Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Mathematics and Operations Research

# Identification of partial differential equation description of nonlinear system from noisy data

**Otso Väärämäki**

Master's Thesis
2026

Aalto-yliopisto
Aalto-universitetet
Aalto University

**Author** Otso Väärämäki

**Title** Identification of partial differential equation description of nonlinear system from noisy data

**Degree programme** Mathematics and Operations Research

**Major** Systems and Operations Research

**Thesis supervisor and advisor** Dr. Riku Linna

**Date** May 15, 2026 **Number of pages** 49 **Language** English

**Abstract**

This work examines computational algorithms that can be used to identify partial differential equations directly from observational data. The main goal of the work is to compare the ability of the algorithms to tolerate measurement noise and to find the correct underlying equation structure.

The study compares two main classes of algorithms: strong-form methods based on direct numerical differentiation (RIMSI and PDE-FIND) and weak-form methods that utilise integration by parts (WSINDy and E-WSINDy). The performance of the algorithms is tested against data produced by the chaotic Kuramoto-Sivashinsky equation, and adding different magnitudes of Gaussian noise to it. In addition, it is investigated how data preprocessing with Savitzky-Golay filtering affects the success of the equation discovery.

The results show that the strong-form algorithms are very sensitive to noise since their ability to find the correct equation structure vanishes even in the presence of relatively small noise. The weak-form methods, on the other hand, handle the task much more reliably. They manage to extract the correct terms even when the magnitude of noise is very high. In the case of large candidate libraries, E-WSINDy is able to eliminate incorrect model terms more effectively than the basic WSINDy algorithm.

Savitzky-Golay filtering is found to be useful mainly in the presence of strong noise. For lower noise levels, it was found that filtering could even reduce the accuracy of the weak-form methods. The study, moreover, reveals that the dynamics pertinent to the data must be sufficiently complex to allow the algorithms to identify the underlying equation. In the case of a simple travelling wave, the algorithms end up identifying a reduced model describing the wave motion instead of the true equation. The results obtained strongly support the use of weak-form methods for identifying partial differential equation description of complex, noisy dynamical systems from measurement data.

**Keywords** partial differential equations, dynamical system, system identification, noise, robustness to noise, inference

Aalto-yliopisto
Aalto-universitetet
Aalto University

**Tekijä** Otso Väärämäki

**Työn nimi** Epälineaaristen systeemien osittaisdifferentiaaliyhtälön kuvauksen tunnistaminen kohinaisesta datasta

**Koulutusohjelma** Matematiikka ja operaatiotutkimus

**Pääaine** Systeemi- ja operaatiotutkimus

**Valvoja ja ohjaaja** Dr. Riku Linna

**Päivämäärä** 15. toukokuuta 2026 **Sivumäärä** 49 **Kieli** englanti

**Tiivistelmä**

Tässä diplomityössä tarkastellaan laskennallisia algoritmeja, joita voidaan käyttää identifioimaan osittaisdifferentiaaliyhtälö havaitusta datasta. Työn päätavoite on vertailla algoritmien kykyä sietää mittauskohinaa ja löytää oikea datan tuottavan yhtälön muoto.

Työssä vertaillaan kahta algoritmien pääluokkaa: vahvan muodon menetelmiä, jotka perustuvat suoraan numeeriseen derivointiin (RIMSI ja PDE-FIND) ja heikon muodon menetelmiä, jotka hyödyntävät osittaisintegrointia (WSINDy ja E-WSINDy). Algoritmien suorituskykyä testataan käyttäen Kuramoto-Sivashinsky-yhtälön numeerisesta integroinnista saatua dataa, johon on lisätty erisuuruista gaussista kohinaa. Tarkastelemme lisäksi, kuinka datan esikäsittely Savitzky-Golay-suodattimella vaikuttaa algoritmien kykyyn löytää yhtälö.

Tulokset osoittavat, että vahvan muodon algoritmit ovat todella herkkiä kohinalle, sillä niiden kyky löytää oikea yhtälömuoto häviää jo suhteellisen pienen kohinan tapauksessa. Heikon muodon menetelmät sen sijaan suoriutuvat tehtävästä paljon luotettavammin. Ne pystyvät löytämään oikeat yhtälötermit jopa silloin, kun kohinan suuruusluokka on suuri. Suurien kandidaattifunktiokirjastojen tapauksessa E-WSINDy kykenee karsimaan mallista vääriä termejä tehokkaammin kuin tavallinen WSINDy.

Tulosten mukaan Savitzky-Golay-suodatin on pääosin hyödyllinen vain erittäin suuren kohinan tapauksessa. Pienemmillä kohinatasoilla suodatus saattaa jopa alentaa heikon muodon menetelmien tarkkuutta. Tulokset lisäksi paljastavat, että datan tuottavan dynamiikan täytyy olla tarpeeksi kompleksista, jotta algoritmit pystyvät identifioimaan oikean yhtälön. Jos kyseessä on yksinkertainen liikkuva aalto, algoritmit päätyvät oikean yhtälön sijaan identifioimaan pelkistetyn mallin, joka kuvaa aaltoliikettä. Tuloksemme tukevat selvästi heikon muodon menetelmien käyttöä monimutkaisten, kohinaisten dynaamisten systeemien oikean kuvauksen löytämisessä osittaisdifferentiaaliyhtälöiden avulla.

**Avainsanat** osittaisdifferentiaaliyhtälöt, dynaaminen systeemi, systeemien identifiointi, kohina, kohinansietokyky, inferenssi

## Preface

I am grateful to my supervisor and advisor, Dr. Riku Linna, for providing helpful guidance and pertinent advice during the entire time I spent working on this thesis. I also want to thank my colleagues, Iiro, who was always willing to help, and Marius, with whom I had many delightful conversations.

Otaniemi, 15th May 2026

Otso Väärämäki

# Contents

# Symbols and abbreviations

## Symbols and operators

| | |
|---|---|
| $\mathbf{U}$ | time-series matrix of state variables |
| $\mathbf{u}_t$ | time derivative vector |
| $\mathbf{\Theta}$ | matrix of candidate functions |
| $\mathbf{w}$ | coefficient vector |
| $\|\cdot\|_n$ | vector $n$-th norm |
| $\mathbf{X}_i$ | the $i$th column vector of matrix $\mathbf{X}$ |
| $U_{i,j}$ | entry in the $i$th row and $j$th column of matrix $\mathbf{U}$ |
| $\mathcal{N}(a,b)$ | normal distribution with mean $a$ and variance $b$ |
| $\mathcal{U}(a,b)$ | uniform distribution from $a$ to $b$ |
| $u_t$ | time derivative |
| $u_x, u^{(n)}$ | spatial derivative – the number of $x$ subscripts or the superscript $n$ denotes the order of the derivative |

## Abbreviations

| | |
|---|---|
| RIMSI | Robust Interpretable Model for System Identification (algorithm) |
| SINDy | Sparse Identification of Non-linear Dynamics (algorithm) |
| WSINDy | Weak SINDy (algorithm) |
| E-WSINDy | Ensemble Weak SINDy (algorithm) |
| SG filtering | Savitzky-Golay filtering |
| TPR | true positive rate |
| wTPR | weighted true positive rate |
| FPR | false positive rate |
| wFPR | weighted false positive rate |
| ASTP error | average short-time prediction error |
| ODE | ordinary differential equation |
| PDE | partial differential equation |
| KS equation | Kuramoto-Sivashinsky equation |

# 1 Introduction

Differential equations can be used to describe many fundamental phenomena of physics and model the dynamics of complex systems. They form the basis for the study in various fields such as population dynamics [1], fluid dynamics [2], material microstructures [3], and biological pattern formation [4]. Additionally, differential equations are increasingly being used in neuroscience [5] and mathematical biology [6], for instance, in describing the spatial and temporal dynamics of neural networks.

Today, due to the rapid development of measuring instruments and computing power, we are ever more moving towards data-driven modelling, in which the equations governing the system under observation can be learned directly from collected data [7–9]. This change has brought many phenomena within the reach of numerical modelling for which the construction of accurate analytical models was previously practically impossible. Equation discovery methods are now used in a diverse range of fields. They are used to study, for example, turbulent flows and aerodynamics [10, 11], ocean dynamics and climate change [12], and plasma physics [13]. The methods are moreover utilised in nonlinear optics [14] and solid mechanics [15]. In addition to physics and technology, these tools may benefit the research of biological systems, and they are now applied to the modelling of infectious disease spreading [16] and dynamics of regulatory biological networks [17].

