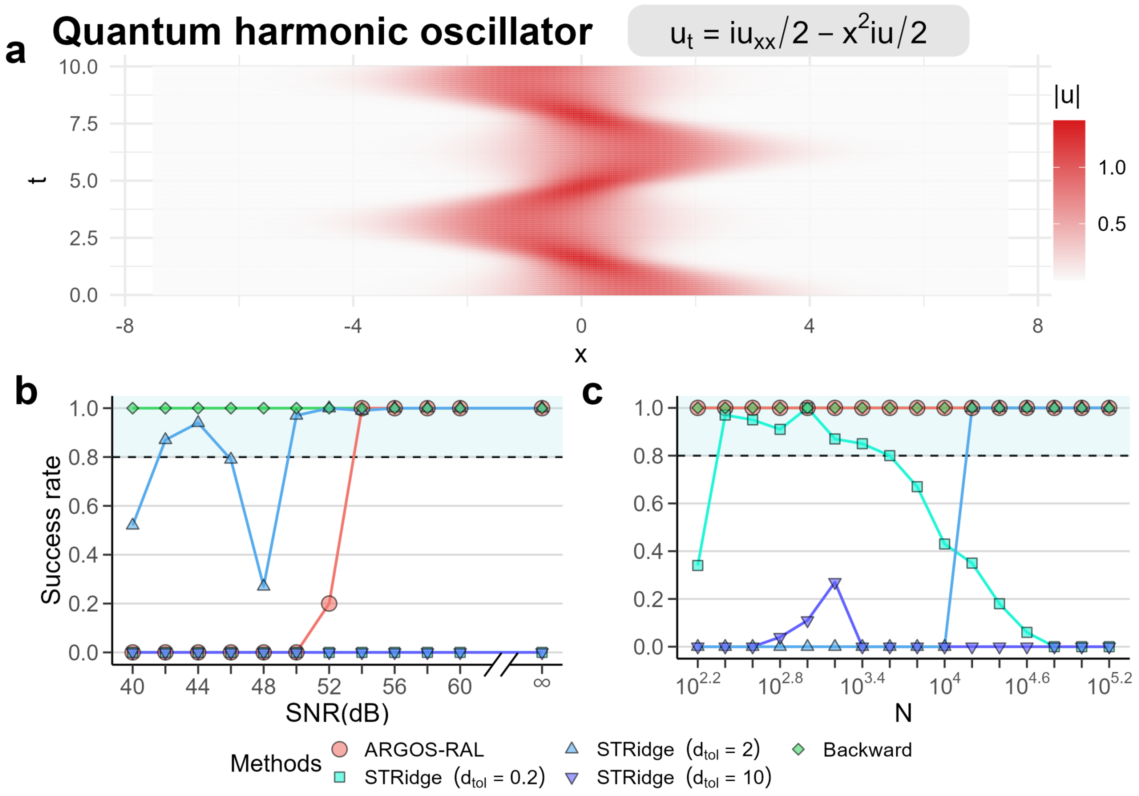


Figure 6.13: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the QHO equation. (a) shows the absolute value of the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 40 dB are removed because the success rates are zero for ARGOS-RAL and STRidge with $d_{tol} = 0.2$ and 10. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

the adaptive lasso when SNR = 28 dB. The correct λ value should be taken between 2408.79 (about $e^{7.79}$) and 16993.55 (about $e^{9.74}$). However, 10-folder cross-validation and the Pareto curve estimate smaller λ values than the correct range. The best λ value from 10-folder cross-validation is 5.19 (about $e^{1.65}$), while it is 589.85 (about $e^{6.38}$) for the Pareto curve. Therefore, using the adaptive lasso with 10-folder cross-validation and the Pareto curve cannot identify the correct QHO equation.

Nonlinear Schrodinger equation

The nonlinear Schrodinger (NLS) equation appears to be one of the universal equations, describing quasi-monochromatic wave evolution with slowly varying packets.

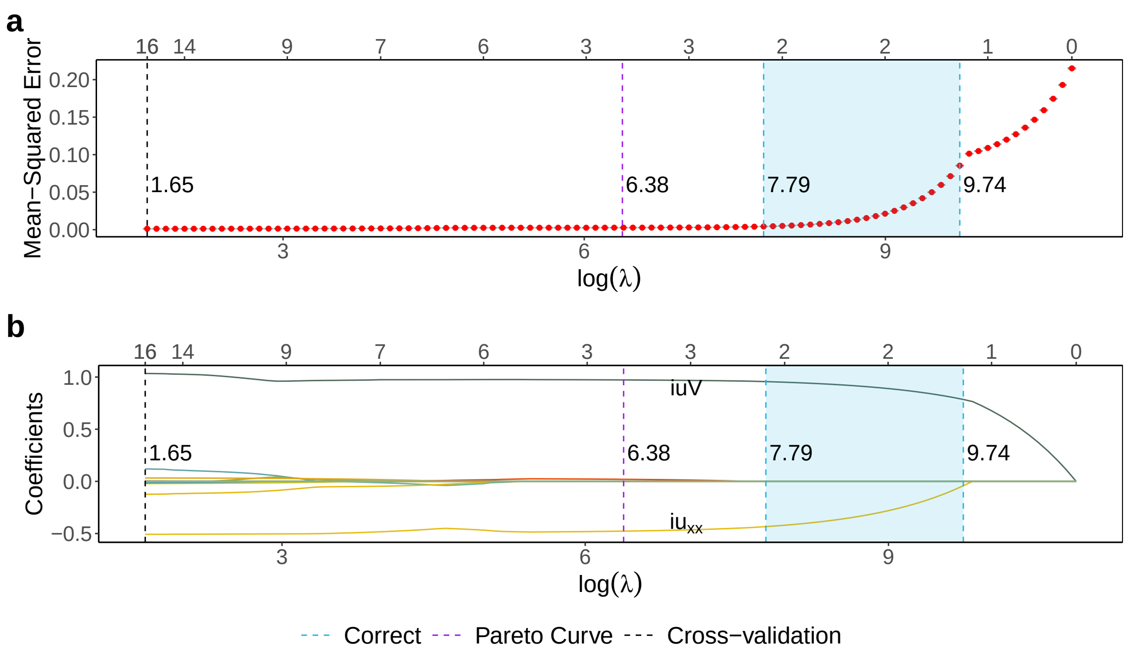


Figure 6.14: The coefficients with different regularisation parameters λ of using the adaptive lasso to identify the QHO equation. The used case is a noisy dataset with SNR $=$ 28 dB. By varying λ, the mean squared errors from 10-folder cross-validation are shown in (a), and the estimated coefficients are shown in (b). The top axis shows the number of non-zero terms.

It is widely applied in many fields, such as light propagation in nonlinear optical fibres, plasma waves, and Bose-Einstein condensates. The NLS equation is:

$$u_t = \frac{1}{2} i u_{xx} + i|u|^2 u. \tag{6.17}$$

In the simulation, the spatial points are set in this way: $x \in [-5, 5], m = 512$; time-steps $t \in [0, \pi], n = 501$. The initial condition is the Gaussian function. Since the function refers to complex numbers, terms depending on the magnitude of the solution, such as $|u|$, are added to the candidate library, and Eq. (6.2) is used to transform the regression in complex numbers to real numbers.

Figure 6.15 shows that backward stepwise selection performs the best, having the highest and the most consistent success rate. For noisy data, Fig. 6.15(b), backward stepwise selection begins identifying the correct equations with a 96% success rate at SNR $=$ 24 dB and keeps at 100% when SNR $\geq$ 28 dB. STRidge with $d_{tol} = 500$ can 100% identify the NLS equation when SNR $\geq$ 32 dB. For noiseless data, Fig. 6.15(c),

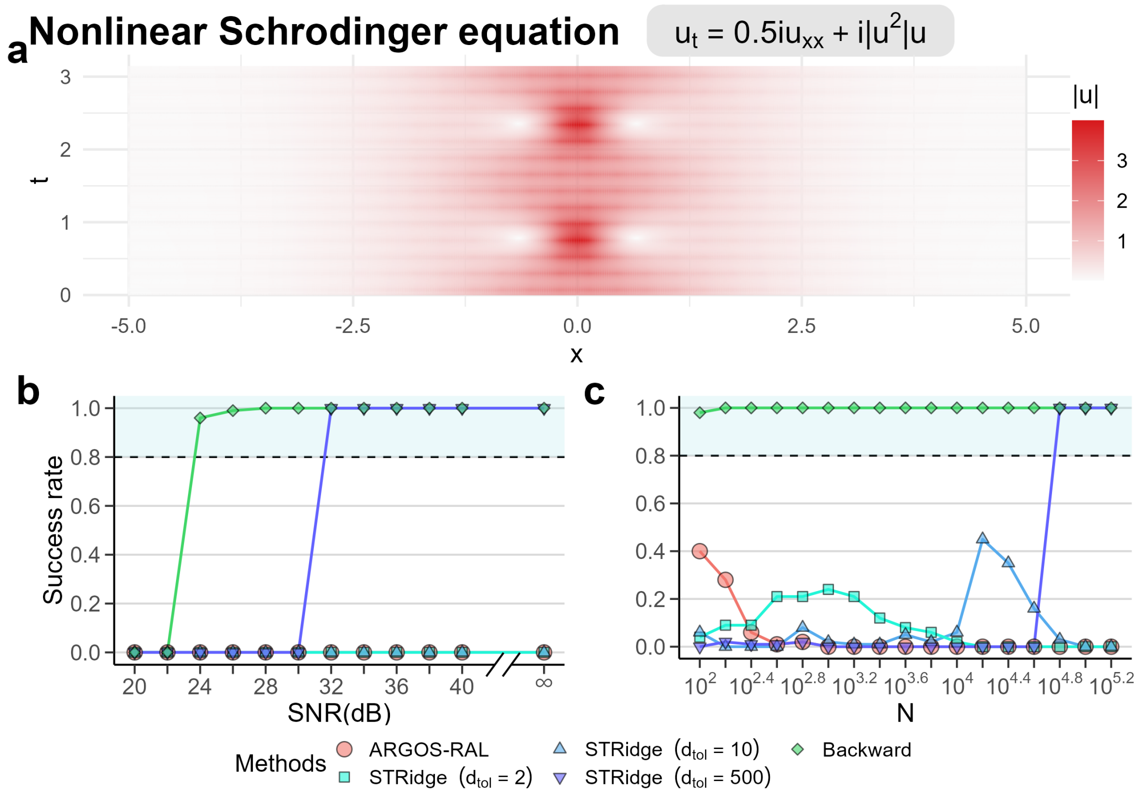


Figure 6.15: Success rates of ARGOS-RAL, STRidge, and the backward stepwise selection in identifying the NLS equation. (a) shows the absolute value of the PDE solution with noiseless data. (b) and (c) illustrate the success rates for different SNRs and sample sizes, respectively. For each success rate of SNR in (b), 100 random noise sets are added to the PDE solution (a). Results with SNR smaller than 20 dB are removed because the success rates are all zero. For each success rate of N in (b), 100 noiseless datasets are randomly sampled from the PDE solution (a). The hyperparameter d_{tol} in STRidge is varied to be 0.2, 2, and 10.

backward stepwise selection has a 98% success rate when 100 data points are used, and the rate keeps at 100% as data sizes increase. STRidge with $d_{tol} = 500$ can consistently identify the NLS equation when data sizes are larger than $10^{4.8}$. For ARGOS-RAL and STRidge with other d_{tol} values, the performance is not robust, and success rates fluctuate and are always lower than 50%.

Figure 6.16 shows that the correct model can be found by the adaptive lasso when identifying the noisy dataset with SNR = 26 dB. The correct λ value in this case should be from 1826.726 (about $e^{7.51}$) and 11742.328 (about $e^{9.37}$). However, the current regularisation parameter's finding methods, 10-folder cross-validation and the Pareto curve, estimate values lower than the correct λ range, leading to non-parsimonious models. The best λ value from 10-folder cross-validation is 17.437

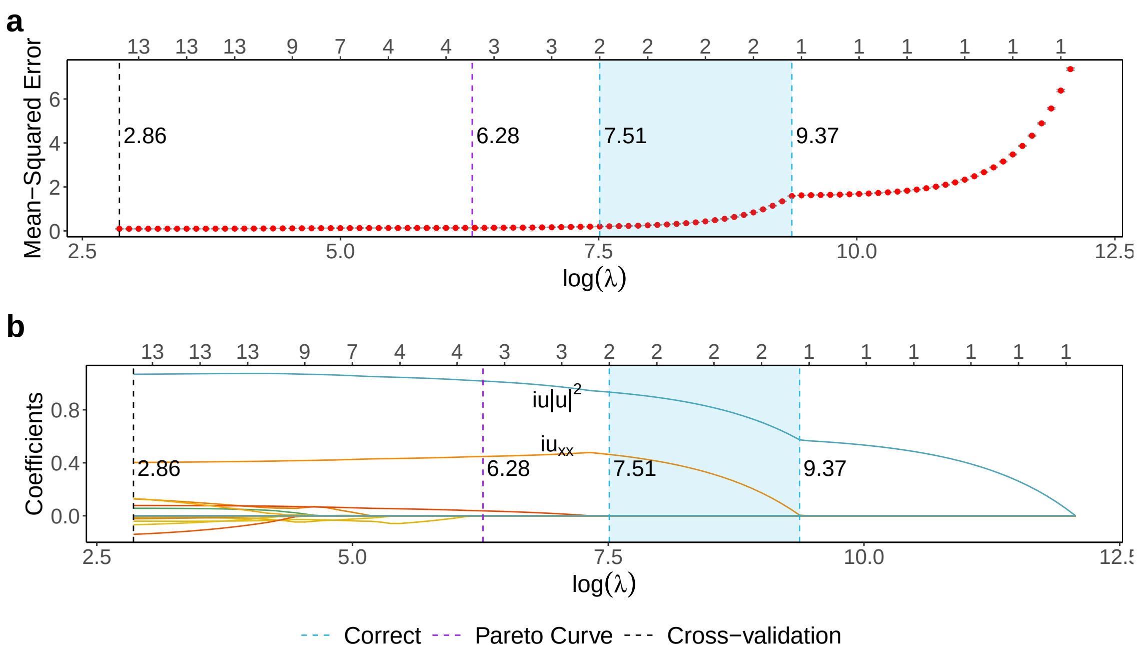


Figure 6.16: The coefficients with different regularisation parameters λ of using the adaptive lasso to identify the NLS equation. The used case is a noisy dataset with SNR = 26 dB. By varying λ, the mean squared errors from 10-folder cross-validation are shown in (a), and the estimated coefficients are shown in (b). The top axis shows the number of non-zero terms.

(about $e^{2.86}$), while the best value for the Pareto curve is 531.185 (about $e^{6.28}$). Therefore, using the adaptive lasso with 10-folder cross-validation and the Pareto curve cannot identify the correct NLS equation.

6.2.3 Computational Costs

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

Figure 6.17 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,

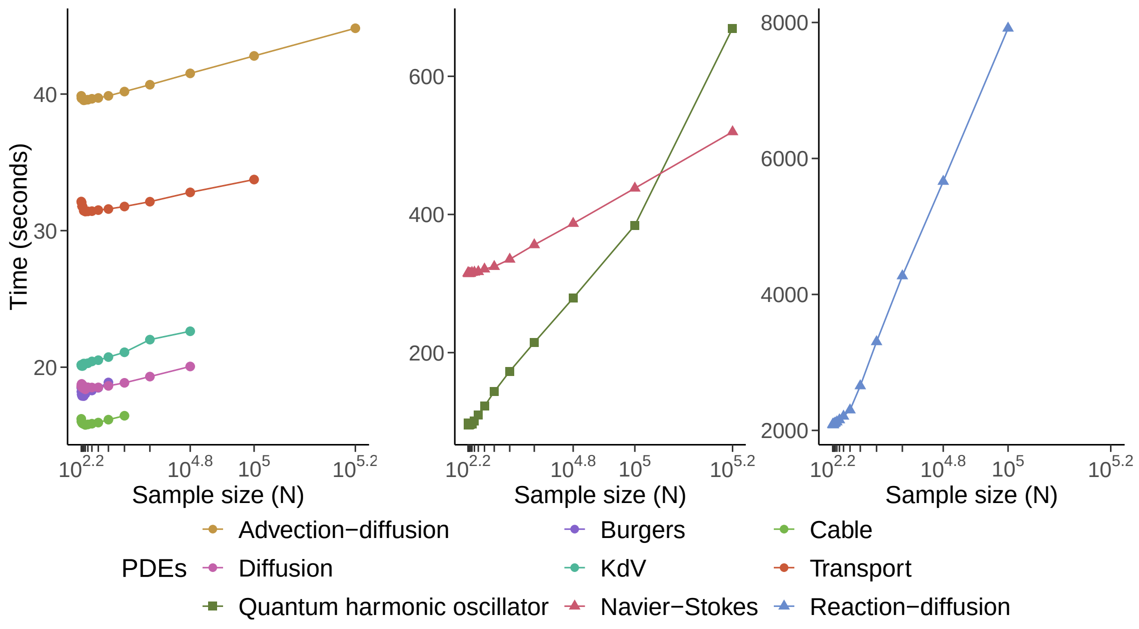


Figure 6.17: Computational costs of ARGOS-RAL as sample size (N) increases for noiseless PDE systems. We run all benchmarks using parallel processing and convert them to total computing time on a single core. Circles and triangles are used for PDEs with one and two spatial dimensions, and squares are for the complex-valued quantum harmonic oscillator.

except for the quantum harmonic oscillator, a PDE containing complex-valued terms whose candidate library is four times larger than those of real-coefficient PDEs, see Eq. (6.2).