In the field of data-driven modelling, the human brain offers an interesting and, at the same time, highly challenging example of such a complex and constantly changing system. When brain activity is measured non-invasively, for example, with electroencephalography (EEG) or magnetoencephalography (MEG), from a mathematical modelling perspective, there are two distinct approaches for analysing the multidimensional data. Sensors placed on the surface of the head register wave-like electromagnetic fields, or topographies, that vary in time and space. Since these head surface fields are continuous, the most fitting mathematical tool for describing their spatio-temporal dynamics is a partial differential equation [18–20]. When these same signals measured from the surface of the head are inversely mapped back to the cortex, a complex network of numerous interacting dipole sources is revealed. While this cortical-level activation has traditionally been described by networks of interconnected ordinary differential equations (ODEs) [21], recent approaches emphasise the spatiotemporal and wave-like nature of these dynamics [22].

This thesis was carried out in parallel and in close collaboration with another thesis conducted at Aalto University [23]. The theses focus on qualitatively different dynamical models, which, for instance, reflect the different approaches to the modelling of brain activity. Whereas the parallel work focused on the identification of ODE systems, this thesis focuses on the identification of partial differential equation (PDE) systems. The aim of the work is to investigate computational algorithms that are able to infer the PDE underlying continuous spatio-temporal data, such as changing EEG topographies or turbulent fluid flows, directly from observational data.

In recent years, the field of data-driven equation discovery has been advanced in particular by sparse regression-based SINDy methods [24, 25]. Since the equations governing even very complex spatial dynamics usually consist of only a few key terms, the objective of the methods is to select a sparse subset from a large library of candidate functions and derivatives that explains the evolution of the data [26]. Although the method has been proven to work under theoretical conditions, real measurement data always contains random noise. This poses a significant challenge for traditional strong-form methods, such as the PDE-FIND algorithm [27]. While the original PDE-FIND framework tries to manage this problem by smoothing the data, for example, using local polynomial interpolation, its basic pointwise numerical differentiation in constructing the candidate derivatives is still a critical bottleneck. Numerical differentiation behaves mathematically like a high-pass filter that amplifies high-frequency errors in the signal. Consequently, this makes the estimation of higher spatial derivatives unstable and even impossible with weaker signal-to-noise ratios.

To solve this fundamental problem, attention has turned to weak-form approaches, which constitute possibly the most fundamental recent advance in the field [28]. In the weak-form based methods, the difficult numerical derivation is bypassed by transferring the derivative operators from the raw data to analytically defined and smooth test functions by means of integration by parts. The integration operation itself acts as a natural low-pass filter, which makes the weak-form WSINDy algorithm [29] significantly more robust than traditional methods that rely on derivation. This concept can be further combined with ensemble learning, resulting in the E-WSINDy method [30].

This work critically and systematically compares the performance of the weak-form methods with respect to strong-form algorithms, especially the recently developed RIMSI algorithm at Aalto University [31]. The data for the comparison is produced by the complex and highly chaotic Kuramoto-Sivashinsky (KS) equation. As it contains spatial derivatives up to the fourth order, it provides a demanding benchmark for measuring the noise tolerance and system identification accuracy of the algorithms.

By exposing the data to different magnitudes of Gaussian noise, we show that strong-form methods based on numerical differentiation (RIMSI and PDE-FIND) perform poorly. Their ability to identify the correct equation structure collapses even at quite low noise levels (about $\sigma_{\text{obs}} \approx 10^{-4}$). The weak-form methods, WSINDy and E-WSINDy, are significantly more robust than strong-form algorithms, and the exact equation structure can be found even in the presence of significantly stronger noise ($\sigma_{\text{obs}} \approx 10^{-1}$). We also find that when dealing with very large candidate libraries, E-WSINDy eliminates erroneous terms more effectively than WSINDy. Our results also show that separate pre-filtering of the data using the Savitzky-Golay (SG) filtering saves the model from collapse in extreme noise conditions, although in milder noise, filtering may even weaken the equation discovery accuracy. Finally, our experiments in the dynamically simple travelling wave state show that sufficient dynamic complexity of the data is a prerequisite for revealing the actual dynamics. If the dynamics are too simple, the sparseness-favouring algorithms end up with reduced models describing purely wave motion instead of the actual PDE.

The work is structured as follows. Section 2 presents the theoretical background and an overview of modelling spatiotemporal systems. In Section 3, the research methods are outlined, along with the metrics with which the algorithms' performance is evaluated. The results of this study are presented in Section 4. Finally, Section 5 summarises the results and provides conclusions based on them.

# 2 Theoretical background

## 2.1 Nonlinear spatiotemporal systems

Dynamical systems provide a framework for understanding diverse physical phenomena, many of which are governed by PDEs. In this thesis, we consider a PDE for the variable $u(x,t)$ of the form:

$$u_t = N(u, u_x, u_{xx}, \ldots, \mu), \tag{1}$$

where $u(x,t)$ represents the state of the system as a function of location $x$ and time $t$, $u_t$ is the partial derivative of the state with respect to time, and $N$ is a nonlinear operator that depends on the state variable, its spatial derivatives, and possible parameters $\mu$. The goal of system identification is to find the structural form and coefficients of the operator $N$ when the state evolution $u(x,t)$ is known from observations.

The behaviour of the system is governed by the form of the operator $N$. In linear systems, the superposition principle applies, and even complex spatial states can be constructed by combining the fundamental solutions. However, once nonlinear terms, such as the products of the state variable with its spatial derivatives, are introduced, the dynamics change fundamentally. It is this non-linearity that gives rise to the complex phenomena we observe in nature and is the mechanism behind spontaneous pattern formation and deterministic chaos [32].

Because of this underlying complexity, solving the inverse problem is extremely challenging, as we must identify the exact underlying mathematical structure based on raw observations. While ODEs track the time evolution of a specific, limited number of discrete variables, PDEs model continuous fields that vary in space and time [33]. Since it is physically impossible to measure a system everywhere continuously, we are forced to infer these continuous dynamics from a limited grid of discrete, noisy measurements. For a data-driven identification method to be successful under these conditions, it must be capable of extracting the correct combination of nonlinear and differential operators from a vast search space of possible equations.

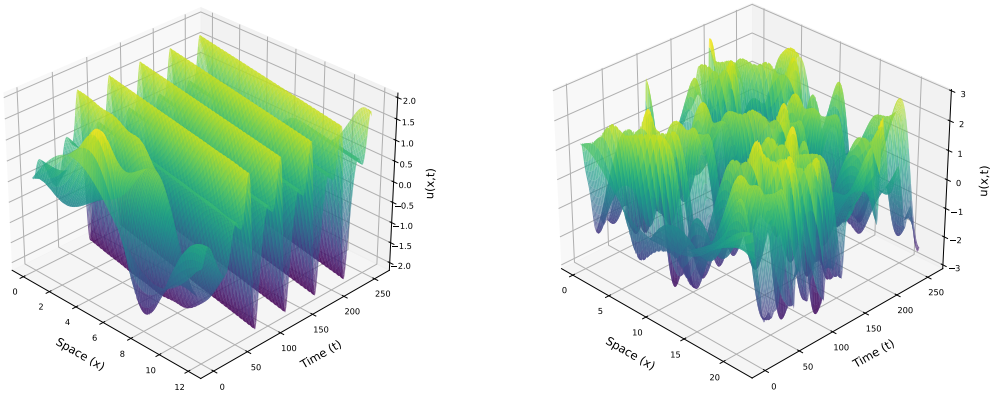
### 2.1.1 Kuramoto-Sivashinsky equation

The focus of this thesis is on the Kuramoto-Sivashinsky equation, which was originally developed for modelling instabilities at flame fronts [34–36]. Its application has since expanded considerably from its original roots. The equation has been successfully applied, for example, to the dynamics of thin liquid films flowing along inclined surfaces and to the study of hydrodynamic turbulence [37, 38]. In this work, we consider the one-dimensional KS equation, which is defined as

$$u_t = -uu_x - u_{xx} - u_{xxxx}. \tag{2}$$

In this form, the unique characteristics of the different terms in the equation are easy to distinguish. The second-order derivative $-u_{xx}$ acts as a reverse diffusion, which causes instability in the system at large scales, while the fourth-order term $-u_{xxxx}$ damps this motion at small scales. The nonlinear advection term $-uu_x$, in turn, transfers energy between these two extremes, which prevents the system from growing uncontrollably.