6.2.4 Robustness Analysis using White Gaussian Noise

To better understand the limits of identification algorithms, an extreme test on a single spatial dimension was designed. 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 ultimate test stage for an algorithm: identifying dynamical systems without signal, where success rates are expected 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.

One hundred white Gaussian noise datasets are generated, 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, three Gaussian distributions with variances spanning three orders of magnitude are used: $\mathcal{N}(0, 0.1^2)$, $\mathcal{N}(0, 1)$, and $\mathcal{N}(0, 10^2)$. The aim is to determine whether ARGOS-RAL and STRidge can identify canonical PDEs under these noise conditions. Table 6.1 shows the percentages of different identified models. Based on the PDEs tested by Rudy *et al.* [4] and this study, parsimonious models are defined as those having three or fewer nonzero coefficients, suggesting they may correspond to specific physical phenomena. In particular, three classic differential equations are highlighted: the ODE $u_t = c_1u^d$, the transport equation $u_t = c_2u_x$, and the heat equation $u_t = c_3u_{xx}$. In contrast, models with more than three nonzero coefficients are classified as non-parsimonious, indicating that their coefficient vectors have a dense composition.

Table 6.1 and Fig. 6.18 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 Fig. 6.18. This is the desired behaviour when the input signal is pure white noise, as it is important to ensure that the algorithms do not identify parsimonious models in such cases. The difference in behaviour between the two methods is most apparent when the noise level is low to moderate ($\sigma_Z \leq 1$). 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 Fig. 6.18. 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.

Table 6.1: Identified models from random noise by ARGOS-RAL and STRidge. The maximums of different monomial and derivative orders in the candidate library are from three to five. The parsimonious model is defined as the number of nonzero coefficients that are less than or equal to three. The F-test is applied to each identified model with the number of significant models (p-value < 0.05) shown in the parentheses. Numbers without parentheses represent the identified models and do not reject the null hypothesis of the F-test. The constant values c_1, c_2, c_3 are arbitrary numbers. The monomial order d in the ODE cell is a positive integer, of which the maximum is the largest monomial order of the candidate library.

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

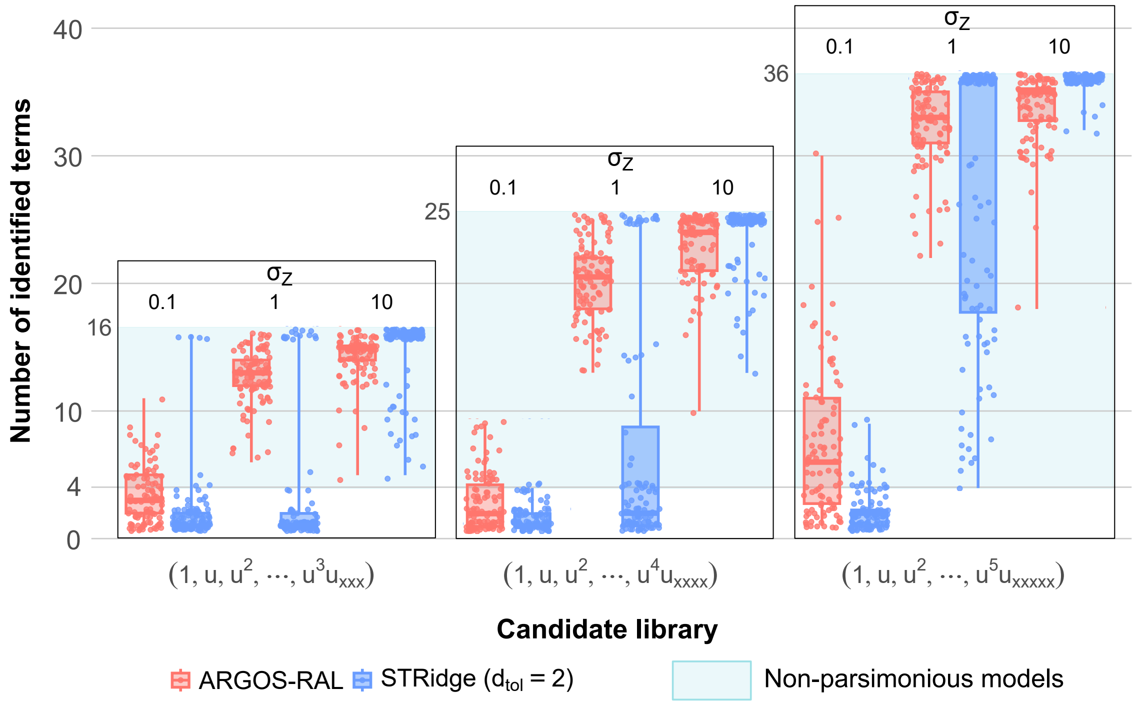


Figure 6.18: Number of nonzero terms identified from 100 random noise datasets using different candidate function libraries. For each case, the number of nonzero coefficients in the sparse regression is counted. The distribution of these counts is displayed using dots for each of the 100 trials and summarised 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.

6.3 Discussion

ARGOS-RAL is proposed 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 modelling 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 previous 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, 4]. 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. ARGOS-RAL focuses on first-order time derivatives [4, 175], 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 [18]. Finally, while ARGOS-RAL provides a more computationally efficient approach than ARGOS [1] 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 modelling process, ARGOS-RAL has the potential to accelerate discovery and insight in fields ranging from physics and engineering to biology and climate science.

6.4 Other Shrinkage Regression Methods in Identifying PDEs

The previous parts of this chapter, Section 6.2.2, have shown the success of using the adaptive lasso in identifying PDEs from data. This part will illustrate the failure of using lasso, elastic-net, and adaptive elastic-net (described in Section 2.4) in identifying PDE. To simplify the illustration, the noiseless data from the Burgers equation is used. The Burgers equation Eq. (6.3) has two active terms uu_x and u_{xx}, which should be found by the following methods: the lasso, the elastic-net, and the adaptive elastic-net. The 10-folder cross-validation will be applied to find the optimal regularisation parameter λ.

6.4.1 The Lasso

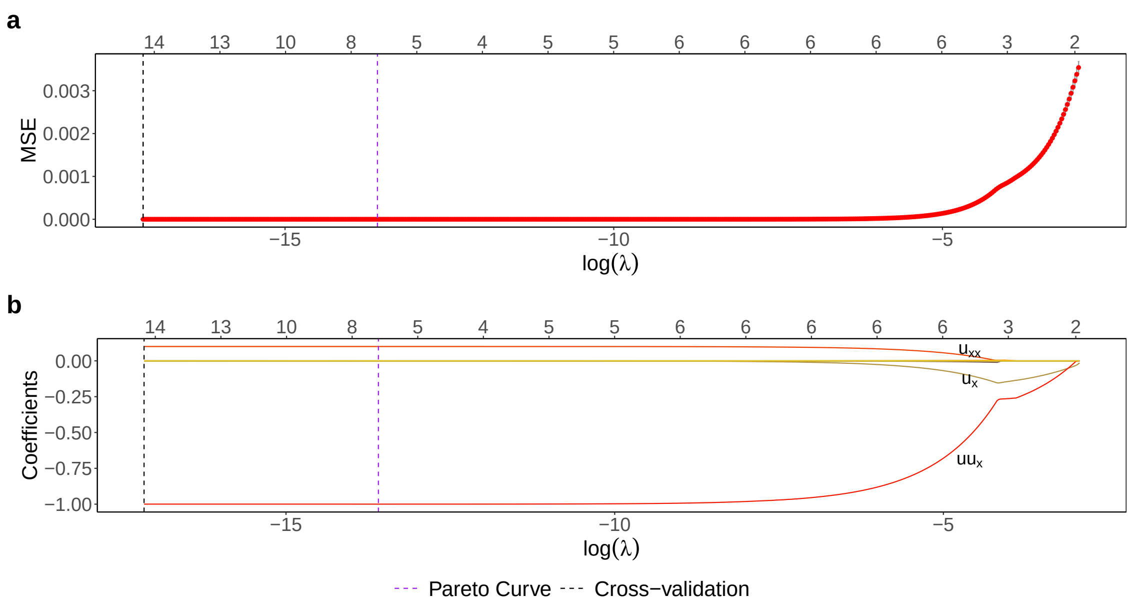


Figure 6.19: The coefficients with different regularisation parameters λ of using the lasso to identify the Burgers equation. By varying λ, the mean squared errors (MSE) from 10-folder cross-validation are shown in (a), and the estimated coefficients are shown in (b). The top axis shows the number of non-zero terms of the lasso estimation.

Figure 6.19 shows that the correct model cannot be found by the lasso. When λ is smaller than e^{-10}, the lasso can find the correct active terms uu_x and u_{xx} with the estimated coefficients to be -1 and 0.1, respectively. Meanwhile, it also finds redundant terms, such as u_x, with the estimated coefficients close to, but not equal to, zero. Although when λ ranges between 0.0209 (about $e^{-3.87}$) and 0.0505 (about $e^{-2.99}$), the lasso finds models containing only two terms: u_x and uu_x, which are incorrect (should be u_{xx} and uu_x). Therefore, if users do not know the approximate values of the coefficients, it is difficult to cut off the redundant terms and identify the correct model.

6.4.2 The elastic-net

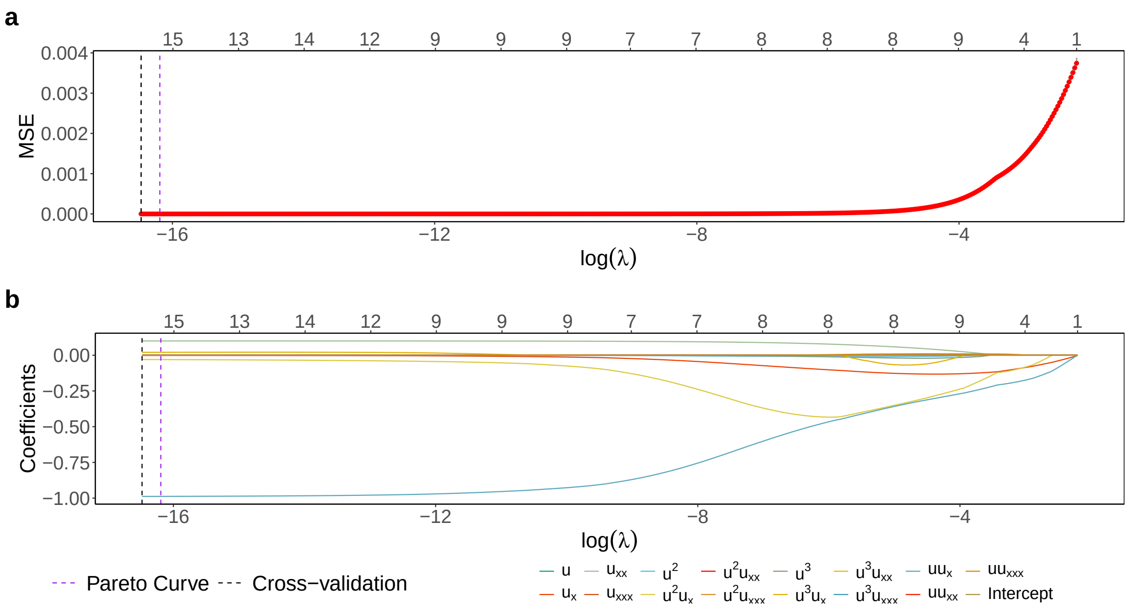


Figure 6.20: The coefficients with different regularisation parameters λ of using the elastic-net with $\alpha = 0.5$ to identify the Burgers equation. By varying λ, the mean squared errors (MSE) from 10-folder cross-validation are shown in (a), and the estimated coefficients are shown in (b). The top axis shows the number of non-zero terms of the lasso estimation.

Figure 6.20 shows that the elastic-net cannot identify the correct equation. When λ is smaller than e^{-12}, it estimates coefficients to about -1 for uu_x and 0.1 for u_{xx}. However, the redundant terms are simultaneously found. As λ increases, most coefficients are penalised close to, but not equal to, zero. However, the estimated coefficients of u^2u_x, u_x and u^3u_x fluctuate, and their values are -0.35, -0.13 and -0.066 respectively, when $\lambda \approx e^{-5}$. Furthermore, when $\lambda \approx 0.033$ (about $e^{-3.41}$), the correct term u_{xx} is penalised to zero, while four extra terms, u_x, u^2u_x, uu_{xxx}

and u^2u_{xxx}, are still active in the equation. Therefore, even though users know the approximate ranges of the correct terms, it is still difficult to apply a threshold to manually attribute redundant terms to zero.

6.4.3 The adaptive elastic-net

Figure 6.21 shows that the adaptive elastic-net with the elastic-net mixing parameter $\alpha = 0.5$ finds the correct equation by using the Pareto curve. However, α is an unknown parameter in real-world analysis and is tuned by cross-validation, which always finds the optimal model with redundant terms.

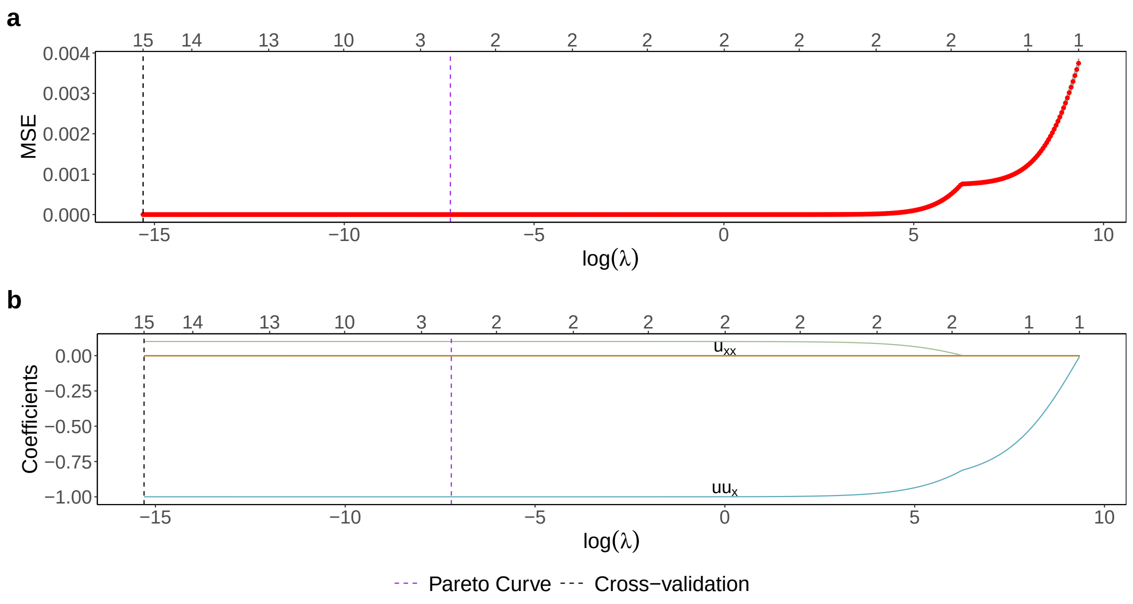


Figure 6.21: The coefficients with different regularisation parameters λ of using the adaptive elastic-net to identify the Burgers equation. By varying λ, the mean squared errors (MSE) from 10-folder cross-validation are shown in (a), and the estimated coefficients are shown in (b). The top axis shows the number of non-zero terms of the lasso estimation.

6.5 Stepwise Selection in Identifying PDEs

This section explains why backward stepwise selection with knowing the number of active terms is used to identify PDEs. The following parts will show the results of identifying Burgers equation from the standard forward, backward and hybrid stepwise selection. As the success rates of forward and hybrid stepwise selection are zero, only the results of the noiseless dataset will be shown. Here, AIC, BIC and adjusted R^2 (see Section 2.2.2) are used to select the best model because these three are the most popular criteria in statistics [33, pg.232].