One of the most intriguing features of the KS equation is the dependence of its dynamics on the domain size under consideration, $L$, which can be considered as the bifurcation parameter of the system. When the domain size is increased, the behaviour of the system changes fundamentally. As illustrated in the 3D plots of Figure 1, for small values, for example $L = 12$, the system produces a dynamically simple state, such as a wave travelling with constant speed. When the value is increased, the KS PDE begins to exhibit chaotic behaviour. Eventually, significantly larger values (for example, $L \approx 100$) would result in very complex spatiotemporal turbulence [39]. The values $L = 12$, $L = 22$, and $L = 36$ used in the experiments of this work represent the stationary and chaotic regimes: the first describes regular wave motion, and the latter two a clearly more chaotic state.




**(a)** Data produced by the KS equation for spatial domain $L = 12$.

**(b)** Data produced by the KS equation for spatial domain $L = 22$.

**Figure 1:** Travelling wave and chaotic regimes of the KS equation with transient periods included.

To quantify the degree of a system's chaoticity, Lyapunov exponents can be utilised [40]. Formally, they measure the exponential rate of divergence of trajectories in the infinite time limit. If two initially very close trajectories of a dynamical system are separated by a small perturbation $\delta u(0)$, this difference grows with time approximately according to the formula $\|\delta u(t)\| \approx e^{\lambda t}\|\delta u(0)\|$. The largest Lyapunov exponent $\lambda$ can thus be expressed as the limit

$$\lambda = \lim_{t\to\infty} \frac{1}{t} \ln \frac{\|\delta u(t)\|}{\|\delta u(0)\|}. \tag{3}$$

Each individual Lyapunov exponent $\lambda_i$ corresponds to the divergence of trajectories in a different orthogonal direction. For a system to be classified as chaotic based on its initial value sensitivity, it must have at least one positive Lyapunov exponent. For the KS equation, the number of these positive exponents generally increases with the region size. We can observe this transition directly through the values of the largest Lyapunov exponents. For a small domain size of $L = 12$, the largest exponent is near zero at 0.003, which aligns with the stable travelling wave state. As the domain size increases to $L = 22$ and $L = 36$, the largest exponent grows to 0.043 and 0.080, respectively [39], which confirms the shift into chaotic dynamics.

## 2.2 Discovering nonlinear dynamics from observations

In the field of data-driven modelling, there are two main branches. First, in system prediction, the goal for a model is to be able to estimate the future states of the system based on the given data. The second branch and the focus of this thesis is system identification, where the aim of an algorithm is not just to imitate observations, but to find an accurate mathematical model, such as a partial differential equation, that underlies the data. We therefore want to reveal the governing laws of the system.

As Rudy et al. [27] point out, the derivation of equations governing complex physical phenomena has traditionally relied heavily on conservation laws and fundamental principles. However, the growth of modern, high-resolution measurement data has allowed a fundamental shift towards data-driven modelling. Earlier breakthroughs in this inference process utilised evolutionary algorithms, such as *symbolic regression*, to deduce free-form laws of nature directly from experimental data [41, 42]. Symbolic regression utilises principles of genetic programming [43] by building mathematical expressions from random building blocks (such as mathematical operators, variables, and constants) and combining them iteratively using genetic algorithms. The expressions that best explain the data remain in the next generation, which gradually allows the models to evolve towards the correct dynamic description. Even though symbolic regression is an extremely flexible approach, it scales poorly

when the dimensions of the system increase and it often becomes computationally heavy for complex spatiotemporal dynamics.

Recently, deep learning architectures have changed the field of data-driven modelling. For example, *Physics-informed neural networks* (PINNs) embed known physical laws directly into the loss functions of neural networks. In simplified terms, if the behaviour of the system is assumed to be governed by a partial differential equation, the neural network is trained to minimise the combined loss function $\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{PDE}}$. The first part of the equation, $\mathcal{L}_{\text{data}}$, measures the deviation of the prediction from the observed data and the second term, $\mathcal{L}_{\text{PDE}}$, penalises the network if its derivatives, which are calculated using automatic differentiation, violate the assumed governing equation. This allows them to approximate solutions and identify system parameters with excellent accuracy [44]. However, the challenge with traditional PINN models is that they require the mathematical structure of the governing equation to be known in advance. If the ultimate goal is to identify the system (that is, to discover the exact physics that governs the dynamics), we need methods that produce clear and mathematically exact expressions instead of opaque black-box approximations [28].

The demand for interpretability has led to an increased interest in sparse regression-based techniques. The motivating assumption behind them is that even if a dynamical system behaves very chaotically, the differential equation governing this behaviour typically consists of only a few terms. By constructing a large and diverse library of possible candidate functions, the challenge of discovering the dynamics of a system can be transformed into a sparse optimisation problem [24]. The task of the algorithm is to find a sparse combination of candidate functions that can accurately reconstruct the time derivative of the observed data.

A well-known framework in this field, *Sparse identification of nonlinear dynamics* (SINDy), which was originally developed for ODEs [24], introduces new complexities when it is utilised for PDE discovery. Because PDEs define the evolution of continuous fields in both time and space, the candidate library must contain spatial derivatives. For traditional strong-form methods, the identification algorithm must be able to estimate these spatial derivatives directly from discrete data grids [27]. Under optimal, noise-free conditions, computing the derivatives is relatively straightforward. However, when observations contain measurement noise, standard difference methods act like high-pass filters, which strongly amplify high-frequency errors [45, 46]. Therefore, finding the true nonlinear dynamics from raw and noisy observational data is not a simple regression problem. It requires robust strategies that can mitigate or, in the best case, avoid the noise amplification that is inherent in pointwise spatial derivation.

This is where weak-form methods such as *Weak SINDy* (WSINDy) [29] come into play. Instead of trying to compute numerical derivatives directly from noisy data, they transfer the differentiation operation to smooth and analytically known test functions via integration by parts. The promise of WSINDy is then to make the PDE discovery significantly more robust to noise.

# 3 Research methods

## 3.1 Spatiotemporal datasets

To systematically evaluate inference methods, the ground-truth datasets in this study are generated by numerical simulations of the Kuramoto-Sivashinsky equation, which is integrated over a defined spatiotemporal grid. Hence, a discrete state matrix $\mathbf{U} \in \mathbb{R}^{N_x \times N_t}$ is produced:

$$\mathbf{U} = \begin{bmatrix} u(x_1, t_1) & u(x_1, t_2) & \dots & u(x_1, t_{N_t}) \\ u(x_2, t_1) & u(x_2, t_2) & \dots & u(x_2, t_{N_t}) \\ \vdots & \vdots & \ddots & \vdots \\ u(x_{N_x}, t_1) & u(x_{N_x}, t_2) & \dots & u(x_{N_x}, t_{N_t}) \end{bmatrix}, \tag{4}$$

where $N_x$ and $N_t$ represent the number of spatial grid points and time points, respectively.

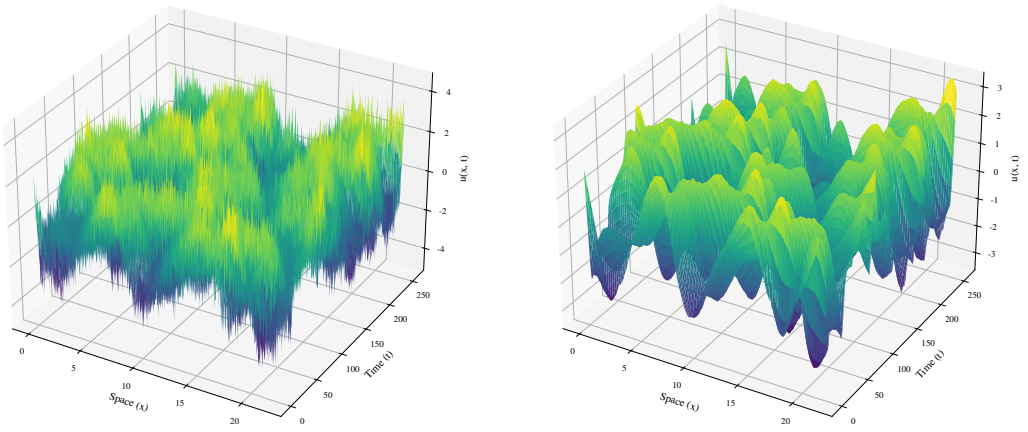
For the system identification task, we assume that the underlying physical laws and governing equations are hidden. Hence, only discrete spatiotemporal measurements are available for our analysis. Furthermore, to assess the performance of the inference methods under non-ideal conditions, which is the norm in the real-world, the clean state matrices, denoted by $\mathbf{U}_{\text{true}}$, are artificially corrupted with additive noise. The resulting observed data set, $\mathbf{U}_{\text{noisy}}$, is generated as follows:

$$\mathbf{U}_{\text{noisy}} = \mathbf{U}_{\text{true}} + \sigma_{\text{obs}}\sigma_{\text{data}}\mathbf{N} = \mathbf{U}_{\text{true}} + \sigma_{\text{noise}}\mathbf{N}, \tag{5}$$

where $\sigma_{\text{obs}} \geq 0$ defines the relative magnitude of the observational noise. The random noise matrix $\mathbf{N}$ consists of independent, identically distributed samples drawn from a standard normal distribution, $\mathcal{N}(0, 1)$, which ensures the noise remains uncorrelated across both space and time. The addition of noise whose magnitude depends on the standard deviation of the data $\sigma_{\text{data}}$ is consistent with the literature [27], and this allows us to reasonably compare the inference methods between different systems. Specifically, this approach maintains a consistent signal-to-noise ratio. Otherwise, adding a fixed absolute noise level would seem to produce worse results for datasets with smaller variances.

## 3.2 Data denoising

Before feeding noisy data into inference methods, data denoising techniques can be used to mitigate the effects of noise. In this study, an adaptive two-dimensional Savitzky-Golay filter is used, as introduced in [47]. The SG filter is chosen as it is shown to have the best overall performance in the case of ODEs [23]. The SG filter smooths the data by fitting a low-degree polynomial to successive subsets of adjacent data points. In this implementation, different values of the degree $p$ of the polynomial were tested, and $p = 3$ was chosen as the best. Figure 2 demonstrates the effect of the filter for noisy data.




**(a)** Data before SG denoising.

**(b)** Data after SG denoising.

**Figure 2:** The effect of SG denoising for data from the KS equation for spatial domain $L = 22$ with 50% noise.

Given the noisy data matrix $\mathbf{U}_{\text{noisy}}$, which is assumed by the SG denoiser to have transposed dimensions, $N_t \times N_x$, the filtering is achieved sequentially. The filter takes candidate window size $w$ as an input, which determines the length of the interval to which a polynomial is fitted. For a specific $w$, the filter is first applied along the spatial dimension and subsequently along the temporal dimension to produce the smoothed estimate, which we denote by $\hat{\mathbf{U}}(w)$.

The algorithm dynamically selects the optimal window $w$ from a set of odd-integer candidates to have a symmetrical window around a point by minimising the Akaike information criterion (AIC) [48]. The candidate window sizes are logarithmically spaced from $w = 5$ up to a maximum threshold equal to the minimum of one-fifth of the temporal or spatial dimension.

For each candidate window size $w$, the residual sum of squares (RSS) is computed as

$$\text{RSS}(w) = \sum_{i=1}^{N_t} \sum_{j=1}^{N_x} \left(U_{i,j} - \hat{U}_{i,j}(w)\right)^2. \tag{6}$$

The effective degrees of freedom for this sequential 2D filter are heuristically estimated as

$$k_{\text{eff}}(w) = \frac{N_t \cdot N_x}{w^2}(p+1)^2. \tag{7}$$

The AIC for a given window size is then evaluated using the following formulation

$$\text{AIC}(w) = (N_t \cdot N_x) \ln\left(\frac{\text{RSS}(w)}{N_t \cdot N_x}\right) + 2k_{\text{eff}}(w). \tag{8}$$

The window size $w^*$ that yields the lowest AIC score is selected as the optimal hyperparameter. The final denoised dataset is generated by applying the SG filter with this optimal window to both data dimensions. The complete SG filter pseudo-code is described in Algorithm 1.

## 3.3 Candidate function library

The core premise of sparse model discovery is that the target dynamics can be formulated as a sparse linear combination of candidate functions selected from a predefined collection [27]. However, depending on whether the method employs pointwise differentiation or a weak mathematical formulation, the construction process has its unique characteristics. Since the weak form candidate functions are essentially tied to the discovery process itself, we delineate them in Section 3.4.3 along with the method that utilises the weak form candidate terms. Hence, this section focuses on the construction of strong-form candidate functions.

**Algorithm 1** Adaptive 2D Savitzky-Golay denoising

**Inputs:** Transposed noisy data matrix $\mathbf{U} \in \mathbb{R}^{N_t \times N_x}$, number of window size candidates $K$ (default: 40)
**Output:** Denoised transposed data matrix $\hat{\mathbf{U}} \in \mathbb{R}^{N_t \times N_x}$
$p = 3$ ▷ Fixed polynomial order
$w_{\max} = \min(\lfloor N_t/5 \rfloor, \lfloor N_x/5 \rfloor)$ ▷ Determine max window size
$\mathcal{W}_{\text{raw}}$ = sequence of $K$ logarithmically spaced values from 5 to $w_{\max}$
$\mathcal{W}$ = unique odd integers derived from $\mathcal{W}_{\text{raw}}$
$AIC_{\min} = \infty$
$w^* = 5$ ▷ Initialize best window
**for** $w \in \mathcal{W}$ **do**
    $\mathbf{U}_s$ = SG_filter($\mathbf{U}$, window = $w$, degree = $p$, axis = 1) ▷ Spatial smoothing
    $\mathbf{U}_t$ = SG_filter($\mathbf{U}_s$, window = $w$, degree = $p$, axis = 0) ▷ Temporal smoothing
    $RSS = \sum_{i=1}^{N_t} \sum_{j=1}^{N_x} \left(U_{i,j} - (\mathbf{U}_t)_{i,j}\right)^2 + 10^{-10}$
    $k_{\text{eff}} = \left(\frac{N_t \cdot N_x}{w^2}\right)(p+1)^2$ ▷ Heuristic 2D degrees of freedom
    $AIC = (N_t \cdot N_x) \ln\left(\frac{RSS}{N_t \cdot N_x}\right) + 2k_{\text{eff}}$
    **if** $AIC < AIC_{\min}$ **then**
        $AIC_{\min} = AIC$
        $w^* = w$
    **end if**
**end for**
$\hat{\mathbf{U}}_s$ = SG_filter($\mathbf{U}$, window = $w^*$, degree = $p$, axis = 1)
$\hat{\mathbf{U}}$ = SG_filter($\hat{\mathbf{U}}_s$, window = $w^*$, degree = $p$, axis = 0) ▷ Final 2D smoothing
**return** $\hat{\mathbf{U}}$

### 3.3.1 Candidate function library for strong-form methods

The goal of system identification is to find the optimal combination of mathematical terms that best describe the dynamics behind the observed data. In strong-form methods, such as RIMSI and PDE-FIND, this process begins by constructing a candidate function library $\mathbf{\Theta}(\mathbf{U})$.

Since experimental data are typically discrete, constructing the library requires numerical estimation of the spatial and temporal derivatives. In practice, we choose the calculation method based on whether the data contains noise.

In the case of noise-free data, the second-order finite differences method is used to estimate derivatives. This method is computationally efficient and provides high accuracy when no noise is present. For example, the first spatial derivative at a discrete point $x_i$ is calculated as a central difference using the formula:

$$u_x(x_i, t) \approx \frac{u(x_{i+1}, t) - u(x_{i-1}, t)}{2\Delta x}, \tag{9}$$

and the second spatial derivative is calculated as

$$u_{xx}(x_i, t) \approx \frac{u(x_{i+1}, t) - 2u(x_i, t) + u(x_{i-1}, t)}{(\Delta x)^2}. \tag{10}$$

However, in noisy data, direct numerical differentiation amplifies measurement errors considerably [27]. To alleviate this, the data is smoothed by fitting local polynomials within a sliding window. This method traverses the data points using a sliding window consisting of $2w + 1$ data points around each centre point of interest, where $w$ is the radius of the window.