Only backward stepwise selection (Fig. 6.22) contains the correct model in the search path, while forward stepwise selection, Fig. 6.23 (a) and hybrid stepwise selection, Fig. 6.23 (b) cannot find the correct model from their search path. How-

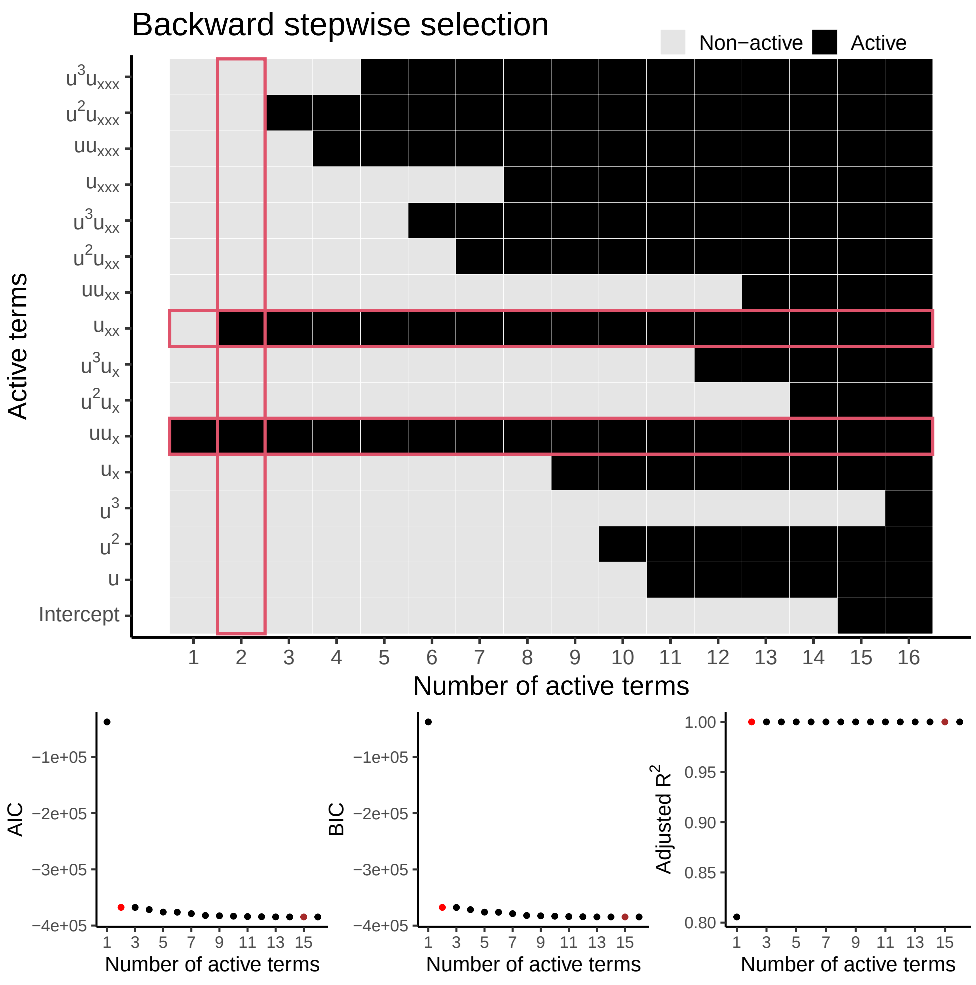


Figure 6.22: Potential models (the top graph) and their goodness-of-fit criteria (bottom three graphs) from the backward stepwise selection. In the top graph, red rectangles represent the correct model, which should only contain two active terms uu_x and u_{xx}. The correct model is shown in red point, whereas the best models for AIC, BIC and adjusted R^2 are shown in brown points.

ever, the criteria graphs, including AIC, BIC, and adjusted R^2, illustrate that the best models of these three stepwise methods always contain redundant terms. The best model of forward stepwise selection is the one with all terms, whereas the best models of backward and hybrid stepwise selection contain 15 terms. Due to this problem, even though the correct model is a candidate model in the backward stepwise selection, the correct one cannot be selected. To enable the stepwise selection method in identifying PDEs, users need to determine how many active terms are in the model and use backward stepwise selection to find the correct one.

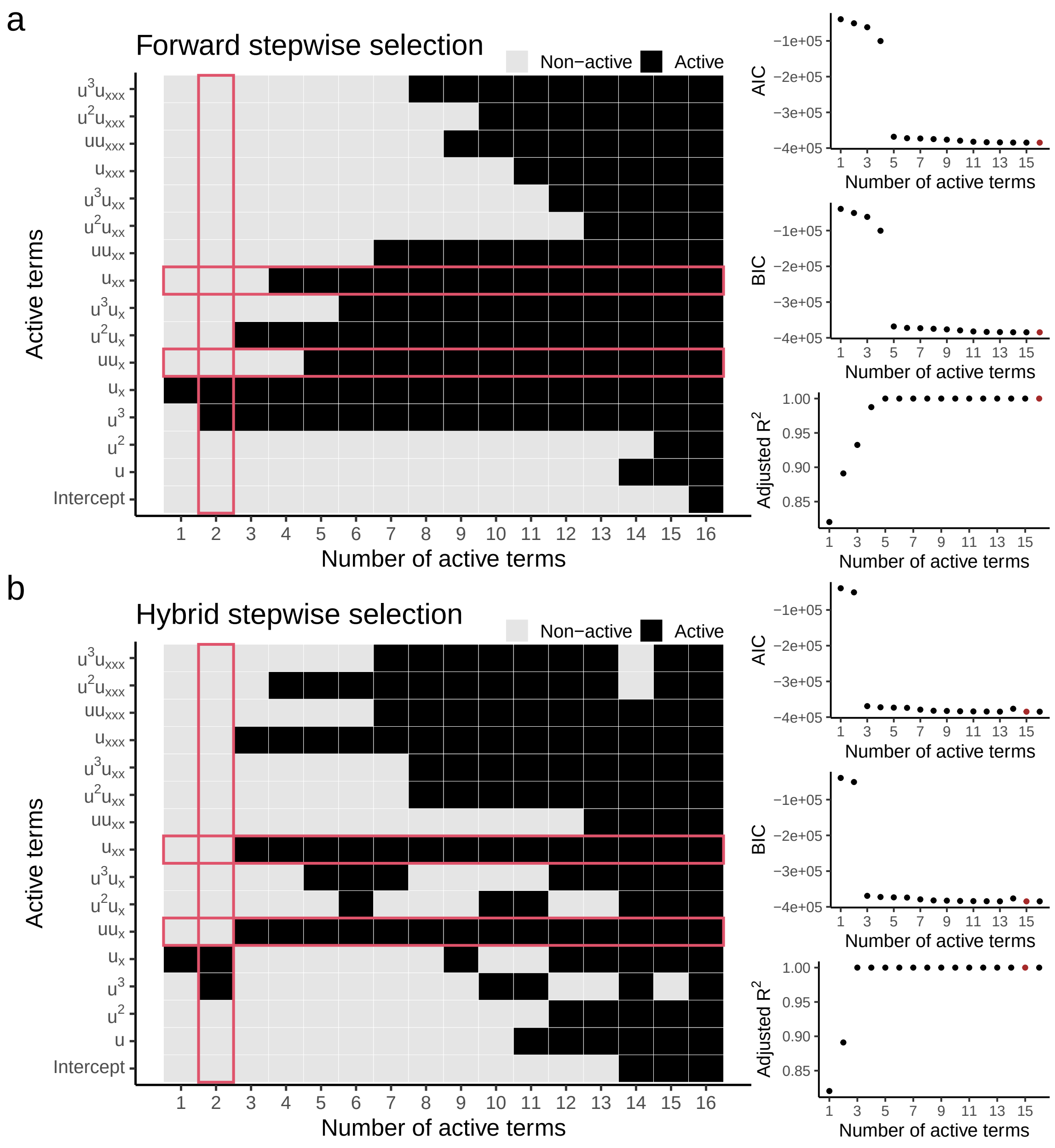


Figure 6.23: Potential models and their goodness-of-fit criteria of the (a) forward and (b) hybrid stepwise selection. On the left graphs of panels (a) and (b), red rectangles represent the correct model, which should only contain two active terms uu_x and u_{xx}. The best models for AIC, BIC and adjusted R^2 are shown in brown points on the right graphs.

6.6 Summary

This chapter introduced ARGOS-RAL as an automatic framework for discovering PDEs from data with limited prior knowledge.

- The method automates the calculation of partial derivatives using an automatic Savitzky-Golay filter with Gaussian blur (Chapter 5), enabling the construction of the candidate library from data without manually tuning parameters. ARGOS-RAL then employs the recurrent adaptive lasso, a sparse regression technique, to identify the active terms in the PDE from the candidate library.

- ARGOS-RAL outperforms STRidge, which is used in SINDy, for most test cases to identify canonical PDEs under varying noise levels and sample sizes. Importantly, when given pure noise as input, ARGOS-RAL correctly identifies either a dense model, avoiding the false discovery of parsimonious physical PDEs, while STRidge has the same performance when noise variation is large.

- The evaluation demonstrates ARGOS-RAL's robustness to noisy and non-uniformly sampled data in identifying canonical PDEs. This work showcases the potential of combining statistics, machine learning, and dynamical systems to accelerate scientific discovery by automatically learning governing equations from data.

CHAPTER 7

ARGOS Framework in Finding COVID-19 Transmission Dynamics

In Chapter 6, ARGOS-RAL has been proposed to automatically extract canonical PDEs from data. Although PDEs can model complex spatiotemporal dynamical systems, ODEs are more frequently employed in practical applications, including power systems [21], infectious disease dynamics [176], ecosystem evolution [177], and biopharmacy [178], due to their relative simplicity and effectiveness. This preference underscores the importance of selecting appropriate mathematical tools to tackle real-world challenges. In light of this, this chapter shifts focus to identifying ODEs from a real-world dataset: finding the governing equations for the evolution of the coronavirus disease 2019 (COVID-19) pandemic in the Chinese mainland. This transition signifies that the ARGOS framework can be a candidate tool for researchers to solve real-world problems.

Since the dawn of humanity, infectious diseases have been a constant presence, significantly shaping the destinies of all countries in the world by evolving into local epidemics or major pandemics. The COVID-19 pandemic, caused by the SARS-CoV-2 (Severe Acute Respiratory Syndrome CoronaVirus 2) virus, marked the first global pandemic in the digital age. Although the first case has yet to be found, Wuhan's (Hubei, China) Municipal Health Commission reported the initial outbreak in January 2020, identifying a cluster of pneumonia cases. To stop the spread of the virus and protect people's lives, Wuhan, a city with a population of ten million, was placed under lockdown on January 23, 2020. Despite these measures, and subsequent lockdowns in almost all Chinese cities, no one expected that this disease would have spread globally. On 11 March 2020, the World Health Organization (WHO) declared COVID-19 a pandemic. Following this, more countries, including the United Kingdom and the United States, implemented their own lockdowns. This unprecedented spread occurred despite what is arguably the most extensive lockdown in history.

During the pandemic period, various groups worked together to overcome the impacts caused by COVID-19. Doctors and nurses tried their best to rescue infected patients, biologists raced to find vaccines, and other scientists analysed the mechanics of the spreading and provided suggestions to their country's governments [19, 179–181]. About three years later, humans finally overcame the pandemic. However, ongoing research on COVID-19 is important to provide valuable references for similar outbreaks that may occur in the future.

ARGOS-RAL focuses on identifying PDEs. However, when applied to ODE systems, such as the Lorenz and Lotka-Volterra, it penalises true terms. While the original ARGOS is computationally expensive [182, 183], this chapter will analyse COVID-19 data in mainland China using Bayesian ARGOS, a new ARGOS algorithm with small computing costs, to identify ODEs. In Section 7.1, the identification method (Bayesian ARGOS), the COVID-19 dataset, and an initial assumption of the dataset (the SIRD model) are all described. Section 7.2 illustrates the results of the identified model. Section 7.3 discusses the practical significance and drawbacks of the model and identification method. Section 7.4 summarises the whole chapter.

7.1 Methods and Materials

7.1.1 Bayesian ARGOS

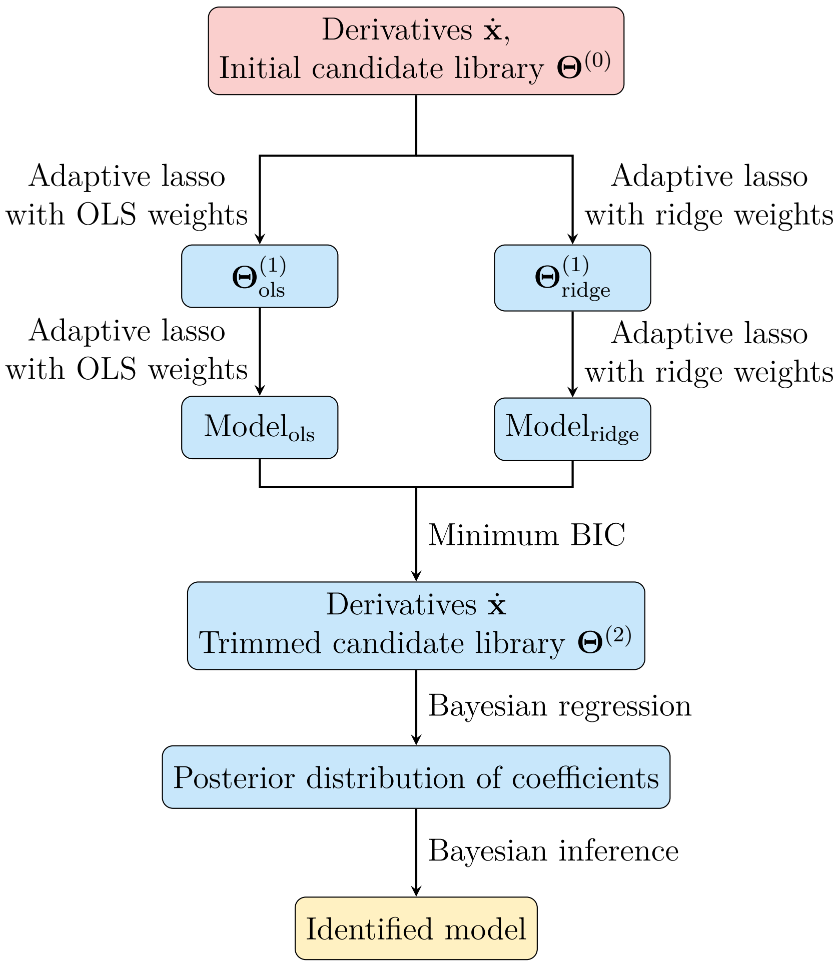

Derivatives $\dot{\mathbf{x}}$,
Initial candidate library $\mathbf{\Theta}^{(0)}$
Adaptive lasso with OLS weights
Adaptive lasso with ridge weights
$\mathbf{\Theta}_{\text{ols}}^{(1)}$
$\mathbf{\Theta}_{\text{ridge}}^{(1)}$
Adaptive lasso with OLS weights
Adaptive lasso with ridge weights
$\text{Model}_{\text{ols}}$
$\text{Model}_{\text{ridge}}$
Minimum BIC
Derivatives $\dot{\mathbf{x}}$
Trimmed candidate library $\mathbf{\Theta}^{(2)}$
Bayesian regression
Posterior distribution of coefficients
Bayesian inference
Identified model

Figure 7.1: Process of Bayesian ARGOS. The process begins with the response variable $\dot{\mathbf{x}}$ and the initial candidate library $\mathbf{\Theta}^{(0)}$. The adaptive lasso is then applied with OLS and ridge regression weights, resulting in two candidate libraries, $\mathbf{\Theta}_{\text{ols}}^{(1)}$ and $\mathbf{\Theta}_{\text{ridge}}^{(1)}$. The adaptive lasso is implemented again to find $\text{Model}_{\text{ols}}$ and $\text{Model}_{\text{ridge}}$ from the candidate terms in $\mathbf{\Theta}_{\text{ols}}^{(1)}$ and $\mathbf{\Theta}_{\text{ridge}}^{(1)}$, respectively. The trimmed candidate library $\mathbf{\Theta}^{(2)}$ is selected from these two models with the minimum Bayesian Information Criterion (BIC). Next, Bayesian regression is performed on this trimmed candidate library to obtain the posterior distribution of coefficients. Finally, Bayesian inference is used to identify the final model.

Bayesian ARGOS enhances the ARGOS framework by integrating Bayesian inference to replace the bootstrap resampling in the sparse regression algorithm. This modification is depicted in Fig. 7.1, focusing specifically on the sparse regression component. Initially, the process applies the adaptive lasso with OLS and ridge regression weights to identify two subsets, $\boldsymbol{\Theta}^{(1)}_{\text{ols}}$ and $\boldsymbol{\Theta}^{(1)}_{\text{ridge}}$, from the preliminary candidate library $\boldsymbol{\Theta}^{(0)}$. Subsequently, these subsets are refined through a second round of the adaptive lasso, aiming to yield sparser models. This process is similar to applying the recurrent adaptive lasso (Section 6.1) with OLS and ridge regression weights and only selecting the second model M^2. As mentioned in Section 6.1, applying the adaptive lasso more than once can significantly shrink the model dimensions.

The BIC is then employed to select the final candidate library $\boldsymbol{\Theta}^{(2)}$ between models from twice adaptive lasso with ols weights ($\text{Model}_{\text{ols}}$) and ridge weights ($\text{Model}_{\text{ridge}}$) based on the minimum BIC value. Instead of estimating the uncertainty of coefficients by bootstrap confidence intervals, Bayesian ARGOS employs Bayesian regression with independent Gaussian priors and Markov chain Monte Carlo (MCMC) sampling to approximate the coefficients' posterior distributions. Here, all Gaussian priors use default settings in R's `rstanarm` package, for which the mean is zero and the standard deviation is 2.5. The final model determination involves statistical inference on coefficients' posterior distributions, for which the 95% credible intervals do not contain zero.

7.1.2 Meta Data information

All epidemic data are collected from daily notifications on the official website of the National Health Commission of China, `http://www.nhc.gov.cn/xcs/yqtb/list_gzbd.shtml` [1]. The first COVID-19 notification published on the national website was on 11 January 2020, and the last one ended on 24 December 2022. From 11 January to 20 January 2020, all notifications only reported the epidemic in Wuhan. After that, notifications were related to all provincial administrative regions of mainland China. At the beginning, despite the inconsistent format, notifications provided important details such as the cumulative number of confirmed cases, the number of patients who have recovered and been discharged from hospitals, instances

[1]All notifications are in Chinese.

of severe conditions, fatalities, close contact cases, individuals released from medical observation, and those still under medical observation. Since 7 February 2020, the uniform daily updates included more helpful information, such as new confirmed, current confirmed, new suspected, current suspected, and new deaths.

To convert all notifications to analysable data, a Python script is used to extract these numbers and create a table with 1078 observations. The first two years of observations (see Fig. 7.2) are used to test Bayesian ARGOS' capability to deal with real-world data because data in this range is more stable than data from the whole time range. The consolidated dataset contains information as follows.