A $p$-degree polynomial, constructed by Chebyshev polynomials $T_k(x)$, is fitted to this local set of points. The coefficients $c_k$ of the Chebyshev polynomials are found such that the resulting polynomial $P(x) = \sum_{k=0}^{p} c_k T_k(x)$ minimises the sum of squares $\sum_{j=i-w}^{i+w} (u(x_j, t) - P(x_j))^2$. Once the polynomial is fitted to the window region, derivatives are calculated analytically from this fitted function.

For the purposes of discovering PDE dynamics, the library typically includes state variables, their higher-order partial derivatives, and nonlinear combinations of these, which can be presented schematically as

$$\mathbf{\Theta}(\mathbf{U}) = \begin{bmatrix} | & | & | & | & | & \\ \mathbf{1} & \mathbf{u} & \mathbf{u}^2 & \mathbf{u}_x & \mathbf{u}\mathbf{u}_x & \cdots \\ | & | & | & | & | & \end{bmatrix}. \tag{11}$$

Here, $\mathbf{1} \in \mathbb{R}^{N_x \cdot N_t}$ is a vector of ones. Each candidate function column vector consists of all values for the candidate across all $N_x \cdot N_t$ points of the spatio-temporal data. Hence, if the measurement data has dimensions $N_x \times N_t$ and we have $K$ candidate functions, then $\mathbf{\Theta}(\mathbf{U}) \in \mathbb{R}^{N_x \cdot N_t \times K}$. The time derivative $\mathbf{u}_t$ is moreover computed and given as a column vector, thus containing $N_x \cdot N_t$ entries. Candidate function vectors are constructed in column-major order (stacking spatial snapshots sequentially over time). For instance,

$$\mathbf{u}\mathbf{u}_x = \begin{bmatrix} u(x_1, t_1)u_x(x_1, t_1) \\ u(x_2, t_1)u_x(x_2, t_1) \\ \vdots \\ u(x_n, t_1)u_x(x_n, t_1) \\ u(x_1, t_2)u_x(x_1, t_2) \\ \vdots \\ u(x_{N_x}, t_{N_t})u_x(x_{N_x}, t_{N_t}) \end{bmatrix}. \tag{12}$$

The library is constructed to contain terms of the form $u^p \frac{\partial^d u}{\partial x^d}$, where $p \in \{0, \ldots, P\}$ is the degree of the polynomial and $d \in \{0, \ldots, D\}$ is the order of the derivative. The number of columns in the library is directly determined by its upper bounds as

$$K = (P + 1)(D + 1). \tag{13}$$

For example, with the parameters $P = 5$ and $D = 5$, the library would contain $K = 36$ candidate terms. However, since the weak-form library construction does not allow as flexible candidate terms as

the strong-form construction (as explained in Section 3.4.3), the weak and strong form libraries are matched to contain the same terms, which naturally decreases their number for given $P$ and $D$. The structure of the matrix can be illustrated by examining the transpose of an example library $\boldsymbol{\Theta}(\mathbf{U})^\top$, which shows how each row corresponds to a particular candidate function at different data points:

$$\boldsymbol{\Theta}(\mathbf{U})^\top = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 & \dots & 1 \\ u(x_1,t_1) & u(x_2,t_1) & \dots & u(x_{N_x},t_1) & u(x_1,t_2) & \dots & u(x_{N_x},t_{N_t}) \\ u^2(x_1,t_1) & u^2(x_2,t_1) & \dots & u^2(x_{N_x},t_1) & u^2(x_1,t_2) & \dots & u^2(x_{N_x},t_{N_t}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_x(x_1,t_1) & u_x(x_2,t_1) & \dots & u_x(x_{N_x},t_1) & u_x(x_1,t_2) & \dots & u_x(x_{N_x},t_{N_t}) \\ uu_x(x_1,t_1) & uu_x(x_2,t_1) & \dots & uu_x(x_{N_x},t_1) & uu_x(x_1,t_2) & \dots & uu_x(x_{N_x},t_{N_t}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_{xx}(x_1,t_1) & u_{xx}(x_2,t_1) & \dots & u_{xx}(x_{N_x},t_1) & u_{xx}(x_1,t_2) & \dots & u_{xx}(x_{N_x},t_{N_t}) \end{bmatrix}. \quad (14)$$

The identification problem can then be expressed as a linear system of equations $\mathbf{u}_t = \boldsymbol{\Theta}(\mathbf{U})\mathbf{w}$, where $\mathbf{w}$ is a sparse coefficient vector to be found. A sparse solution is called for since the matrix $\boldsymbol{\Theta}(\mathbf{U})$ is typically highly overdetermined, meaning it contains many more candidate functions than the underlying equation has terms.

## 3.4 Inference methods

Once a suitable candidate library has been generated, the next key step is the actual model inference, that is, the selection of the terms that best explain the observed dynamics. Since PDEs describing physical phenomena typically contain only a small fraction of all possible candidate terms, the optimisation problem essentially comes down to sparse regression [24, 27]. This chapter presents the methods compared in the work, which can be divided into two main categories: strong-form and weak-form approaches.

Strong-form methods, such as the RIMSI algorithm and the widely known PDE-FIND, are based on directly solving the equation using pointwise derivative estimates. Although such algorithms are mathematically intuitive, their well-known limitation is their sensitivity to noise in the measurement data, as numerical differentiation inevitably amplifies high-frequency errors. Due to this limitation, the section also discusses the weak-form WSINDy method [29] and the E-WSINDy method, which utilises ensemble learning [30].

### 3.4.1 RIMSI

RIMSI, *Robust interpretable model for system identification*, is a novel method for discovering governing equations developed at Aalto University [31]. It was originally developed to identify ODEs, and for this thesis, it is modified to be used with PDEs.

The selection of terms to be included in the final model is a two-step process. In the first step, *forward voting*, the model is built iteratively one term at a time. The algorithm performs $B$ bootstrap samples of size $b$ at each iteration.

The target variable is the time derivative vector $\mathbf{u}_t \in \mathbb{R}^{N_x \cdot N_t}$ estimated from the data. Let $\boldsymbol{\Theta}_S \in \mathbb{R}^{N_x \cdot N_t \times |S|}$ be a submatrix of the candidate library, which includes only the candidate functions belonging to the currently selected subset $S$. The algorithm tries each candidate term that has not yet been selected, adding it to the model one by one. Thus, forward voting begins by adding individual terms to an empty model, one at a time. Let $\mathbf{u}_b \in \mathbb{R}^b$ denote the time derivative vector restricted to a specific bootstrap sample, and let $\boldsymbol{\Theta}_{S,b} \in \mathbb{R}^{b \times |S|}$ represent the corresponding submatrix. The performance of each subset is assessed by computing an ordinary least squares (OLS) estimate of the coefficients as

$$\hat{\mathbf{w}}_S = (\boldsymbol{\Theta}_{S,b}^\top \boldsymbol{\Theta}_{S,b})^{-1} \boldsymbol{\Theta}_{S,b}^\top \mathbf{u}_b. \quad (15)$$

After this, the projection residual vector is calculated, which tells the difference between the actual time derivative and the prediction given by the model as

$$\mathbf{r}_S = \mathbf{u}_b - \boldsymbol{\Theta}_{S,b} \hat{\mathbf{w}}_S. \quad (16)$$

Finally, how well an added candidate term complements the model is measured by the voting error $E(S)$, which is the $L_2$ norm of this residual vector as

$$E(S) = \|\mathbf{r}_S\|_2 = \sqrt{\sum_{i=1}^{b} r_{S,i}^2}, \tag{17}$$

where $r_{S,i}$ refers to the individual elements of the residual vector. In one bootstrap sample, the term that produces the smallest voting error earns one vote. After all $B$ samples have been processed, the term that received the most votes is retained as a permanent part of the model.

This incremental addition is continued until adding a new term does not decrease sufficiently the projection error, which is defined as the standard deviation of the residual vector of a bootstrap sample, denoted $E_{\text{proj}}(S) = \sigma(\mathbf{r}_S)$. The forward voting phase stops when the absolute change in the average projection error over all bootstrap samples between the last two iterations falls below a set threshold $\epsilon_{\text{fwd}}$, which is scaled by the standard deviation of the entire time derivative $\sigma(\mathbf{u}_t)$. The forward phase thus ends when

$$\left|\bar{E}_{\text{current}} - \bar{E}_{\text{last}}\right| \leq \epsilon_{\text{fwd}}\sigma(\mathbf{u}_t) \tag{18}$$

where $\bar{E}_{\text{current}}$ and $\bar{E}_{\text{last}}$ denote the average of the standard deviations $E_{\text{proj}}(S)$ over all $B$ bootstrap samples at the current and previous iterations, respectively.