- Date: the date publication;
- Daily new confirmed cases: the daily number of individuals with a positive nucleic acid amplification test (NAAT) for SARS-CoV-2;
- Daily new suspected: the daily number of individuals with clinical symptoms of COVID-19 infection but a negative SARS-CoV-2 NAAT;
- Daily new cured and discharged: the daily number of individuals who recover from COVID-19, i.e., negative SARS-CoV-2 NAATs and no clinical symptoms, after hospitalisation;
- New deaths: the daily number of individuals who die due to COVID-19;
- Current confirmed: the number of individuals who are confirmed cases but exclude cured cases or deaths;
- Current suspected: the number of individuals who are suspected to be infected by COVID-19, excluding confirmed cases;
- Current under medical observation: the number of individuals who are close contacts of confirmed cases and monitored by health professionals;
- Cumulative confirmed: the total number of individuals who have tested positive for SARS-CoV-2, including cured cases or deaths;
- Cumulative cured and discharged: the total number of individuals who recover from COVID-19;
- Cumulative deaths: the total number of deaths due to COVID-19.

Figure 7.2 shows the COVID-19 spread from 10 January 2020 to 10 January 2022. The initial outbreak caused by the shortage of medical supplies and inappropriate

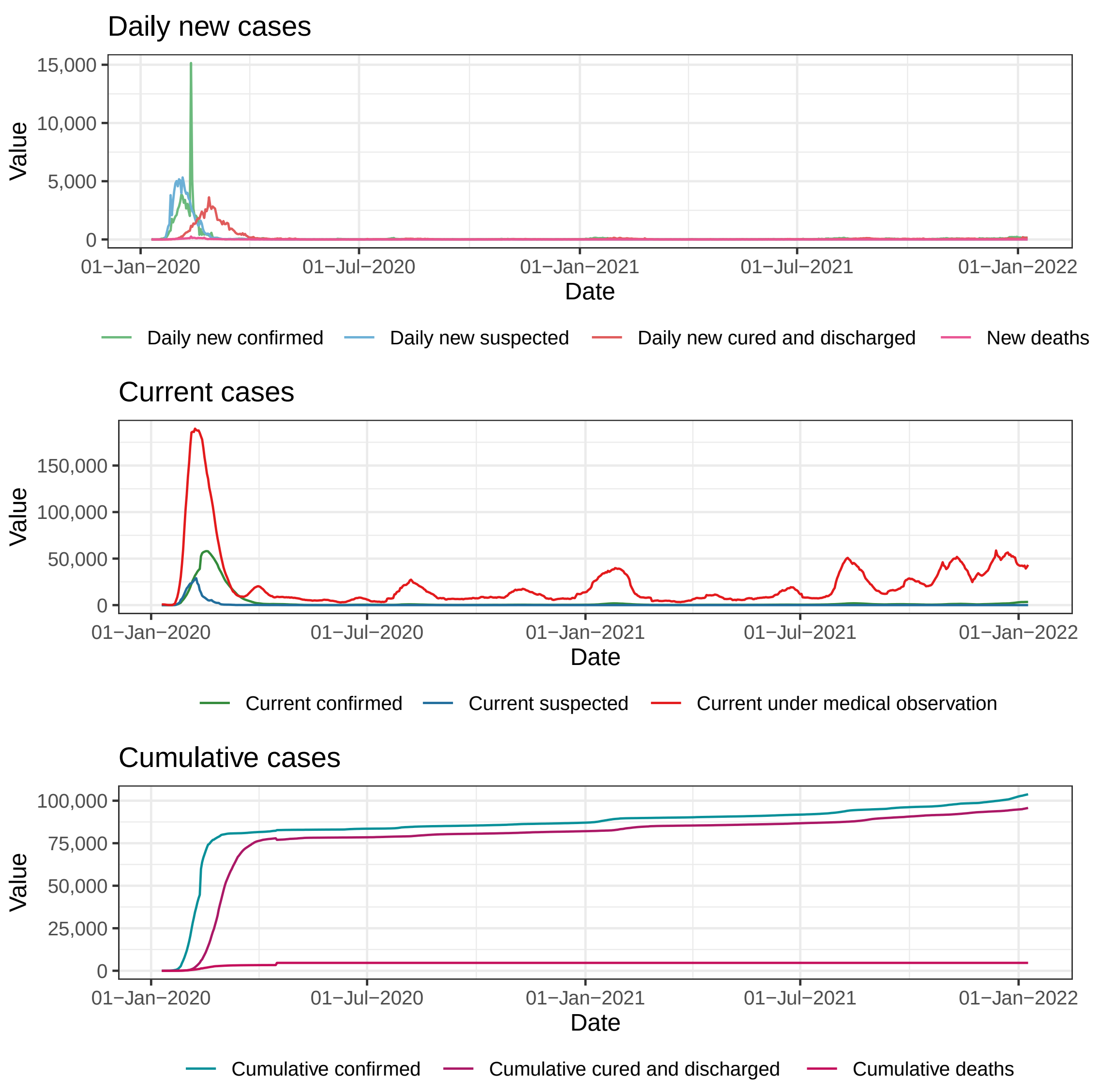


Figure 7.2: COVID-19 spread data from 10 January 2020 to 10 January 2022. The top panel shows daily new cases, including confirmed, suspected, cured and discharged, and deaths. The middle panel presents current cases, including confirmed, suspected, and under medical observation. The bottom panel illustrates cumulative cases, including confirmed, cured and discharged, and deaths over the same period.

safeguards is clearly seen. After this, COVID-19 spread in the Chinese mainland was controlled. All daily new cases were kept at a low level, and all cumulative cases slowly increased, although the current under medical observation was waving. Table 7.1 shows some summary statistics of COVID-19 data between 1 April 2020 and 10 January 2022. During this period, the average daily numbers of newly confirmed, suspected and recovered cases are 33.76, 1.31 and 31.57, respectively. Particularly, the day with the most new confirmed cases (231) was 31 December

Table 7.1: Summary statistics of COVID-19 data from 1 April 2020 to 10 January 2022. This table summarises the minimum, the first quartile, the median, the mean, the third quartile, and the maximum.

	Min	The first quartile	Median	Mean	The third quartile	Max
Daily confirmed	0	11.00	21.0	33.76	39.00	231
Daily suspected	0	0.00	1.0	1.31	2.00	49
Daily cured and discharged	0 0	12.00	20.5	31.57	43.00	213
New deaths	0	0.00	0.0	0.049	0.00	6
Current confirmed	55	279.00	421.5	658.89	909.00	3404
Current suspected	0	1.00	2.0	5.54	4.00	153
Current under medical observation	2892	7001.50	10299.5	16774.09	23477.50	58721
Cumulative confirmed	81589	85164.25	89827.5	89680.80	93029.50	103776
Cumulative cured and discharged	76408	80372.25	84753.0	84416.97	87354.25	95736
Cumulative deaths	3318	4634.00	4636.0	4604.94	4636.00	4636

2021, and the maximum cases of daily cured and discharged from hospitals was 213 on 4 April 2020.

7.1.3 Initial assumption and data preprocessing

Based on the collected data, the SIRD model can be an approximate model to describe the spread dynamics of COVID-19 in China. The SIRD model assumes that under the fixed populations (i.e., $N = S+I+R+D$) with homogeneous mixing, the dynamics of the disease spread are governed by a set of ODEs, describing the rate of movement between compartments. Compared with China's total population, the additional births during the analysing period can be ignored, which meets the fixed population assumption in the SIRD model. Additionally, before implementing the lockdown and facial mask policies, the COVID-19 data fit the homogeneous mixing assumption, i.e., all individuals are randomly in contact with each other, and each individual has the same chance of coming into contact with any other individual. Furthermore, the rate of movement between compartments is assumed to be fixed. Hence, the SIRD model, a set with four ODEs, is used to describe the disease spread. The basic SIRD model is:

$$
\begin{aligned}
\frac{dS}{dt} &= \dot{S} = -\frac{\beta}{N}SI, \\
\frac{dI}{dt} &= \dot{I} = \frac{\beta}{N}SI - \gamma I - \delta I, \\
\frac{dR}{dt} &= \dot{R} = \gamma I, \\
\frac{dD}{dt} &= \dot{D} = \delta I,
\end{aligned}
\tag{7.1}
$$

where β, γ, δ are coefficients of the transmission rate, recovery rate, and mortality rate of the disease, respectively. The basic reproduction number, $R_0 = \beta/(\gamma + \delta)$, can simply represent the spread of the disease. An $R_0 > 1$ indicates that the disease will spread, while an $R_0 < 1$ represents that the outbreak will eventually disappear. Figure 7.3(a) visualises the epidemic's evolution described by the SIRD model over time.

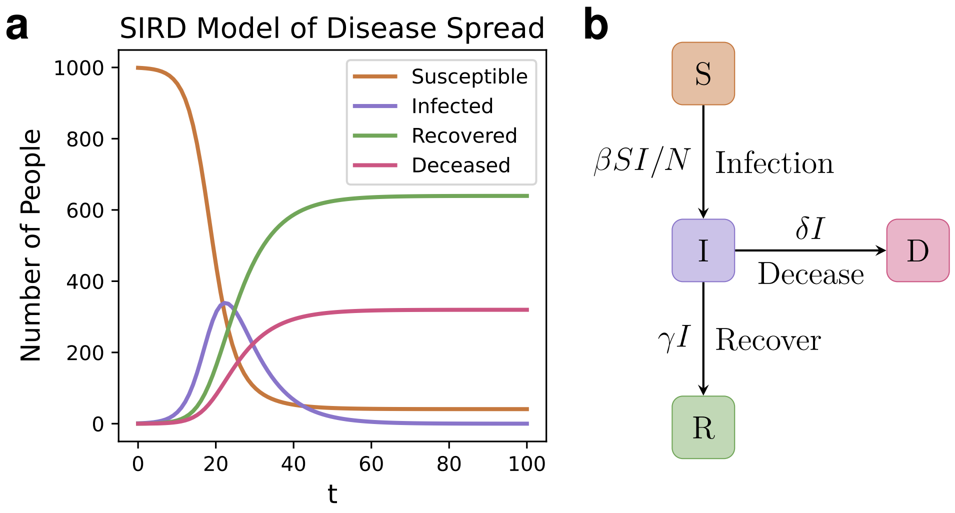


Figure 7.3: The dynamics of a disease described by the SIRD model, Eq. (7.1). (a) The number of people evolution with $\beta = 0.5, \gamma = 0.1, \delta = 0.05$, 999 initial susceptible people, and one infected person. (b) The compartment diagram of the SIRD model illustrates the transition between all four groups.

From Eq. (7.1) and Fig. 7.3(b), the SIRD model can describe diseases without secondary or multiple infections. However, COVID-19 is a more complicated disease in which people may be reinfected multiple times [184]. In the SIRD model, this phenomenon represents the transition from recovered (R) to susceptible (S) and infected (I). Furthermore, in real-world circumstances (Fig. 7.2), the trend of the susceptible group (current suspected) first increases from zero to the peak and then decreases, rather than starting at a high level, see Fig. 7.3 (a). This means that the total population N is much greater than $S+I+R+D$ in COVID-19 data. Due to this, modified SIRD models, including model-driven [179, 180] and data-driven [19], are more suitable than the original one to describe the epidemic dynamics in particular countries and time periods. In particular, Beira and Sebastião [179] suggest including the protected group in the modified model. In the following analysis, the recovered group R will be replaced with the protected group $P = N - S - I - D$. Hence, the model will be called the SIPD model.

Linking with the dataset (Fig. 7.2), each component in the model can be described as follows.

- N: the total population in mainland China;
- S: current suspected cases;
- $\dot{S}$: daily new suspected cases;
- I: current confirmed cases;
- $\dot{I}$: daily new confirmed cases;
- P: the protected people, including cured and discharged from hospitals cases;
- $\dot{P} = \dot{R}$: daily new cured and discharged from hospitals cases;
- D: current (cumulative) deaths;
- $\dot{D}$: daily new deaths;

Therefore, finding the epidemic dynamics involves identifying the relationships between daily new cases, the left-hand side of Eq. (7.1), and current cases, the right-hand side. To identify interpretable dynamics of COVID-19 from Bayesian ARGOS, the candidate library should only contain polynomials, up to the second order, of current cases, as suggested by Eq. (7.1) and recent research [19, 179, 180].

As the COVID-19 dataset does not fully fit the SIRD model's assumptions, some modifications are required to improve the identified model. These modifications can be summarised in three ways: adding extra compartments to the model [179, 185–188], varying the rate of movement between compartments with time [189, 190], and considering the stochastic SIRD model [191]. Here, extra compartments are added to the model to improve the explanation of the identified model. Since these extra compartments (variables) are not recorded in the COVID-19 dataset, each variable will be denoted as C_i. For example, C_1, C_2 and C_3 represent three extra variables.

7.2 Results

To analyse diverse outcomes caused by random samples in ridge regression, Bayesian ARGOS is applied with 100 different random seeds to identify ODEs using the same COVID-19 dataset. Based on the previous description, the maximum polynomial order of the candidate library is set to two:

$$\boldsymbol{\Theta} = \begin{pmatrix} S & I & R & D & SI & SD & \cdots & RD & S^2 & I^2 & R^2 & D^2 \end{pmatrix}. \tag{7.2}$$

From the identification, all identified ODEs are linear models, excluding interaction and squared terms. Interestingly, each $\dot{S}$, $\dot{P}$ and $\dot{D}$ equation has two different models, while $\dot{I}$ only has one. Therefore, Bayesian ARGOS identifies eight (i.e., 2^3) potential linear ODE sets, all of which represent reasonable dynamics of COVID-19.

Equation (7.3) to (7.6) and Fig. 7.4 to 7.7 show the identified models for all compartments and their coefficient posterior distributions. Component $\dot{S}$ has two potential models:

$$\dot{S}_1 = \hat{\beta}_{10} + \hat{\beta}_{11}S + \hat{\beta}_{14}D, \tag{7.3a}$$

$$\dot{S}_2 = \hat{\beta}_{10} + \hat{\beta}_{11}S + \hat{\beta}_{12}I + \hat{\beta}_{14}D. \tag{7.3b}$$

Both Eq. (7.3a) and (7.3b) have similar estimated coefficient distributions for the intercept, S and D, since their 95% credible intervals have overlapping regions. Although the median of $\hat{\beta}_{12}$ in Eq. (7.3b) is -0.0026, close to zero, the estimated coefficient is still significantly smaller than zero because the upper bound of 95% credible interval is -0.00083.

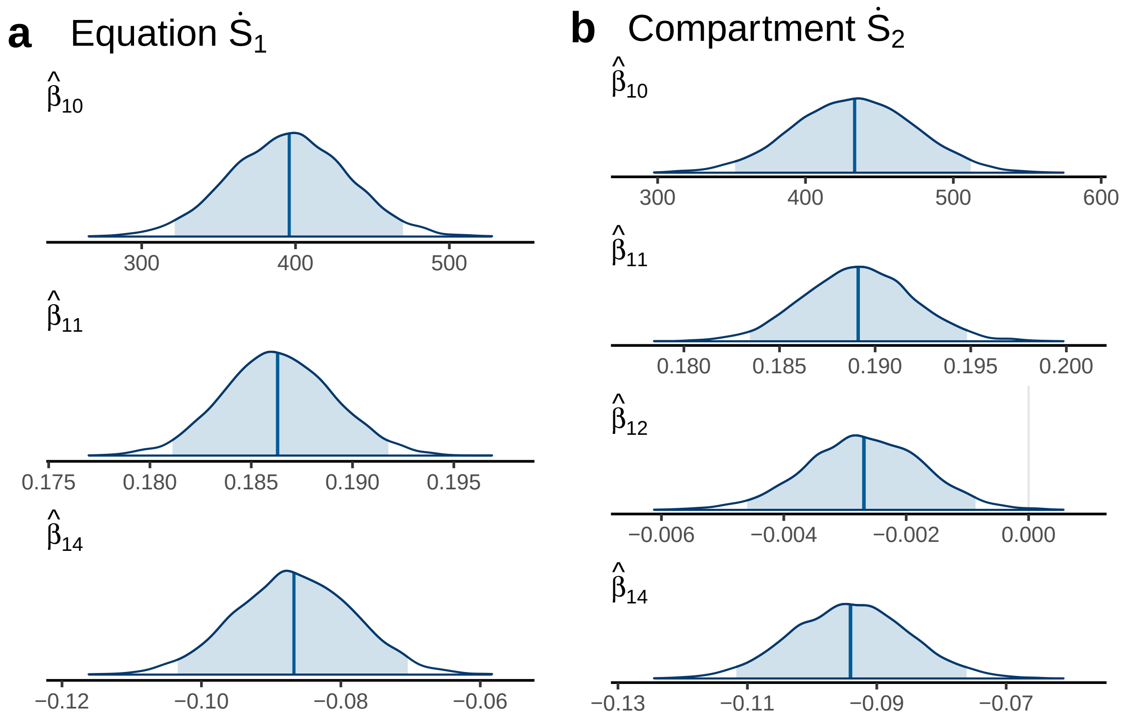


Figure 7.4: Posterior distributions of estimated coefficients for two $\dot{S}$ equations (a and b). The thick line is the median, and the shaded regions are 95% credible intervals. Particularly, $\hat{\beta}_{10}$ is the estimated intercept, $\hat{\beta}_{11}$, $\hat{\beta}_{12}$ and $\hat{\beta}_{14}$ are estimated coefficients of compartment S, I and D, respectively.

Component $\dot{I}$ only has one model:

$$\dot{I} = \hat{\beta}_{21}S + \hat{\beta}_{22}I. \tag{7.4}$$

Both estimated coefficients are positive, and the medians of $\hat{\beta}_{21}$ and $\hat{\beta}_{22}$ are 0.13 and 0.02, respectively.