Since forward selection selects terms iteratively and does not consider how a term added at the current iteration complements terms to be included later, the algorithm is completed with the *backward voting* phase. First, the entire model produced by the forward voting phase is used, and the algorithm analyses how removing a single term $i$ that has already been selected affects the model's overall mean projection error, denoted by $E_i$, again using the standard deviation of the residuals. The term whose removal results in the lowest projection error is selected for removal. If the increase in error is within the allowed absolute tolerance $\epsilon_{\text{bwd}}\sigma(\mathbf{u}_t)$, the term is permanently dropped from the model. Now, the model contains one fewer terms, and the process continues with the current model until the error increases too much. Additionally, the model must contain at least one term, and this will ultimately stop the backward voting phase if the error threshold is not crossed at any point before then.

After both phases, once the optimal, sparse structure of the equation has been locked in, the model's final coefficients are estimated via OLS averaging over bootstrap samples. The RIMSI pseudo-code, with both forward and backward voting phases, is given in Algorithm 2.

### 3.4.2 Sparse regression via STRidge (PDE-FIND)

Standard sparse model identification methods treat equation discovery as an overdetermined linear regression problem. The original SINDy algorithm uses a sequentially thresholded least-squares method [24], and its extension to partial differential equations, PDE-FIND, introduces sequential threshold ridge regression (STRidge) to address the challenges of PDE finding [27].

The STRidge algorithm starts by computing an initial estimate of the coefficients using standard $\ell_2$-regularised ridge regression:

$$\mathbf{w}^{(0)} = \arg\min_{\mathbf{w}} \|\mathbf{u}_t - \boldsymbol{\Theta}(\mathbf{U})\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2. \tag{19}$$

The $\ell_2$ penalty term is controlled by the hyperparameter $\lambda$ and is used to ensure numerical stability. This is especially important for finding PDE equations, as the candidate library $\boldsymbol{\Theta}(\mathbf{U})$ often has strong correlations between different terms.

After the initial regression, a hard threshold is applied, which is inspired by $\ell_1$-regularised Lasso methods [50] to ensure sparsity in the model. All coefficients with absolute values below a specified threshold `tol` are forced to zero. It is assumed that these insignificant terms explain only the observational noise, not the underlying dynamics. The indices of the thresholded terms are removed from the active candidate library. The regression is then rerun with only the remaining active features. This alternating process of thresholding and refitting continues until the active set of coefficients stabilises or a predefined maximum number of iterations is reached.

To find the optimal threshold value, PDE-FIND places STRidge inside an outer training loop, as shown in Algorithm 3. The algorithm tests different `tol` values by training the model on the training

**Algorithm 2** RIMSI (Adopted from [49])

**Inputs:** Candidate library $\mathbf{\Theta} \in \mathbb{R}^{N_x \cdot N_t \times K}$, time derivative $\mathbf{u}_t \in \mathbb{R}^{N_x \cdot N_t}$, bootstrap iterations $B$, sample size $b$, thresholds $\epsilon_{\text{fwd}}$ and $\epsilon_{\text{bwd}}$
**Output:** Active term indices $\mathcal{I}_{\text{active}}$, final coefficients $\hat{\mathbf{w}}$
Initialize active set $\mathcal{I}_{\text{active}} = \emptyset$, candidates $C = \{1, \ldots, K\}$
Initialize error bounds: $\bar{E}_{\text{last}} = \infty$, $\bar{E}_{\text{current}} =$ large value
$\sigma = \text{std}(\mathbf{u}_t)$ ▷ Standard deviation of the time derivative
**Phase 1: Forward voting**
**while** $|\bar{E}_{\text{current}} - \bar{E}_{\text{last}}| > \sigma\epsilon_{\text{fwd}}$ **and** $C \neq \emptyset$ **do**
    Initialize vote counts $v \in \mathbb{R}^{K \times 1}$ with $v(k) = 0$
    **for** $k = 1, \ldots, B$ **do**
        $\mathcal{I}_b \sim \mathcal{U}\{1, \ldots, N_x \cdot N_t\}$ of size $b$ ▷ Sample bootstrap indices with replacement
        $\mathbf{u}_b = \mathbf{u}_t(\mathcal{I}_b)$
        $k^* = \arg\min_{k \in C} \min_{\mathbf{w}} \|\mathbf{u}_b - \mathbf{\Theta}(\mathcal{I}_b, \mathcal{I}_{\text{active}} \cup \{k\})\mathbf{w}\|_2$ ▷ Best candidate for this subset
        $v(k^*) = v(k^*) + 1$
    **end for**
    $k_{\text{best}} = \arg\max_{k \in C} v(k)$ ▷ Select overall winner
    $\mathcal{I}_{\text{active}} = \mathcal{I}_{\text{active}} \cup \{k_{\text{best}}\}, \quad C = C \setminus \{k_{\text{best}}\}$
    $\bar{E}_{\text{last}} = \bar{E}_{\text{current}}$
    Compute new $\bar{E}_{\text{current}}$ by averaging projection error of $\mathcal{I}_{\text{active}}$ over $B$ new bootstraps
**end while**
**if** $|\mathcal{I}_{\text{active}}| > 1$ **and** $C \neq \emptyset$ **then**
    Remove the last added term from $\mathcal{I}_{\text{active}}$ ▷ Discard term that triggered stop condition
    $\bar{E}_{\text{current}} = \bar{E}_{\text{last}}$
**end if**
**Phase 2: Backward voting**
**while** $|\bar{E}_{\text{current}} - \bar{E}_{\text{last}}| < \sigma\epsilon_{\text{bwd}}$ **and** $|\mathcal{I}_{\text{active}}| > 1$ **do**
    **for** $i \in \mathcal{I}_{\text{active}}$ **do**
        Compute mean projection error $E_i$ using model $\mathcal{I}_{\text{active}} \setminus \{i\}$ over $B$ bootstraps
    **end for**
    $d = \arg\min_{i \in \mathcal{I}_{\text{active}}} E_i$ ▷ Identify least important term
    $\bar{E}_{\text{last}} = \bar{E}_{\text{current}}, \quad \bar{E}_{\text{current}} = E_d$
    **if** $|\bar{E}_{\text{current}} - \bar{E}_{\text{last}}| < \sigma\epsilon_{\text{bwd}}$ **then**
        $\mathcal{I}_{\text{active}} = \mathcal{I}_{\text{active}} \setminus \{d\}$
    **else**
        **break**
    **end if**
**end while**
**Phase 3: Final coefficient estimation**
Initialize coefficient matrix $\mathbf{W} \in \mathbb{R}^{B \times |\mathcal{I}_{\text{active}}|}$
**for** $k = 1, \ldots, B$ **do**
    $\mathcal{I}_b \sim \mathcal{U}\{1, \ldots, N_x \cdot N_t\}$ of size $b$
    $\mathbf{u}_b = \mathbf{u}_t(\mathcal{I}_b)$
    $\mathbf{A}_b = \mathbf{\Theta}(\mathcal{I}_b, \mathcal{I}_{\text{active}})$
    $\mathbf{w}^{(k)} = (\mathbf{A}_b^\top \mathbf{A}_b)^{-1} \mathbf{A}_b^\top \mathbf{u}_b$
    $\mathbf{W}(k, :) = \mathbf{w}^{(k)}$
**end for**
**return** $\mathcal{I}_{\text{active}}$, $\hat{\mathbf{w}} = \text{mean}(\mathbf{W})$

data and validating it on a separate testing set. The threshold that minimises the error term, which combines the complexity and accuracy of the model on the test data, is chosen.

In this work, the sparse model generated by the PDE-FIND algorithm can be further refined by applying backward voting (BV) to it, which follows similar logic as in the RIMSI method. The procedure, also shown in Algorithm 3, starts with the set of active terms found by the STRidge algorithm. First, a standard error based on the entire data is calculated for the current model. Here, as in the RIMSI algorithm, the standard deviation of the residuals, called the projection error, is used as the error. The algorithm then tests the importance of each term by removing it in turn. The effect of removing a term is estimated using $B$ bootstrap samples of size $b$. For each sample, we calculate how much the error increases when the term in question is missing from the model. The term whose removal causes the smallest increase in error on average across the samples is set as the weakest term in the model.

Finally, the removal of this weakest term is examined again on the entire data. If the projection error of the new model, truncated by one term, does not exceed the error of the original PDE-FIND model by more than the allowed tolerance $\epsilon_{\text{bwd}}\sigma(\mathbf{u}_t)$, the term is dropped permanently. This elimination is repeated iteratively until removing the remaining terms would increase the error above the tolerance or until only one term remains in the model. The final coefficients of the remaining terms are estimated using the standard least squares method using the entire available data.

### 3.4.3 Weak SINDy

A well-known weakness of the traditional SINDy algorithm is its sensitivity to measurement noise, which is due to the algorithm's requirement to estimate the derivatives of the state variables pointwise from noisy data [27]. Standard difference methods act as high-pass filters that amplify high-frequency noise in the data. Weak SINDy, as first presented in [29], circumvents this problem by converting the strong form of the partial differential equations to a weak formulation.

Suppose that the dynamics of the system is described by the equation $u_t = N(u, u_x, u_{xx}, \dots)$, where $N$ is some nonlinear operator. If the terms of the operator can be expressed as derivatives of functions of the state variable, this equation can be turned into a weak formulation. In this formulation, the equation is multiplied by a compactly supported and smooth test function $\psi(x,t)$, after which it is integrated over the spatial and temporal domains. By utilising integration by parts, the derivative operators applied to the original data can be transferred to operate on the analytically known test function. For example, in the case of the time derivative $u_t$, we get

$$\iint u_t(x,t)\psi(x,t)\,dx\,dt = -\iint u(x,t)\psi_t(x,t)\,dx\,dt. \tag{20}$$

To eliminate the need for numerical differentiation in noisy data, the terms of the nonlinear operator $N$ must be such that they are representable by a derivative of a function that only depends on the state variable. For example, the nonlinear advection term $uu_x$ in the Kuramoto-Sivashinsky equation is allowed for WSINDy, since it can be expressed as $\frac{1}{2}(u^2)_x$ due to the chain rule. In this case, integration by parts successfully transfers the derivative to the test function

$$\iint uu_x\psi\,dx\,dt = \iint \frac{1}{2}(u^2)_x\psi\,dx\,dt = -\iint \frac{1}{2}u^2\psi_x\,dx\,dt. \tag{21}$$

Since the test function $\psi(x,t)$ has compact support, that is, the function goes to zero at the edges of the integration region, the boundary term of integration by parts disappears. The numerical differentiation of the noisy data $u(x,t)$ is thus replaced by integration, which acts as a natural low-pass filter. It attenuates noise in the data and, as will be seen in Section 4, enables the identification of dynamics even from very noisy measurements without separate pre-filtering.

A key step for the performance and noise robustness of WSINDy is the choice of the test functions $\psi$ (Algorithm 4). WSINDy utilises separable test functions, meaning a multidimensional test function can be expressed as a product of 1D functions: $\psi(x,t) = \phi_x(x)\phi_t(t)$.

The basis of a single 1D function is a polynomial defined for the reference interval $[-1,1]$ as

$$\phi_p(y) = \begin{cases} (1-y^2)^p, & \text{when } |y| \le 1 \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

**Algorithm 3** PDE-FIND with backward voting (Phases 1 and 2 adopted from the supplement of [27])

**Inputs:** Library $\boldsymbol{\Theta} \in \mathbb{R}^{N_x \cdot N_t \times K}$, time derivative $\mathbf{u}_t \in \mathbb{R}^{N_x \cdot N_t}$, ridge penalty $\lambda$, tolerance step $d_{\text{tol}}$, max outer iterations `maxit`, STR iterations `STR_iters`. BV parameters: bootstrap iterations $B$, sample size $b$, threshold $\epsilon_{\text{bwd}}$
**Output:** Sparse coefficient vector $\mathbf{w}_{\text{best}}$
Split data into training $(\boldsymbol{\Theta}^{\text{train}}, \mathbf{u}_t^{\text{train}})$ and testing $(\boldsymbol{\Theta}^{\text{test}}, \mathbf{u}_t^{\text{test}})$ sets
Set $l_0$ penalty $\eta = 10^{-3}\kappa(\boldsymbol{\Theta}^{\text{train}})$. ▷ $\kappa$ is the condition number of the matrix
$\mathbf{w}_{\text{best}} = \arg\min_{\mathbf{w}} \|\mathbf{u}_t^{\text{train}} - \boldsymbol{\Theta}^{\text{train}}\mathbf{w}\|_2^2$ ▷ Baseline least-squares estimate
$e_{\text{best}} = \|\mathbf{u}_t^{\text{test}} - \boldsymbol{\Theta}^{\text{test}}\mathbf{w}_{\text{best}}\|_2 + \eta\|\mathbf{w}_{\text{best}}\|_0$
`tol` $= d_{\text{tol}}$
**for** `iter` $= 1, \ldots,$ `maxit` **do**
    **Phase 1: STRidge**
    $\mathbf{w} = \arg\min_{\mathbf{w}} \|\mathbf{u}_t^{\text{train}} - \boldsymbol{\Theta}^{\text{train}}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$
    $\mathcal{I}_{\text{active}} = \{1, \ldots, K\}$
    **for** $j = 1, \ldots,$ `STR_iters` **do**
        $\mathcal{I}_{\text{small}} = \{i \in \mathcal{I}_{\text{active}} : |w_i| <$ `tol`$\}$
        $\mathbf{w}(\mathcal{I}_{\text{small}}) = 0, \quad \mathcal{I}_{\text{active}} = \mathcal{I}_{\text{active}} \setminus \mathcal{I}_{\text{small}}$
        **if** $\mathcal{I}_{\text{active}}$ unchanged **or** empty **then break**
        **end if**
        $\mathbf{w}(\mathcal{I}_{\text{active}}) = \arg\min_{\mathbf{w}} \|\mathbf{u}_t^{\text{train}} - \boldsymbol{\Theta}^{\text{train}}(\mathcal{I}_{\text{active}})\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$
    **end for**
    $\mathbf{w}(\mathcal{I}_{\text{active}}) = \arg\min_{\mathbf{w}} \|\mathbf{u}_t^{\text{train}} - \boldsymbol{\Theta}^{\text{train}}(\mathcal{I}_{\text{active}})\mathbf{w}\|_2^2$ ▷ Final unregularized fit
    **Phase 2: Tolerance search**
    $e_{\text{current}} = \|\mathbf{u}_t^{\text{test}} - \boldsymbol{\Theta}^{\text{test}}\mathbf{w}\|_2 + \eta\|\mathbf{w}\|_0$
    **if** $e_{\text{current}} \leq e_{\text{best}}$ **then**
        $e_{\text{best}} = e_{\text{current}}, \quad \mathbf{w}_{\text{best}} = \mathbf{w}, \quad$ `tol` $=$ `tol` $+ d_{\text{tol}}$
    **else** ▷ Tolerance too high; adjust
        `tol` $= \max(0,$ `tol` $- 2d_{\text{tol}})$
        $d_{\text{tol}} = 2d_{\text{tol}}/($`maxit` $-$ `iter`$)$
        `tol` $=$ `tol` $+ d_{\text{tol}}$
    **end if**
**end for**
**Phase 3: Backward voting (Optional)**
$\mathcal{I}_{\text{active}} = \{i : w_{\text{best},i} \neq 0\}$
$\mathbf{w}_{\text{OLS}} = \arg\min_{\mathbf{w}} \|\mathbf{u}_t - \boldsymbol{\Theta}(\mathcal{I}_{\text{active}})\mathbf{w}\|_2^2$
$e_{\text{base}} = \text{std}(\mathbf{u}_t - \boldsymbol{\Theta}(\mathcal{I}_{\text{active}})\mathbf{w}_{\text{OLS}})$ ▷ Initial projection error
$\sigma = \text{std}(\mathbf{u}_t)$
**while** $|\mathcal{I}_{\text{active}}| > 1$ **do**
    **for** $i \in \mathcal{I}_{\text{active}}$ **do**
        Compute mean projection error $E_i$ using model $\mathcal{I}_{\text{active}} \setminus \{i\}$ over $B$ bootstrap samples of size $b$
    **end for**
    $d = \arg\min_{i \in \mathcal{I}_{\text{active}}} E_i$ ▷ Identify least important term
    $\mathbf{w}_{\text{trial}} = \arg\min_{\mathbf{w}} \|\mathbf{u}_t - \boldsymbol{\Theta}(\mathcal{I}_{\text{active}} \setminus \{d\})\mathbf{w}\|_2^2$
    $e_{\text{trial}} = \text{std}(\mathbf{u}_t - \boldsymbol{\Theta}(\mathcal{I}_{\text{active}} \setminus \{d\})\mathbf{w}_{\text{trial}})$
    **if** $e_{\text{trial}} < e_{\text{base}} + \sigma\epsilon_{\text{bwd}}$ **then**
        $\mathcal{I}_{\text{active}} = \mathcal{I}_{\text{active}} \setminus \{d\}$
    **else**
        **break**
    **end if**
**end while**
$\mathbf{w}_{\text{best}} = \mathbf{0}$
**if** $|\mathcal{I}_{\text{active}}| > 0$ **then**
    $\mathbf{w}_{\text{best}}(\mathcal{I}_{\text{active}}) = \arg\min_{\mathbf{w}} \|\mathbf{u}_t - \boldsymbol{\Theta}(\mathcal{I}_{\text{active}})\mathbf{w}\|_2^2$ ▷ Final OLS fit
**end if**
**return** $\mathbf{w}_{\text{best}}$