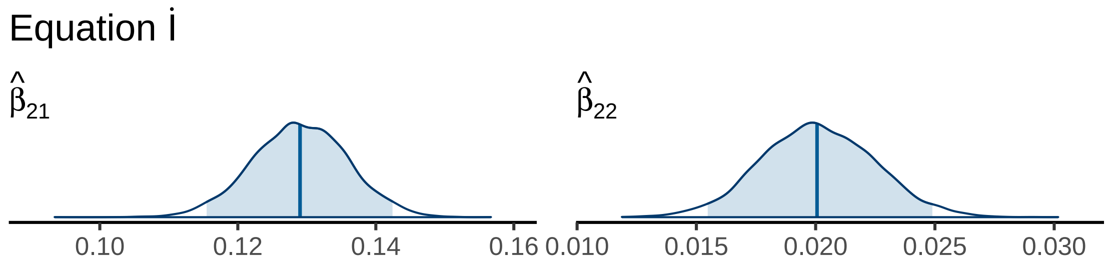


Figure 7.5: Posterior distributions of estimated coefficients for equation $\dot{I}$. The thick lines are the medians, and the shaded regions are 95% credible intervals. Particularly, $\hat{\beta}_{21}$ and $\hat{\beta}_{22}$ are estimated coefficients of compartment S and I, respectively.

Component $\dot{P}$ has two models:

$$\dot{P}_1 = \hat{\beta}_{30} + \hat{\beta}_{31}S + \hat{\beta}_{32}I + \hat{\beta}_{34}D \tag{7.5a}$$

$$\dot{P}_2 = \hat{\beta}_{30} + \hat{\beta}_{31}S + \hat{\beta}_{32}I + \hat{\beta}_{33}P + \hat{\beta}_{34}D \tag{7.5b}$$

In Eq. (7.5a) and (7.5b), $\hat{\beta}_{31}$ has similar posterior distributions because their 95%

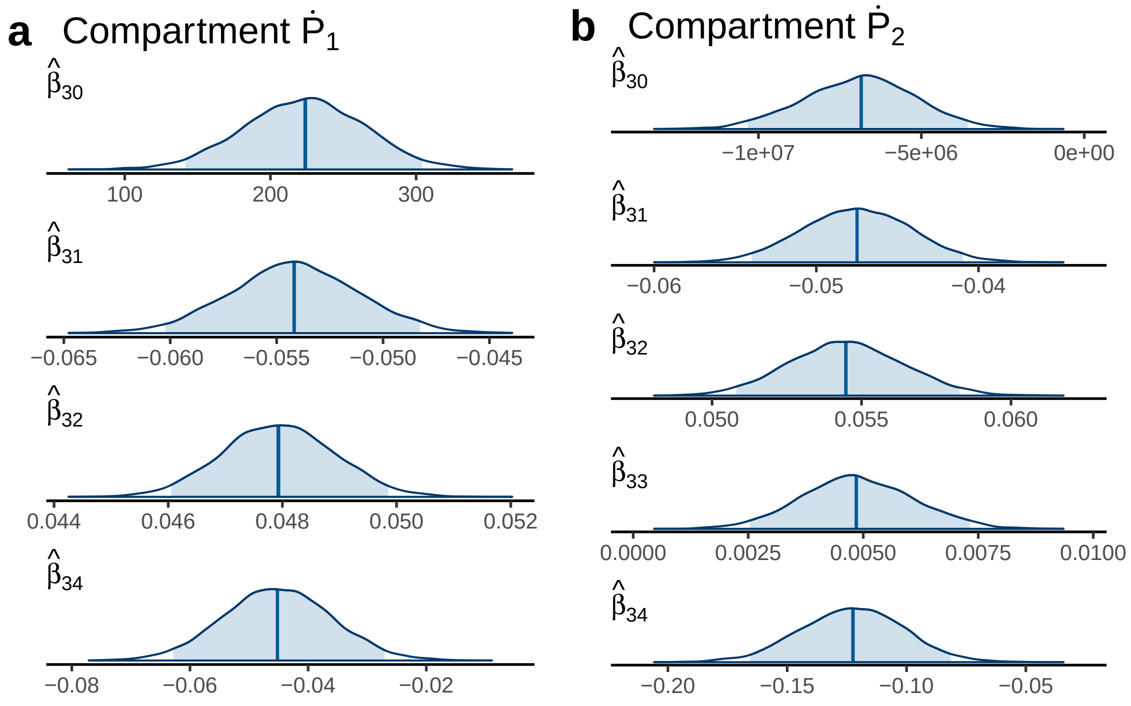


Figure 7.6: Posterior distributions of estimated coefficients for two $\dot{P}$ equations (a and b). The thick line is the median, and the shaded regions are 95% credible intervals. Particularly, $\hat{\beta}_{30}$ is the estimated intercept, $\hat{\beta}_{31}$, $\hat{\beta}_{32}$, $\hat{\beta}_{33}$ and $\hat{\beta}_{34}$ are estimated coefficients of compartment S, I, P and D, respectively.

credible intervals have overlapping areas. However, the coefficient posterior distributions of the $\hat{\beta}_{30}$, $\hat{\beta}_{32}$ and $\hat{\beta}_{34}$ display significant differences. Although $\hat{\beta}_{32}$'s distributions in both equations are close, the upper bound of the 95% credible interval in Eq. (7.5a) is smaller than the lower bound in Eq. (7.5b). The 95% credible interval of the intercept in Eq.(7.5b) is between -1.02×10^7 and -3.57×10^6, much larger than that in Eq.(7.5a) (distributed between 140.46 and 305.16). The interval for $\hat{\beta}_{34}$ in Eq. (7.5a) spans from -0.063 to -0.027, contrasting sharply with the range of -0.16 to -0.079 in Eq. (7.5b). Furthermore, $\hat{\beta}_{33}$ in Eq. (7.5b) is significantly greater than zero.

Component $\dot{D}$ has two models:

$$\dot{D}_1 = \hat{\beta}_{40} + \hat{\beta}_{41}S + \hat{\beta}_{42}I \tag{7.6a}$$

$$\dot{D}_2 = \hat{\beta}_{40} + \hat{\beta}_{41}S + \hat{\beta}_{42}I + \hat{\beta}_{43}P + \hat{\beta}_{44}D \tag{7.6b}$$

Similar to the Component $\dot{P}$, the coefficient posterior distributions of S and I

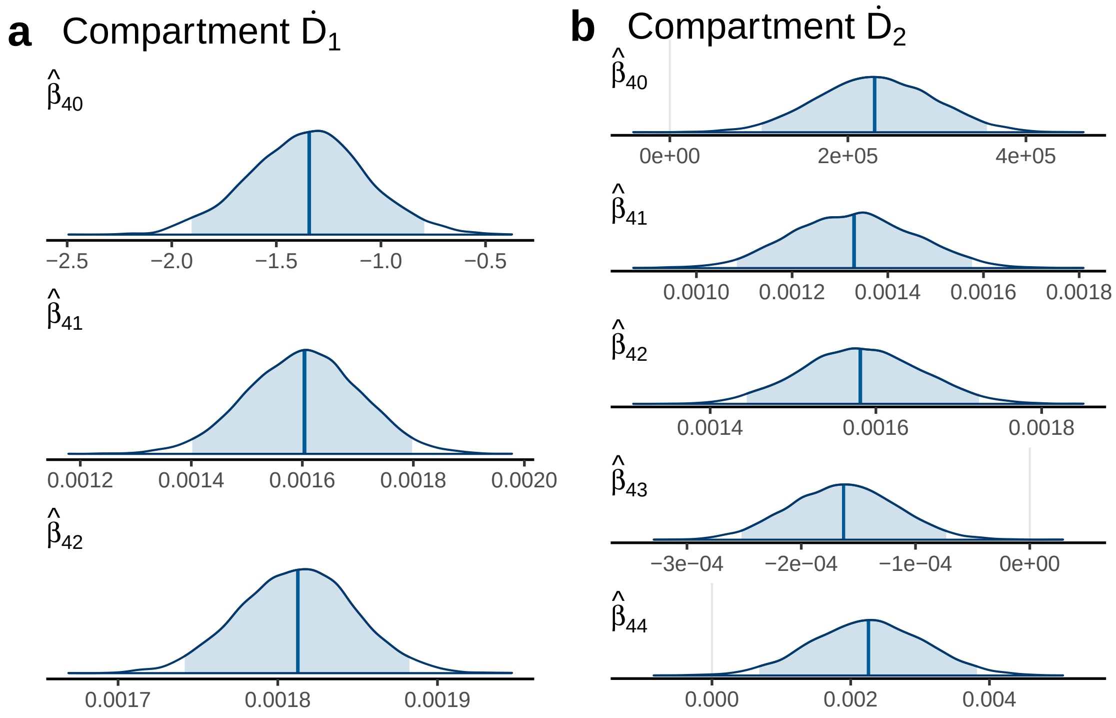


Figure 7.7: Posterior distributions of estimated coefficients for two $\dot{D}$ equations (a and b). The thick line is the median, and the shaded regions are 95% credible intervals. Particularly, $\hat{\beta}_{40}$ is the estimated intercept, $\hat{\beta}_{41}$, $\hat{\beta}_{42}$, $\hat{\beta}_{43}$ and $\hat{\beta}_{44}$ are estimated coefficients of compartment S, I, P and D, respectively.

in Eq. (7.6a) and (7.6b) exhibit similarities, as indicated by their overlapping 95% credible intervals. Particularly, in Eq. (7.6b), $\hat{\beta}_{43}$ and $\hat{\beta}_{44}$ in both equations approach but remain significantly different from zero. The upper bound of 95% credible interval of $\hat{\beta}_{43}$ is -0.000077, whereas the lower bound for $\hat{\beta}_{44}$ is 0.00072. Furthermore, the intercept in Eq. (7.6b) is significantly larger than the corresponding value in Eq. (7.6a). The 95% credible interval of the intercept in Eq. (7.6a) ranges from -1.89 to -0.83, while this interval in Eq. (7.6b) is between 1.10×10^5 and 3.54×10^5.

Figure 7.8 illustrates that Bayesian ARGOS is more frequent in identifying sparser models for $\dot{S}$, $\dot{P}$ and $\dot{D}$ equations, i.e., $\dot{S}_1$, $\dot{P}_1$ and $\dot{D}_1$. As the identification between each component is independent, after multiplying all frequencies, see Table 7.2, the probability of identifying the most parsimonious ODE set, combined with Eq. (7.3a), (7.4), (7.5a) and (7.6a), stands 39.17%, the highest among all eight models. Conversely, the probability of identifying the most complicated model, comprised of Eq. (7.3b), (7.4), (7.5b) and (7.6b), is 1.21%, which is the lowest.

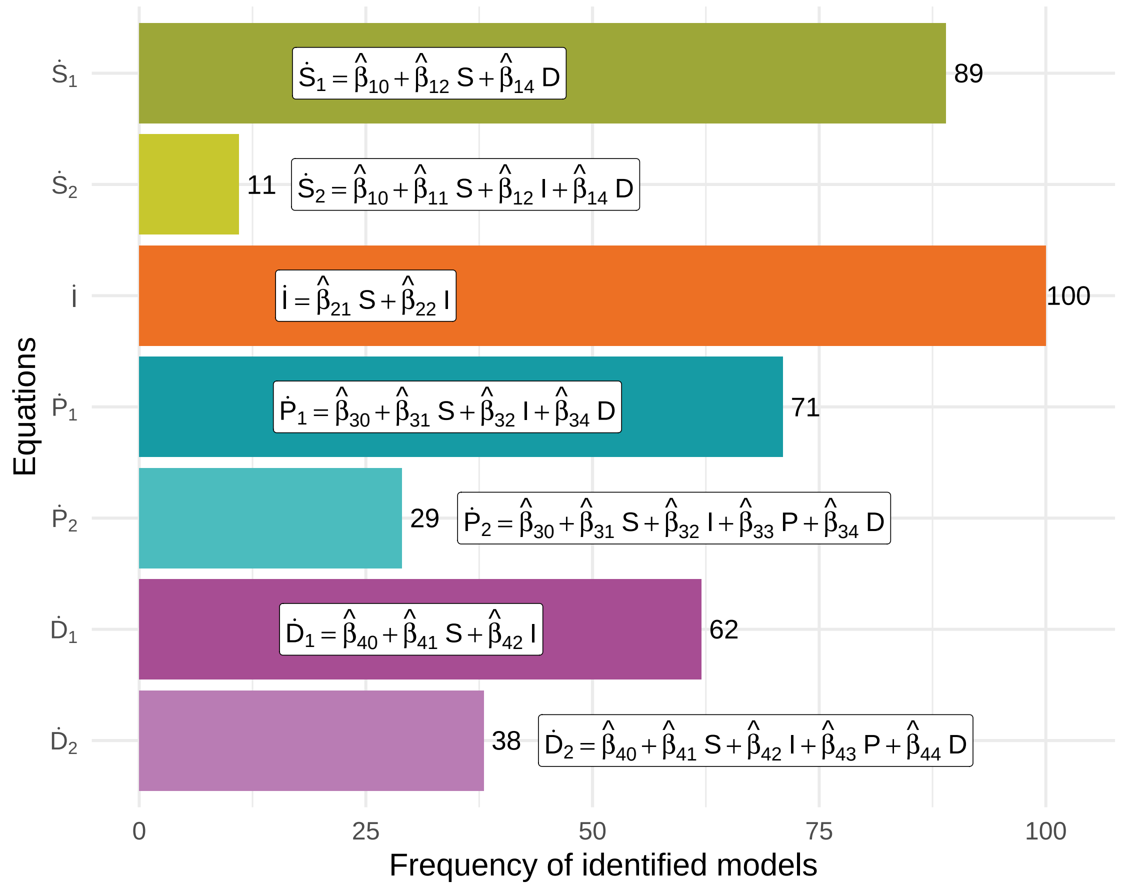


Figure 7.8: Number of times the model is identified from Bayesian ARGOS. Each bar illustrates one equation, and $\hat{\beta}_{ij}$ ($i = \{1, 2, 3, 4\}$ represents the equations, $j = \{0, 1, 2, 3, 4\}$ represents intercept and predictors) are the coefficients of the identified model.

Table 7.2: Probabilities of all identified ODE sets from COVID-19 dataset.

Models	Probability (%)	Models	Probability (%)
$\{\dot{S}_1, \dot{I}, \dot{P}_1, \dot{D}_1\}$	39.18	$\{\dot{S}_2, \dot{I}, \dot{P}_1, \dot{D}_2\}$	4.84
$\{\dot{S}_1, \dot{I}, \dot{P}_1, \dot{D}_2\}$	24.01	$\{\dot{S}_2, \dot{I}, \dot{P}_1, \dot{D}_2\}$	2.97
$\{\dot{S}_1, \dot{I}, \dot{P}_2, \dot{D}_1\}$	16.00	$\{\dot{S}_2, \dot{I}, \dot{P}_2, \dot{D}_1\}$	1.98
$\{\dot{S}_1, \dot{I}, \dot{P}_2, \dot{D}_2\}$	9.81	$\{\dot{S}_2, \dot{I}, \dot{P}_2, \dot{D}_2\}$	1.21

7.3 Discussion

7.3.1 COVID-19 dynamics description

To simplify the explanation of the epidemic dynamics, all coefficient values of the identified model with the highest probability, Eq. (7.7), are estimated by OLS:

$$\begin{aligned}
\dot{S} &= 396.65 + 0.19S - 0.087D, \\
\dot{I} &= 0.13S + 0.020I, \\
\dot{P} &= 223.57 - 0.054S + 0.048I - 0.045D, \\
\dot{D} &= -1.35 + 0.0016S + 0.0018I.
\end{aligned} \tag{7.7}$$

In this model, positive coefficient values indicate an increase in compartment numbers, while negative values denote a decrease. For daily cases in compartment S, the total increase rate is recorded at 0.19, with a mortality rate of 0.087. Within compartment I, the daily infection rates sourced from the suspected group and the group itself are 0.13 and 0.02, respectively. The daily fluctuations in compartment P reveal a suspect rate of 0.054, a mortality rate of 0.045, and a recovery rate from the infection group at 0.048. Meanwhile, in compartment D, the daily mortality rates from the suspected and infection groups stand at 0.0016 and 0.0018, respectively. Figure 7.9 visualises the COVID-19 dynamics enhanced by introducing three extra compartments (C_1, C_2 and C_3) to refine the explanation based on Eq. (7.7).

Despite the meaningful model, Eq. (7.7) shows inconsistencies in daily new cases of S and I, and the transitions from S to D. A significant conflict arises in compartments S and I: the terms suggest that current cases can influence daily new cases, which is unreasonable in real-world observations. In other words, the terms the coefficient of S in $\dot{S}$ ($\beta_{11}S$) and the coefficient of I in $\dot{I}$ ($\beta_{22}I$) are calculated as 0.19 and 0.02, respectively, but ideally, they should be zero. Another contradiction is observed in the mortality rates between S and D. The mortality rate is estimated as 0.087 in $\dot{S}$ but is only 0.00016 in $\dot{D}$. These discrepancies suggest that the model still needs further human modification, i.e., including more compartments in the candidate library, to better capture the dynamics of the disease transmission and effects.

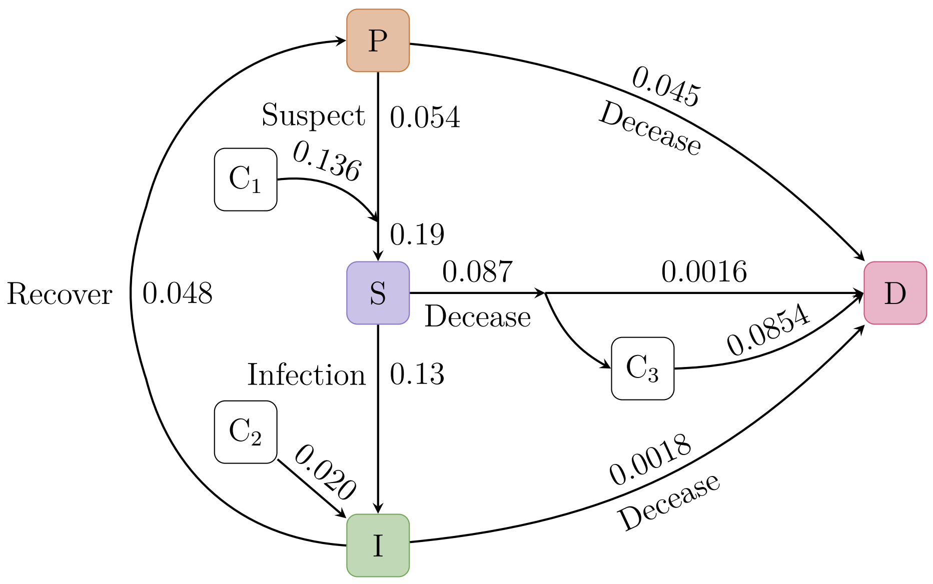


Figure 7.9: Daily compartment evolution diagram of the SIPD model from two-year COVID-19 data in mainland China. Rectangles with S, I, P and D represent compartments in the SIPD model. Rectangles with C$_1$, C$_2$ and C$_3$ are three unknown compartments that are not included in the candidate library. Arrows represent the transformation directions between different compartments. All texts on the arrows represent the events of the transformation. All numbers on the arrows are the transformation rate.

Figure 7.9 visualises the COVID-19 dynamics enhanced by introducing three extra compartments, refining the explanation based on Eq. (7.7). In Eq. (7.7), the total suspect rate should be 0.19, but only 0.054 is assigned to the transition from the protected group. The shortfall of 0.136 in the suspect rate is caused by an unidentified compartment, C$_1$. Similarly, the self-increase observed in the infection compartment might be representative of another unidentified compartment C$_2$, which shares statistical similarities with compartment I, and thus the effects of C$_2$ is credited to the compartment I. Furthermore, introducing an unrecorded compartment C$_3$ can used to reconcile the different mortality rates between equation $\dot{S}$ and $\dot{D}$. Here, the direct mortality rate from S is 0.0016, whereas the rate at which suspected individuals first transition to C$_3$ before decreasing stands at 0.0854. The inclusion of these three compartments suggests that more information is required to accurately interpret the COVID-19 dynamics.

7.3.2 Inconsistent identified models

Tracing the reason for inconsistent results in the Bayesian ARGOS process (see Fig. 7.1), the various ODEs are caused by the cross-validation in determining the shrinkage parameter for ridge regression, $\lambda^{(1)}_{\text{ridge}}$, during the initial adaptive lasso phase. As outlined in Section 2.2.3, cross-validation relies on random sampling, leading to slight variations in the estimated coefficients of the best-fit ridge regression when random seeds are not fixed. These variations are magnified during the regularisation process as the adaptive lasso employs the inverse of the ridge coefficients as the adaptive weights, potentially altering the candidate library $\mathbf{\Theta}^{(1)}_{\text{ridge}}$ after weighted ℓ_1-norm penalisation.

As shown in Fig. 7.1, these discrepancies in the candidate library from the first adaptive lasso with ridge weights, $\mathbf{\Theta}^{(1)}_{\text{ridge}}$, result in different models during the second adaptive lasso with ridge weights. As BIC selects models between the adaptive lasso with OLS and ridge weights, this inconsistency affects the trimmed candidate library $\mathbf{\Theta}^{(2)}$ and leads to varied identification outcomes. In contrast, models derived from adaptive lasso using OLS weights avoid randomness, since the adaptive weights from OLS are based on matrix operations and remain unaffected by random seed variations.

7.4 Summary

This chapter employs Bayesian ARGOS to identify meaningful COVID-19 transmission dynamics in mainland China from data between 10 January 2020 and 10 January 2022. This success enhances the method's capability for automated system identification of real-world data.

- The initial assumption of COVID-19 transmission dynamics is the susceptible-infectious-recovered-deceased (SIRD) model. However, as an individual may be reinfected multiple times, the SIRD model is modified to the SIPD model, which changes the recovery to the protected compartment.
- Bayesian ARGOS identifies multiple possible models due to random sampling of the cross-validation in calculating ridge weights for the adaptive lasso. Running Bayesian ARGOS 100 times with different random seeds can calculate the probabilities of each possible model. The identified models are all linear ODEs, and the sparsest models always have the highest probabilities.
- By visualising the most likely ODE set, the epidemic dynamics require three additional compartments to explain the transmission more reasonably.

CHAPTER 8

Conclusion

8.1 Contributions

This thesis explored the automatic framework for discovering governing equations from collected data, generalising and employing an ARGOS framework [1] and presenting innovative contributions to identify dynamical systems using data-driven methods.

Chapter 1 highlighted the importance of data-centric engineering in recent research works, following Chapter 2 showing the fundamental techniques in statistics and machine learning areas. Chapter 3 and 4 reviewed and evaluated the notable development in data-driven discovering dynamical systems, especially SINDy-based methods. The results indicate that SINDy-based methods lack the trade-off: the absence of fully automated algorithms requires users to engage in manual tuning and iterative usage of semi-automated algorithms.

Chapter 5 developed an automatic numerical differentiation method, ASG, which combines the Savitzky-Golay filter and Gaussian blur and automatically tunes their parameters using machine learning approaches. A comparison with another automatic method, PSG [75], demonstrated that ASG without boundaries is superior to other methods with high accuracy in smoothing and differentiating data. From a signal-processing perspective, this automatic method greatly reduces the time to adjust the parameters when denoising data.

Chapter 6 proposed the ARGOS-RAL, based on the ARGOS framework, employing ASG and the recurrent adaptive lasso to automate the discovery of PDEs from collected data. The ARGOS-RAL has been compared with STRidge, illustrating ARGOS-RAL's efficiency and accuracy in automating model discovery. In particular, the backward stepwise selection performs better than ARGOS-RAL and STRidge in identifying PDEs when the number of equation terms is known. This finding indicates that traditional statistical methods are still powerful under specific conditions.

Chapter 7 employs another ARGOS framework method, Bayesian ARGOS [182], to find governing dynamics from COVID-19 data. The discovered ODEs are explainable and show meaningful results, indicating Bayesian ARGOS's ability to analyse real-world datasets.

An additional contribution of this thesis is demonstrating that the three employed tests of SNR, sample size, and pure white noise with different variances provide a systematic testing template for evaluating the uncertainty of the results caused by the random samples in machine learning techniques. These tests enable users to verify the acceptance rates of the algorithm's outcomes. This approach helps prevent the false success of an algorithm due to random chance.

8.2 Future Directions

The ARGOS framework in this thesis provides robust outcomes for identifying governing equations of dynamical systems, which is crucial for engineers to extract valuable information by modelling collected data. However, the uncertainty quantification for the identified model is based on controlled variable experiments that repeat the running of ARGOS-RAL 100 times under the same conditions (SNR or sample size) but change one component (random noise or samples), which is computationally expensive. This drawback is significant in evaluating the original

ARGOS [1], which requires about three days with 32 CPU cores and 180 GB RAM to identify one equation of an ODE system. Although ARGOS-RAL in this thesis gets rid of 2000-times bootstrapping for calculating confidence intervals, the building of 1200 candidate libraries (each of them has 5000 points) for the Navier-Stokes equation still costs about nine hours with 60 CPU threads. Without high-performance computers, such an evaluation is difficult to implement. Combining uncertainty qualification theory [192] and post-selection inference [193, 194] may be a potential solution to qualify the algorithm's uncertainty. Integrating these methods into the ARGOS may create a more efficient and robust framework, but the statistical community focuses on developing secure implementation procedures, hindering the development of the application in this instance.

Identifying stochastic differential equations is another important research direction that the ARGOS framework can be used for. As coefficients of stochastic differential equations are mixed effects models for which the random effects are related to time, the identification methods for normal differential equations are inappropriate. Although some proposed methods employ SINDy [125], sparse Bayesian inference [195–197], and neural networks [198] to identify stochastic differential equations successfully, these methods can only identify specific equations and still require users to determine parameters with their expert knowledge. Integrating Bayesian hierarchical modelling and Bayesian ARGOS can be a potential solution for identifying a general form of stochastic differential equations. Such a hierarchical ARGOS can identify both stochastic differential equations and equations whose coefficients change with locations, such as using one model to describe COVID-19 dynamics around the world.

After finding models, engineers and scientists are interested in the prediction ability of the models. However, current differential equation solvers can only solve specific equations, not universal solvers. Although some numerical solvers, such as `odeint` in `Python` and `ode45` in `MATLAB`, can solve most ODEs and some simple PDEs, they still require users to define the equation expressions in `Python` or `MATLAB`. If these equations can be automatically converted to programming functions, solving the identified differential equations from the ARGOS framework will enable it to automatically discover and predict dynamical systems simultaneously. In this way, engineers and scientists can easily forecast the evolution of a system.

Automating the discovery of dynamical systems can speed up data analysis across a range of scientific disciplines, including engineering, physics, chemistry, biology and economics. These methods enable engineers to create large models with numerous variables, which would be challenging and time-consuming to develop manually. The proposed algorithms reduce the risk of errors and inconsistencies often found in manually developed models, thereby enhancing their accuracy and reliability. Although physics-informed neural networks (PINNs) tolerate more noise than traditional statistical methods, as mentioned in Section 3.1, these methods currently have a contradiction against automation: all parameters are set empirically without fundamental mathematical theory. This makes PINN an outstanding semi-automatic method that requires human intervention. If mathematicians and engineers can solve this problem in the future, research in all fields will flourish. In summary, methods for automatically discovering dynamical systems from data are important for advancing scientific research and accelerating technological progress.

Bibliography

[1] K. Egan, W. Li, and R. Carvalho, "Automatically discovering ordinary differential equations from data with sparse regression," *Communications Physics*, vol. 7, no. 1, pp. 1–10, Jan. 2024. (document), 1.1, 1.3, 3.1, 3.3.1, 3.4.2, 4, 4.1, 4.1.2, 4.3.1, 6, 6.1.1, 6.1.2, 6.1.2, 6.2.1, 6.2.1, 6.3, 8.1, 8.2

[2] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, Jun. 2007. 1.1, 3.1

[3] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016. 1.1, 1.3, 3, 3.1, 3.1, 3.2, 3.2.1, 3.2.1, 3.2.1, 3.2.2, 3.3.1, 3.4, 4, 4.1.1, 6.1.1, 6.3

[4] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science Advances*, vol. 3, no. 4, p. e1602614, Apr. 2017. 1.1, 2.5.1, 2.5.3, 3, 3.1, 3.1, 3.2.1, 3.3, 3.3, 3.3.1, 3.3.2, 3.3.2, 3.4, 1, 3.4.1, 2, 5, 6, 6.1.1, 6.2.1, 6.2.2, 6.2.2, 6.2.4, 6.3

[5] J. Edghill and D. Towill, "The use of system dynamics in manufacturing systems engineering," *Transactions of the Institute of Measurement and Control*, vol. 11, no. 4, pp. 208–216, Nov. 1989. 1.1

[6] P. Denno, C. Dickerson, and J. A. Harding, "Dynamic production system identification for smart manufacturing systems," *Special Issue on Smart Manufacturing*, vol. 48, pp. 192–203, Jul. 2018.

[7] J. Zhang, P. Wang, R. Yan, and R. X. Gao, "Long short-term memory for machine remaining life prediction," *Special Issue on Smart Manufacturing*, vol. 48, pp. 78–86, Jul. 2018.

[8] J. Xie, Z. Chai, L. Xu, X. Ren, S. Liu, and X. Chen, "3D temperature field prediction in direct energy deposition of metals using physics informed neural

network," *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 5, pp. 3449–3468, Mar. 2022. 1.1

[9] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh, "Analyses of internal structures and defects in materials using physics-informed neural networks," *Science Advances*, vol. 8, no. 7, p. eabk0644, Feb. 2022. 1.1

[10] S. Hiemer and S. Zapperi, "From mechanism-based to data-driven approaches in materials science," *Materials Theory*, vol. 5, no. 1, p. 4, Sep. 2021. 1.1

[11] G. Makrygiorgos, A. J. Berliner, F. Shi, D. S. Clark, A. P. Arkin, and A. Mesbah, "Data-driven flow-map models for data-efficient discovery of dynamics and fast uncertainty quantification of biological and biochemical systems," *Biotechnology and Bioengineering*, vol. 120, no. 3, pp. 803–818, Mar. 2023. 1.1

[12] K. Worden, C. R. Farrar, J. Haywood, and M. Todd, "A review of nonlinear dynamics applications to structural health monitoring," *Structural Control and Health Monitoring*, vol. 15, no. 4, pp. 540–567, Jun. 2008. 1.1

[13] X. Hong, R. Mitchell, S. Chen, C. Harris, K. Li, and G. Irwin, "Model selection approaches for non-linear system identification: a review," *International Journal of Systems Science*, vol. 39, no. 10, pp. 925–946, Oct. 2008. 1.1

[14] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 2005. 1.1

[15] M. Schmidt and H. Lipson, "Distilling Free-Form Natural Laws from Experimental Data," *Science*, vol. 324, no. 5923, pp. 81–85, Apr. 2009. 1.1, 3, 3.1

[16] X. Xun, J. Cao, B. Mallick, A. Maity, and R. J. Carroll, "Parameter Estimation of Partial Differential Equation Models," *Journal of the American Statistical Association*, vol. 108, no. 503, pp. 1009–1020, Sep. 2013. 3.1

[17] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019. 1.1, 3, 3.1

[18] P. Y. Lu, J. Ariño Bernad, and M. Soljačić, "Discovering sparse interpretable dynamics from partial observations," *Communications Physics*, vol. 5, no. 1, p. 206, Aug. 2022. 1.1, 6.3

[19] Y.-X. Jiang, X. Xiong, S. Zhang, J.-X. Wang, J.-C. Li, and L. Du, "Modeling and prediction of the transmission dynamics of COVID-19 based on the SINDy-LM method," *Nonlinear Dynamics*, vol. 105, no. 3, pp. 2775–2794, Aug. 2021. 1.1, 3.1, 7, 7.1.3

[20] S. Maddu, B. L. Cheeseman, I. F. Sbalzarini, and C. L. Müller, "Stability selection enables robust learning of differential equations from limited noisy data,"

Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 478, no. 2262, p. 20210916, Jun. 2022.

[21] Y. Cai, X. Wang, G. Joós, and I. Kamwa, “An Online Data-Driven Method to Locate Forced Oscillation Sources From Power Plants Based on Sparse Identification of Nonlinear Dynamics (SINDy),” *IEEE Transactions on Power Systems*, vol. 38, no. 3, pp. 2085–2099, May 2023. 1.1, 7

[22] X. Sun, J. Qian, and J. Xu, “Compressive-sensing model reconstruction of nonlinear systems with multiple attractors,” *International Journal of Mechanical Sciences*, vol. 265, p. 108905, Mar. 2024. 1.1

[23] B. C. Daniels and I. Nemenman, “Automated adaptive inference of phenomenological dynamical models,” *Nature Communications*, vol. 6, no. 1, p. 8133, Aug. 2015. 1.1

[24] S.-M. Udrescu and M. Tegmark, “AI Feynman: A physics-inspired method for symbolic regression,” *Science Advances*, vol. 6, no. 16, p. eaay2631, Apr. 2020. 3.1

[25] R. Guimerà, I. Reichardt, A. Aguilar-Mogas, F. A. Massucci, M. Miranda, J. Pallarès, and M. Sales-Pardo, “A Bayesian machine scientist to aid in the solution of challenging scientific problems,” *Science Advances*, vol. 6, no. 5, p. eaav6971, Jan. 2020.

[26] J. L. Callaham, J. V. Koch, B. W. Brunton, J. N. Kutz, and S. L. Brunton, “Learning dominant physical processes with data-driven balance models,” *Nature Communications*, vol. 12, p. 1016, Dec. 2021. 1.1

[27] Z. Chen, Y. Liu, and H. Sun, “Physics-informed learning of governing equations from scarce data,” *Nature Communications*, vol. 12, no. 1, p. 6136, Oct. 2021. 1.1, 3.1

[28] P. Thanasutives, T. Morita, M. Numao, and K.-i. Fukui, “Noise-aware physics-informed machine learning for robust PDE discovery,” *Machine Learning: Science and Technology*, vol. 4, no. 1, p. 015009, Mar. 2023. 1.1, 3.1

[29] K. Duraisamy, G. Iaccarino, and H. Xiao, “Turbulence Modeling in the Age of Data,” *Annual Review of Fluid Mechanics*, vol. 51, no. 1, pp. 357–377, Jan. 2019. 1.1

[30] C. Hansen, X. I. A. Yang, and M. Abkar, “Data-Driven Dynamical System Models of Roughness-Induced Secondary Flows in Thermally Stratified Turbulent Boundary Layers,” *Journal of Fluids Engineering*, vol. 145, no. 061102, Mar. 2023. 1.1

[31] A. Cortiella, K. C. Park, and A. Doostan, “Sparse identification of nonlinear dynamical systems via reweighted ℓ_1-regularized least squares,” *Computer Methods in Applied Mechanics and Engineering*, vol. 376, p. 113620, Apr. 2021. 1.1, 2.4.6, 2.4.6, 3.1, 3.1, 6.1.2

[32] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York: Springer, 2009. 1.4, 2.1, 2.1.4, 2.1.4, 2.2.2, 2.2.3, 2.2.4, 2.2.4, 2.3, 2.4, 2.4.3

[33] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 2nd ed., ser. Springer Texts in Statistics. New York: Springer, 2021, vol. 103. 1.4, 2.1.1, 2.2.1, 2.2.3, 2.2.4, 2.3, 4.1, 4.1, 6.5

[34] T. Hastie, *Statistical learning with Sparsity : the lasso and generalizations*, ser. Monographs on statistics and applied probability (Series) ; 143. Chapman and Hall/CRC, 2015. 1.4

[35] J. J. Faraway, *Linear models with R*, ser. Texts in statistical science. Boca Raton, Fla., ; London: Chapman & Hall/CRC, 2005. 2.1, 2.1, 2.1.1, 2.1.1

[36] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning : With Applications in R.* New York: Springer, 2013. 2.1, 2.1.1, 2.2, 2.2.1, 2.2.2, 2.2.2, 2.2.2, 2.2.2, 2.3, 2.4, 2.4.2

[37] J. J. Faraway, *Linear models with R*, 2nd ed., ser. Texts in statistical science. New York: Chapman & Hall/CRC, 2014. 2.1.1, 2.1.5, 2.3

[38] D. S. Young, *Handbook of regression methods*, 1st ed. New York: CRC Press, 2017. 2.1.1, 2.1.5, 2.1.5, 2.1.5, 2.2.2, 2.2.2, 2.4.3

[39] G. Petris, S. Petrone, and P. Campagnoli, *Dynamic Linear Models with R.* New York: Springer, 2009. 2.1.2

[40] A. Gałecki and T. Burzykowski, *Linear Mixed-Effects Models Using R: A Step-by-Step Approach.* New York: Springer, 2013. 2.1.2, 2.1.2

[41] R. Christensen, W. Johnson, A. Branscum, and T. E. Hanson, *Bayesian Ideas and Data Analysis: An Introduction for Scientists and Statisticians.* Boca Raton: CRC Press, Jul. 2010. 2.1.3

[42] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian Model Averaging: A Tutorial," *Statistical Science*, vol. 14, no. 4, pp. 382–401, 1999. 2.1.3

[43] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, 3rd ed. New York: Chapman and Hall/CRC, Jul. 2015. 2.1.3, 2.1.3

[44] Samprit Chatterjee and Ali S. Hadi, *Regression Analysis by Example*, 5th ed. Somerset: John Wiley & Sons, Ltd, 2012. 2.1.5, 2.1.5, 2.3

[45] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970. 2.1.5, 2.4.1

[46] H. Akaike, “Information Theory and an Extension of the Maximum Likelihood Principle,” in *Selected Papers of Hirotugu Akaike*, E. Parzen, K. Tanabe, and G. Kitagawa, Eds. New York: Springer, 1998, pp. 199–213. 2.2.2

[47] H. Zou, T. Hastie, and R. Tibshirani, “On the “degrees of freedom” of the lasso,” *The Annals of Statistics*, vol. 35, no. 5, pp. 2173–2192, Oct. 2007. 2.2.2

[48] G. Schwarz, “Estimating the Dimension of a Model,” *The Annals of Statistics*, vol. 6, no. 2, pp. 461 – 464, 1978. 2.2.2

[49] Y. Yang, “Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation,” *Biometrika*, vol. 92, no. 4, pp. 937–950, Dec. 2005. 2.2.2, 12

[50] S. I. Vrieze, “Model selection and psychological theory: A discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC).” *Psychological Methods*, vol. 17, no. 2, pp. 228–243, 2012.

[51] K. Aho, D. Derryberry, and T. Peterson, “Model selection for ecologists: the worldviews of AIC and BIC,” *Ecology*, vol. 95, no. 3, pp. 631–636, Mar. 2014. 2.2.2, 12

[52] B. Efron, “Bootstrap Methods: Another Look at the Jackknife,” *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, Jan. 1979. 2.2.4

[53] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, ser. Monographs on statistics and applied probability. New York: Chapman & Hall, 1993, no. 57. 2.2.4, 2.2.4

[54] A. C. Davison and D. V. Hinkley, Eds., *Bootstrap Methods and their Application*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press, 1997. 2.2.4

[55] B. Efron and R. Tibshirani, “Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy,” *Statistical Science*, vol. 1, no. 1, pp. 54–75, Feb. 1986. 2.2.4

[56] A. M. Zoubir and D. R. Iskander, *Bootstrap Techniques for Signal Processing.* Cambridge: Cambridge University Press, 2004. 2.2.4

[57] M. L. Rizzo, *Statistical computing with R*, 2nd ed. Boca Raton: CRC Press, Taylor & Francis Group, 2019. 2.2.4

[58] P. C. Hansen, “7. Parameter-Choice Methods,” in *Rank-Deficient and Discrete Ill-Posed Problems*, ser. Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics, Jan. 1998, pp. 175–208. 2.4

[59] R. Tibshirani, “Regression Shrinkage and Selection Via the Lasso,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996. 2.4.2, 12

[60] S. Chatterjee and J. S. Simonoff, *Handbook of Regression Analysis With Applications in R*, 2nd ed. Wiley, 2020. 2.4.2

[61] J. H. Friedman, T. Hastie, and R. Tibshirani, "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software, Articles*, vol. 33, no. 1, pp. 1–22, 2010. 2.4.2, 2.4.3, 2.4.6, 6.1.2

[62] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. 2.4.3

[63] J. Fan and R. Li, "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001. 2.4.4

[64] H. Zou, "The Adaptive Lasso and Its Oracle Properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006. 2.4.4, 2.4.4, 6.1.2, 12, 6.2.2

[65] Peter Bühlmann and Lukas Meier, "Discussion: One-step sparse estimates in nonconcave penalized likelihood models," *The Annals of Statistics*, vol. 36, no. 4, pp. 1534–1541, Aug. 2008. 2.4.4

[66] H. Zou and H. H. Zhang, "On the adaptive elastic-net with a diverging number of parameters," *The Annals of Statistics*, vol. 37, no. 4, pp. 1733 – 1751, 2009. 2.4.5

[67] P. C. Hansen, "Analysis of Discrete Ill-Posed Problems by Means of the L-Curve," *SIAM Review*, vol. 34, no. 4, pp. 561–580, Dec. 1992. 2.4.6, 2.4.6, 6.1.2

[68] J. Nasehi Tehrani, A. McEwan, C. Jin, and A. van Schaik, "L1 regularization method in electrical impedance tomography by using the L1-curve (Pareto frontier curve)," *Applied Mathematical Modelling*, vol. 36, no. 3, pp. 1095–1105, Mar. 2012. 2.4.6, 2.4.6, 6.1.2

[69] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004. 2.4.6

[70] A. Cultrera and L. Callegaro, "A simple algorithm to find the L-curve corner in the regularisation of ill-posed inverse problems," *IOP SciNotes*, vol. 1, no. 2, p. 025004, Aug. 2020. 2.4.6, 6.1.2

[71] E. van den Berg and M. P. Friedlander, "Probing the Pareto Frontier for Basis Pursuit Solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, Jan. 2009. 2.4.6

[72] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, ser. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 2007. 2.5, 2.5.1

[73] D. Gottlieb, J. S. Hesthaven, and S. Gottlieb, Eds., *Spectral Methods for Time-Dependent Problems*, ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge: Cambridge University Press, 2007. 2.5

[74] J. Cullum, "Numerical Differentiation and Regularization," *SIAM Journal on Numerical Analysis*, vol. 8, no. 2, pp. 254–265, Jun. 1971. 2.5, 2.5.3

[75] F. V. Breugel, J. N. Kutz, and B. W. Brunton, "Numerical Differentiation of Noisy Data: A Unifying Multi-Objective Optimization Framework," *IEEE Access*, vol. 8, pp. 196 865–196 877, 2020. 2.5.1, 2.5.2, 2.5.3, 2.5.3, 2.5.4, 2.5.4, 2.5.4, 3.1, 5, 6.1.1, 8.1

[76] K. Ahnert and M. Abel, "Numerical differentiation of experimental data: local versus global methods," *Computer Physics Communications*, vol. 177, no. 10, pp. 764–774, Nov. 2007. 2.5.2, 2.5.3, 3.1

[77] R. C. Gonzalez and R. E. Woods, *Digital image processing.* New York: Pearson, 2018. 2.5.2, 2.5.2, 2.5.2, 5.1

[78] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964. 2.5.2, 2.5.3, 3.1, 6.1.1

[79] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, Nov. 1992. 2.5.2

[80] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000. 2.5.2

[81] T. Holton, "Infinite impulse response filters," in *Digital Signal Processing: Principles and Applications.* Cambridge: Cambridge University Press, 2021, pp. 499–596. 2.5.2

[82] I. Knowles and R. J. Renka, "Methods for numerical differentiation of noisy data," in *Variational and Topological Methods: Theory, Applications, Numerical Simulations, and Open Problems (2012)*, vol. 21. Flagstaff, Arizona, USA: Electronic Journal of Differential Equations, 2014, pp. 235–246. 2.5.3, 5

[83] R. W. Schafer, "What Is a Savitzky-Golay Filter? [Lecture Notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 111–117, Jul. 2011. 2.5.3, 6.1.1

[84] G. Vivó-Truyols and P. J. Schoenmakers, "Automatic Selection of Optimal Savitzky-Golay Smoothing," *Analytical Chemistry*, vol. 78, no. 13, pp. 4598–4608, Jul. 2006. 2.5.3

[85] S. R. Krishnan and C. S. Seelamantula, "On the Selection of Optimum Savitzky-Golay Filters," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 380–391, Jan. 2013. 2.5.3

[86] M. Sadeghi, F. Behnia, and R. Amiri, “Window Selection of the Savitzky–Golay Filters for Signal Recovery From Noisy Measurements,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 8, pp. 5418–5427, Aug. 2020. 2.5.3

[87] Y. Liu, B. Dang, Y. Li, H. Lin, and H. Ma, “Applications of Savitzky-Golay Filter for Seismic Random Noise Reduction,” *Acta Geophysica*, vol. 64, no. 1, pp. 101–124, Feb. 2016. 2.5.3

[88] I. G. Roy, “An optimal Savitzky–Golay derivative filter with geophysical applications: an example of self-potential data,” *Geophysical Prospecting*, vol. 68, no. 3, pp. 1041–1056, 2020. 2.5.3

[89] W. S. Cleveland, “Robust Locally Weighted Regression and Smoothing Scatterplots,” *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, Dec. 1979. 2.5.3

[90] W. S. Cleveland and S. J. Devlin, “Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting,” *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 596–610, Sep. 1988. 2.5.3

[91] G. Romo-Cárdenas, G. J. Avilés-Rodríguez, J. d. D. Sánchez-López, M. Cosío-León, P. A. Luque, C. M. Gómez-Gutiérrez, J. I. Nieto-Hipólito, M. Vázquez-Briseño, and C. X. Navarro-Cota, “Nyquist-Shannon theorem application for Savitzky-Golay smoothing window size parameter determination in bio-optical signals,” *Results in Physics*, vol. 11, pp. 17–22, Dec. 2018. 2.5.3

[92] F. Lejarza and M. Baldea, “Data-driven discovery of the governing equations of dynamical systems via moving horizon optimization,” *Scientific Reports*, vol. 12, no. 1, p. 11836, Jul. 2022. 2.5.3

[93] J. M. Varah, “A Spline Least Squares Method for Numerical Parameter Estimation in Differential Equations,” *SIAM Journal on Scientific and Statistical Computing*, vol. 3, no. 1, pp. 28–46, Mar. 1982. 3.1

[94] J. P. Crutchfield and B. S. Mcnamara, “Equations of motion from a data series,” *Complex Systems*, vol. 1, pp. 417–452, 1987. 3.1

[95] M. Bär, R. Hegger, and H. Kantz, “Fitting partial differential equations to space-time dynamics,” *Physical Review E*, vol. 59, no. 1, pp. 337–342, Jan. 1999. 3.1

[96] T. Müller and J. Timmer, “Fitting parameters in partial differential equations from partially observed noisy data,” *Physica D: Nonlinear Phenomena*, vol. 171, no. 1, pp. 1–7, Oct. 2002.

[97] H. Liang and H. Wu, “Parameter Estimation for Differential Equation Models Using a Framework of Measurement Error in Regression Models,” *Journal of the American Statistical Association*, vol. 103, no. 484, pp. 1570–1583, Dec. 2008. 3.1

[98] H. Wu, A. A. Ding, and V. De Gruttola, “Estimation of HIV dynamic parameters,” *Statistics in Medicine*, vol. 17, no. 21, pp. 2463–2485, Nov. 1998. 3.1

[99] H. Wu and A. A. Ding, “Population HIV-1 Dynamics In Vivo: Applicable Models and Inferential Tools for Virological Data from AIDS Clinical Trials,” *Biometrics*, vol. 55, no. 2, pp. 410–418, Jun. 1999. 3.1

[100] H. Putter, S. H. Heisterkamp, J. M. A. Lange, and F. de Wolf, “A Bayesian approach to parameter estimation in HIV dynamical models,” *Statistics in Medicine*, vol. 21, no. 15, pp. 2199–2214, Aug. 2002. 3.1

[101] H. Xu and D. Zhang, “Robust discovery of partial differential equations in complex situations,” *Physical Review Research*, vol. 3, no. 3, p. 033270, Sep. 2021. 3.1

[102] P. Zheng, T. Askham, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin, “A Unified Framework for Sparse Relaxed Regularized Regression: SR3,” *IEEE Access*, vol. 7, pp. 1404–1423, 2019. 3.1

[103] S. Rudy, A. Alla, S. L. Brunton, and J. N. Kutz, “Data-Driven Identification of Parametric Partial Differential Equations,” *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 2, pp. 643–660, Jan. 2019. 3.1, 3.1

[104] E. Kaiser, J. N. Kutz, and S. L. Brunton, “Sparse identification of nonlinear dynamics for model predictive control in the low-data limit,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, p. 20180335, Nov. 2018.

[105] D. E. Shea, S. L. Brunton, and J. N. Kutz, “SINDy-BVP: Sparse identification of nonlinear dynamics for boundary value problems,” *Physical Review Research*, vol. 3, no. 2, p. 023255, Jun. 2021.

[106] Y. Yang, M. Aziz Bhouri, and P. Perdikaris, “Bayesian differential programming for robust systems identification under uncertainty,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 476, no. 2243, p. 20200290, Nov. 2020. 3.1, 3.1

[107] S. H. Rudy, S. L. Brunton, and J. N. Kutz, “Smoothing and parameter estimation by soft-adherence to governing equations,” *Journal of Computational Physics*, vol. 398, p. 108860, Dec. 2019. 3.1

[108] Z. Zhang and Y. Liu, “A robust framework for identification of PDEs from noisy data,” *Journal of Computational Physics*, vol. 446, p. 110657, Dec. 2021. 3.1

[109] D. A. Messenger and D. M. Bortz, “Weak SINDy for partial differential equations,” *Journal of Computational Physics*, vol. 443, p. 110525, Oct. 2021. 3.1

[110] Q. He and J.-S. Chen, “A physics-constrained data-driven approach based on locally convex reconstruction for noisy database,” *Computer Methods in Applied Mechanics and Engineering*, vol. 363, p. 112791, May 2020.

[111] S. H. Rudy, J. Nathan Kutz, and S. L. Brunton, “Deep learning of dynamics and signal-noise decomposition with time-stepping constraints,” *Journal of Computational Physics*, vol. 396, pp. 483–506, Nov. 2019. 3.1, 5.2

[112] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, “Data-driven discovery of coordinates and governing equations,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22 445–22 451, 2019. 3.1

[113] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nature Communications*, vol. 9, no. 1, p. 4950, 2018.

[114] J. Koch, “Data-driven modeling of nonlinear traveling waves,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 4, p. 043128, Apr. 2021. 3.1

[115] N. M. Mangan, S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics,” *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol. 2, no. 1, pp. 52–63, 2016. 3.1

[116] K. Kaheman, J. N. Kutz, and S. L. Brunton, “SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 476, no. 2242, p. 20200279, 2020. 3.1

[117] K. P. Champion, S. L. Brunton, and J. N. Kutz, “Discovery of Nonlinear Multiscale Systems: Sampling Strategies and Embeddings,” *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 1, pp. 312–333, Jan. 2019. 3.1

[118] J. J. Bramburger, D. Dylewsky, and J. N. Kutz, “Sparse identification of slow timescale dynamics,” *Physical Review E*, vol. 102, no. 2, p. 022204, Aug. 2020. 3.1

[119] S. Li, E. Kaiser, S. Laima, H. Li, S. L. Brunton, and J. N. Kutz, “Discovering time-varying aerodynamics of a prototype bridge by sparse identification of nonlinear dynamical systems,” *Physical Review E*, vol. 100, no. 2, p. 022220, Aug. 2019. 3.1

[120] M. Gopalakrishnan Meena, A. G. Nair, and K. Taira, “Network community-based model reduction for vortical flows,” *Physical Review E*, vol. 97, no. 6, p. 063103, Jun. 2018. 3.1

[121] M. Hoffmann, C. Fröhner, and F. Noé, “Reactive SINDy: Discovering governing reactions from concentration data,” *The Journal of Chemical Physics*, vol. 150, no. 2, p. 025101, Jan. 2019. 3.1

[122] B. Bhadriraju, A. Narasingam, and J. S.-I. Kwon, “Machine learning-based adaptive model identification of systems: Application to a chemical process,” *Chemical Engineering Research and Design*, vol. 152, pp. 372–383, Dec. 2019.

[123] J. K. Johnson, S. Geng, M. W. Hoffman, H. Adesnik, and R. Wessel, “Precision multidimensional neural population code recovered from single intracellular recordings,” *Scientific Reports*, vol. 10, no. 1, p. 15997, Sep. 2020.

[124] A. Arzani and S. T. M. Dawson, “Data-driven cardiovascular flow modelling: examples and opportunities,” *Journal of The Royal Society Interface*, vol. 18, no. 175, p. 20200802, Feb. 2021. 3.1

[125] L. Boninsegna, F. Nüske, and C. Clementi, “Sparse learning of stochastic dynamical equations,” *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241723, Jun. 2018. 3.1, 8.2

[126] W. Zhao, G. Yin, and E.-W. Bai, “Sparse system identification for stochastic systems with general observation sequences,” *Automatica*, vol. 121, p. 109162, 2020.

[127] J. L. Callaham, J.-C. Loiseau, G. Rigas, and S. L. Brunton, “Nonlinear stochastic modelling with Langevin regression,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 477, no. 2250, p. 20210092, Jun. 2021. 3.1

[128] N. M. Mangan, T. Askham, S. L. Brunton, J. N. Kutz, and J. L. Proctor, “Model selection for hybrid dynamical systems via sparse regression,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 475, no. 2223, p. 20180534, Mar. 2019. 3.1

[129] Z. Long, Y. Lu, and B. Dong, “PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network,” *Journal of Computational Physics*, vol. 399, p. 108925, Dec. 2019. 3.1

[130] Y. Yuan, X. Tang, W. Zhou, W. Pan, X. Li, H.-T. Zhang, H. Ding, and J. Goncalves, “Data driven discovery of cyber physical systems,” *Nature Communications*, vol. 10, no. 1, p. 4894, Oct. 2019.

[131] R. Rai and C. K. Sahu, “Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus,” *IEEE Access*, vol. 8, pp. 71 050–71 073, 2020. 3.1

[132] C. Chen, N. N. Xiong, X. Guo, and J. Ren, “The System Identification and Prediction of the Social Earthquakes Burst in Human Society,” *IEEE Access*, vol. 8, pp. 103 848–103 859, 2020. 3.1

[133] S. Zhang and G. Lin, “Robust data-driven discovery of governing physical laws with error bars,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2217, p. 20180305, Sep. 2018. 3.1, 3.1

[134] D. R. Gurevich, P. A. K. Reinbold, and R. O. Grigoriev, "Robust and optimal sparse regression for nonlinear PDE models," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 10, p. 103113, Oct. 2019. 3.1

[135] P. A. K. Reinbold, D. R. Gurevich, and R. O. Grigoriev, "Using noisy or incomplete data to discover models of spatiotemporal dynamics," *Physical Review E*, vol. 101, no. 1, p. 010203, Jan. 2020. 3.1

[136] D. A. Messenger and D. M. Bortz, "Weak SINDy: Galerkin-Based Data-Driven Model Selection," *Multiscale Modeling & Simulation*, vol. 19, no. 3, pp. 1474–1497, Jan. 2021. 3.1

[137] P. Bühlmann and S. van de Geer, *Statistics for High-Dimensional Data: Methods, Theory and Applications.* Berlin, Heidelberg: Springer, 2011. 3.1, 6.1.2

[138] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, Feb. 2020. 3.1

[139] J. H. Lagergren, J. T. Nardini, G. Michael Lavigne, E. M. Rutter, and K. B. Flores, "Learning partial differential equations for biological transport models from noisy spatio-temporal data," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 476, no. 2234, p. 20190800, Feb. 2020.

[140] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, Jun. 2021. 3.1

[141] H. Xu, H. Chang, and D. Zhang, "DL-PDE: Deep-learning based data-driven discovery of partial differential equations from discrete and noisy data," *Communications in Computational Physics*, vol. 29, no. 3, pp. 698–728, Jun. 2021.

[142] D. Jia, X. Zhou, S. Li, S. Liu, and H. Shi, "Governing equation discovery based on causal graph for nonlinear dynamic systems," *Machine Learning: Science and Technology*, vol. 4, no. 4, p. 045008, Oct. 2023. 3.1, 6.2.2

[143] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," *Journal of Computational Physics*, vol. 395, pp. 620–635, Oct. 2019. 3.1

[144] K. T. Carlberg, A. Jameson, M. J. Kochenderfer, J. Morton, L. Peng, and F. D. Witherden, "Recovering missing CFD data for high-order discretizations using deep neural networks and dynamics learning," *Journal of Computational Physics*, vol. 395, pp. 105–124, Oct. 2019. 3.1

[145] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *Journal of Computational Physics*, vol. 375, pp. 1339–1364, Dec. 2018. 3.1

[146] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators,” *Nature Machine Intelligence*, vol. 3, no. 3, pp. 218–229, Mar. 2021.

[147] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, “DeepXDE: A Deep Learning Library for Solving Differential Equations,” *SIAM Review*, vol. 63, no. 1, pp. 208–228, Jan. 2021. 3.1

[148] H. Xu, D. Zhang, and N. Wang, “Deep-learning based discovery of partial differential equations in integral form from sparse and noisy data,” *Journal of Computational Physics*, vol. 445, p. 110592, Nov. 2021. 3.1

[149] J. Berg and K. Nyström, “Data-driven discovery of PDEs in complex datasets,” *Journal of Computational Physics*, vol. 384, pp. 239–252, May 2019. 3.1

[150] Y. Li, K. Wu, and J. Liu, “Discover governing differential equations from evolving systems,” *Physical Review Research*, vol. 5, no. 2, p. 023126, May 2023. 3.1

[151] D.-X. Zhou, “Universality of deep convolutional neural networks,” *Applied and Computational Harmonic Analysis*, vol. 48, no. 2, pp. 787–794, Mar. 2020. 3.1

[152] H. M. Romero Ugalde, J.-C. Carmona, J. Reyes-Reyes, V. M. Alvarado, and J. Mantilla, “Computational cost improvement of neural network models in black box nonlinear system identification,” *Neurocomputing*, vol. 166, pp. 96–108, Oct. 2015. 3.1

[153] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge: Cambridge University Press, 2019. 3.2.1, 3.2.1, 3.2.1

[154] L. Zhang and H. Schaeffer, “On the Convergence of the SINDy Algorithm,” *Multiscale Modeling & Simulation*, vol. 17, no. 3, pp. 948–972, Jan. 2019. 3.2.2

[155] U. Fasel, J. N. Kutz, B. W. Brunton, and S. L. Brunton, “Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 478, no. 2260, p. 20210904, Apr. 2022. 4, 4.1, 4.1.2, 4.4

[156] S. M. Hirsh, D. A. Barajas-Solano, and J. N. Kutz, “Sparsifying priors for Bayesian uncertainty quantification in model discovery,” *Royal Society Open Science*, vol. 9, no. 2, p. 211823, Feb. 2022. 4, 4.2, 4.2.1, 4.2.1, 4.4

[157] K. Kaheman, S. L. Brunton, and J. Nathan Kutz, “Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data,” *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015031, Mar. 2022. 4, 4.3, 4.3, 4.3.1, 4.3.1, 1, 2, 4.3.1, 4.4

[158] C. B. Delahunt and J. N. Kutz, “A Toolkit for Data-Driven Discovery of Governing Equations in High-Noise Regimes,” *IEEE Access*, vol. 10, pp. 31 210–31 234, 2022. 4

[159] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996. 4.1

[160] P. Bühlmann, “Bagging, Boosting and Ensemble Methods,” in *Handbook of Computational Statistics: Concepts and Methods*, J. E. Gentle, W. K. Härdle, and Y. Mori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 985–1022. 4.1

[161] K. P. Murphy, *Probabilistic machine learning: an introduction*, ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2022. 4.3

[162] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic Differentiation in Machine Learning: a Survey,” *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018. 5

[163] K. L. Ratzlaff and J. T. Johnson, “Computation of two-dimensional polynomial least-squares convolution smoothing integers,” *Analytical Chemistry*, vol. 61, no. 11, pp. 1303–1305, Jun. 1989. 5.1

[164] R. C. Gonzalez and R. E. Woods, “Smoothing (Lowpass) Spatial Filters,” in *Digital image processing.* New York: Pearson, 2018, pp. 164–175. 5.2

[165] R. Modonesi, M. Dalai, P. Migliorati, and R. Leonardi, “A Note on Probabilistic and Geometric Shaping for the AWGN Channel,” *IEEE Communications Letters*, vol. 24, no. 10, pp. 2119–2122, Oct. 2020. 5.2

[166] A. Goldsmith, *Wireless Communications.* Cambridge: Cambridge University Press, 2005. 5.2

[167] E. N. Lorenz, “Deterministic Nonperiodic Flow,” *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, Mar. 1963. 5.2.1

[168] M. C. Cross and P. C. Hohenberg, “Pattern formation outside of equilibrium,” *Reviews of Modern Physics*, vol. 65, no. 3, pp. 851–1112, Jul. 1993. 5.2.2, 6.2.2

[169] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, 2nd ed. Cambridge: Cambridge University Press, 2022. 5.2.2, 6.2.2

[170] K. Taira and T. Colonius, “The immersed boundary method: A projection approach,” *Journal of Computational Physics*, vol. 225, no. 2, pp. 2118–2137, 2007. 6.2.2

[171] T. Colonius and K. Taira, “A fast immersed boundary method using a null-space approach and multi-domain far-field boundary conditions,” *Immersed Boundary Method and Its Extensions*, vol. 197, no. 25, pp. 2131–2146, 2008. 6.2.2

[172] Andrei D. Polyanin and Valentin F. Zaitsev, *Handbook of Nonlinear Partial Differential Equations*, 2nd ed. New York: Chapman and Hall/CRC, 2012. 6.2.2

[173] M. Gavish and D. L. Donoho, "The Optimal Hard Threshold for Singular Values is $4/\sqrt{3}$," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 5040–5053, 2014. 6.2.2

[174] Ralf W. Wittenberg, "Kuramoto-Sivashinsky equation - Encyclopedia of Mathematics," 2020. 6.2.2

[175] B. Chen, K. Huang, S. Raghupathi, I. Chandratreya, Q. Du, and H. Lipson, "Automated discovery of fundamental variables hidden in experimental data," *Nature Computational Science*, vol. 2, no. 7, pp. 433–442, Jul. 2022. 6.3

[176] O. N. Bjørnstad, K. Shea, M. Krzywinski, and N. Altman, "The SEIRS model for infectious disease dynamics," *Nature Methods*, vol. 17, no. 6, pp. 557–558, Jun. 2020. 7

[177] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and Stephan Munch, "Detecting causality in complex ecosystems," *Science*, vol. 338, no. 6106, pp. 496–500, 2012. 7

[178] M. S. Hong, K. A. Severson, M. Jiang, A. E. Lu, J. C. Love, and R. D. Braatz, "Challenges and opportunities in biopharmaceutical manufacturing control," *Computers & Chemical Engineering*, vol. 110, pp. 106–114, Feb. 2018. 7

[179] M. J. Beira and P. J. Sebastião, "A differential equations model-fitting analysis of COVID-19 epidemiological data to explain multi-wave dynamics," *Scientific Reports*, vol. 11, no. 1, p. 16312, Aug. 2021. 7, 7.1.3

[180] V. Martínez, "A Modified SIRD Model to Study the Evolution of the COVID-19 Pandemic in Spain," *Symmetry*, vol. 13, no. 4, 2021. 7.1.3

[181] K. S. Nisar, S. Ahmad, A. Ullah, K. Shah, H. Alrabaiah, and M. Arfan, "Mathematical analysis of SIRD model of COVID-19 with Caputo fractional derivative based on real data," *Results in Physics*, vol. 21, p. 103772, Feb. 2021. 7

[182] K. Egan, "Automatic Extraction of Ordinary Differential Equations from Data: Sparse Regression Tools for System Identification," Doctoral, Durham University, 2023. 7, 8.1

[183] W. Li and R. Carvalho, "Automating the discovery of partial differential equations in dynamical systems," *Machine Learning: Science and Technology*, vol. 5, no. 3, p. 035046, Aug. 2024. 7

[184] J. Wei, N. Stoesser, P. C. Matthews, T. Khera, O. Gethings, I. Diamond, R. Studley, N. Taylor, T. E. A. Peto, A. S. Walker, K. B. Pouwels, D. W. Eyre, E. Rourke, T. Thomas, D. Pienaar, J. Preece, S. Crofts, L. Lloyd, M. Bowen,

D. Ayoubkhani, R. Black, A. Felton, M. Crees, J. Jones, E. Sutherland, D. W. Crook, E. Pritchard, K.-D. Vihta, A. Howarth, B. D. Marsden, K. K. Chau, L. M. Ferreira, W. Dejnirattisai, J. Mongkolsapaya, S. Hoosdally, R. Cornall, D. I. Stuart, G. Screaton, K. Lythgoe, D. Bonsall, T. Golubchik, H. Fryer, J. N. Newton, J. I. Bell, S. Cox, K. Paddon, T. James, T. House, J. Robotham, P. Birrell, H. Jordan, T. Sheppard, G. Athey, D. Moody, L. Curry, P. Brereton, I. Jarvis, A. Godsmark, G. Morris, B. Mallick, P. Eeles, J. Hay, H. VanSteenhouse, J. Lee, S. White, T. Evans, L. Bloemberg, K. Allison, A. Pandya, S. Davis, D. I. Conway, M. MacLeod, C. Cunningham, and the COVID-19 Infection Survey team, “Risk of SARS-CoV-2 reinfection during multiple Omicron variant waves in the UK general population,” *Nature Communications*, vol. 15, no. 1, p. 1008, Feb. 2024. 7.1.3

[185] J. Fernández-Villaverde and C. I. Jones, “Estimating and simulating a SIRD Model of COVID-19 for many countries, states, and cities,” *Journal of Economic Dynamics and Control*, vol. 140, p. 104318, Jul. 2022. 7.1.3

[186] A. A. Haghrah, S. Ghaemi, and M. A. Badamchizadeh, “Fuzzy-SIRD model: Forecasting COVID-19 death tolls considering governments intervention,” *Artificial Intelligence in Medicine*, vol. 134, p. 102422, Dec. 2022.

[187] S. Shringi, H. Sharma, P. Narayan Rathie, J. Chand Bansal, A. Nagar, and D. Lal Suthar, “Predicting COVID-19 outbreak in India using modified SIRD model,” *Applied Mathematics in Science and Engineering*, vol. 32, no. 1, p. 2305191, Dec. 2024.

[188] C. Çakmaklı and Y. Şimşek, “Bridging the Covid-19 data and the epidemiological model using the time-varying parameter SIRD model,” *Journal of Econometrics*, vol. 242, no. 1, p. 105787, May 2024. 7.1.3

[189] A. Bousquet, W. H. Conrad, S. O. Sadat, N. Vardanyan, and Y. Hong, “Deep learning forecasting using time-varying parameters of the SIRD model for Covid-19,” *Scientific Reports*, vol. 12, no. 1, p. 3030, Feb. 2022. 7.1.3

[190] G. M. Athayde and A. P. Alencar, “Forecasting Covid-19 in the United Kingdom: A dynamic SIRD model,” *PLOS ONE*, vol. 17, no. 8, p. e0271577, Aug. 2022. 7.1.3

[191] V. E. Papageorgiou and G. Tsaklidis, “A stochastic SIRD model with imperfect immunity for the evaluation of epidemics,” *Applied Mathematical Modelling*, vol. 124, pp. 768–790, Dec. 2023. 7.1.3

[192] R. C. Smith, *Uncertainty quantification: Theory, implementation, and applications.* Philadelphia: Society for Industrial and Applied Mathematics, 2013. 8.2

[193] R. Berk, L. Brown, A. Buja, Kai Zhang, and L. Zhao, “Valid post-selection inference,” *The Annals of Statistics*, vol. 41, no. 2, pp. 802–837, Apr. 2013. 8.2

[194] J. Taylor and R. J. Tibshirani, “Statistical learning and selective inference,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 25, pp. 7629–7634, Jun. 2015. 8.2

[195] Y. Wang, H. Fang, J. Jin, G. Ma, X. He, X. Dai, Z. Yue, C. Cheng, H.-T. Zhang, D. Pu, D. Wu, Y. Yuan, J. Gonçalves, J. Kurths, and H. Ding, “Data-Driven Discovery of Stochastic Differential Equations,” *Engineering*, vol. 17, pp. 244–252, Oct. 2022. 8.2

[196] Y. Huang, Y. Mabrouk, G. Gompper, and B. Sabass, “Sparse inference and active learning of stochastic differential equations from data,” *Scientific Reports*, vol. 12, no. 1, p. 21691, Dec. 2022.

[197] Y. C. Mathpati, T. Tripura, R. Nayek, and S. Chakraborty, “Discovering stochastic partial differential equations from limited data using variational Bayes inference,” *Computer Methods in Applied Mechanics and Engineering*, vol. 418, p. 116512, Jan. 2024. 8.2

[198] F. Dietrich, A. Makeev, G. Kevrekidis, N. Evangelou, T. Bertalan, S. Reich, and I. G. Kevrekidis, “Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 2, p. 023121, Feb. 2023. 8.2