To fit this reference function to the actual computational grid around a query point (for example, a spatial point $x_q$), the variable $y$ is scaled by the support radius $m_x$ and the grid step size $\Delta x$. The spatial test function then takes the form

$$\phi_x(x) = \phi_{p_x}\left(\frac{x - x_q}{m_x \Delta x}\right). \tag{23}$$

This shows the importance of the spatial and temporal dimensions' support radii $m_x$ and $m_t$, since they scale the test function so that it is nonzero in a region exactly $2m$ wide around the query point. Combined with the time coordinate, the complete 2D test function centered at the query point $(x_q, t_q)$ is thus

$$\psi_q(x,t) = \phi_{p_x}\left(\frac{x - x_q}{m_x \Delta x}\right)\phi_{p_t}\left(\frac{t - t_q}{m_t \Delta t}\right). \tag{24}$$

The shape of the test function is thus governed by the support radius $m$, which determines the size of the integration window on the physical grid, and the polynomial degree $p$, which determines the smoothness of the function. To ensure exact analytical derivatives, the polynomial degree is dynamically chosen as the minimum integer $p \geq D + 1$ such that the function decays below a strict numerical tolerance $\tau$ near the edges of its support domain.

As a result of the integration operation, the overdetermined linear system of equations $\mathbf{u}_t = \boldsymbol{\Theta}(\mathbf{U})\mathbf{w}$ of traditional SINDy is transformed into a weak-form linear system $\mathbf{b} = \mathbf{G}\mathbf{w}$, where $\mathbf{b}$ is the target vector, $\mathbf{G}$ is the Gram matrix, and $\mathbf{w}$ contains the model coefficients. Although matrix ill-conditioning is a common problem in traditional SINDy as well, the mechanism of the problem is different in WSINDy due to large polynomials and numerical derivatives. In WSINDy, candidate functions are integrated against the derivatives of the test function ($D^{\alpha^s}\psi$). In the case of high derivatives, the test functions oscillate strongly. As a consequence, the condition number of the Gram matrix $\kappa(\mathbf{G})$ can be large, especially if the original data is poorly scaled.

In traditional SINDy, matrix ill-conditioning is often remedied by normalising the matrix columns, as is done with PDE-FIND, but this has no connection to the underlying physics of the system. WSINDy solves the problem in a numerically more stable and physically justified way. The state variable $u$ and the spatial and temporal grids are first scaled using the scale invariance of the partial differential equation. The scaling factors are chosen so that the norms of the columns of the resulting scaled Gram matrix $\tilde{\mathbf{G}}$ remain close to the norm of the original data, which significantly reduces the condition number of the matrix. From these scaled variables $(\tilde{u}, \tilde{x}, \tilde{t})$, a scaled weak Gram matrix $\tilde{\mathbf{G}}$ and a scaled target vector $\tilde{\mathbf{b}}$ are constructed.

As a result of this spatio-temporal scaling, the linear system to be solved has the form

$$\tilde{\mathbf{b}} = \tilde{\mathbf{G}}\mathbf{w}. \tag{25}$$

Here, the vector $\tilde{\mathbf{b}}$ contains the weak-form time derivatives at each query point $q$. By integration by parts, the derivative operator is transferred to the test function, where the function to be integrated is simply the scaled data multiplied by the time derivative of the test function: $\tilde{b}_q = -\iint \tilde{u}(x,t)(\psi_q)_t \, dx\, dt$.

The weak Gram matrix $\tilde{\mathbf{G}}$, in turn, ties the candidate function library $\boldsymbol{\Theta}(\tilde{\mathbf{U}})$ together with the test functions. The single element of the matrix $\tilde{\mathbf{G}}$ at row $q$ and column $j$ is formed by integrating the $j$th candidate function $\theta_j(\tilde{\mathbf{U}})$ multiplied by the test function $\psi_q(x,t)$ centered at query point $q$:

$$\tilde{G}_{q,j} = \iint \theta_j(\tilde{\mathbf{U}})\psi_q(x,t)\, dx\, dt. \tag{26}$$

If the candidate function $\theta_j$ is polynomial (e.g. $\tilde{u}^2$ or $\tilde{u}^3$), the integral is calculated as such with the derivative-free test function $\psi_q$. If the candidate function contains spatial derivatives, integration by parts transfers the derivative from the data to the test function. In this case, the integrand is the polynomial term of the data multiplied by the corresponding derivative of the test function. WSINDy constructs the matrix $\tilde{\mathbf{G}}$ efficiently by interpreting the integrals as convolutions and utilising the fast Fourier transform (FFT), as shown in Algorithm 5.

Once the candidate library has been integrated and the weak Gram matrix $\tilde{\mathbf{G}}$ has been constructed, we move on to solving the model coefficients $\mathbf{w}$ (Algorithm 6). Since the candidate terms integrated by the test functions can be strongly correlated (especially when high spatial derivatives are involved), WSINDy uses a modified sequential-thresholding least-squares (MSTLS) algorithm for optimisation.

The MSTLS algorithm operates in two nested loops. First, in the outer loop, the algorithm traverses a predefined search space of sparsity thresholds $\Lambda$. For each individual threshold value $\lambda \in \Lambda$, the algorithm performs successive thresholding in the inner loop. MSTLS sets dynamic lower and upper bounds $L_i^\lambda$ and $U_i^\lambda$ for each coefficient $w_i$. The bounds are scaled with respect to the norm of the target vector $\|\tilde{\mathbf{b}}\|_2$ and the norm of the corresponding column of the Gram matrix $\|\tilde{\mathbf{G}}_i\|_2$:

$$L_i^\lambda = \lambda \max\left\{1, \frac{\|\tilde{\mathbf{b}}\|_2}{\|\tilde{\mathbf{G}}_i\|_2}\right\}, \quad U_i^\lambda = \frac{1}{\lambda} \min\left\{1, \frac{\|\tilde{\mathbf{b}}\|_2}{\|\tilde{\mathbf{G}}_i\|_2}\right\}. \tag{27}$$

In this inner loop, MSTLS identifies those coefficients $w_i$ that fall outside the allowed interval $[L_i^\lambda, U_i^\lambda]$. The candidate terms corresponding to these coefficients are removed from the active set, and the model is refitted using the least squares method with the remaining terms. Threshold and refit are repeated until the active set of coefficients stabilises. The result is a converged, sparse model for that $\lambda$ value.

After the algorithm has generated a model alternative for each $\lambda \in \Lambda$, it finally selects the sparsity threshold $\hat{\lambda}$ that minimizes the loss function $\mathcal{L}(\lambda)$, which is defined as

$$\mathcal{L}(\lambda) = \frac{\|\tilde{\mathbf{G}}(\mathbf{w}^\lambda - \mathbf{w}^{LS})\|_2}{\|\tilde{\mathbf{G}}\mathbf{w}^{LS}\|_2} + \frac{\|\mathbf{w}^\lambda\|_0}{K}, \tag{28}$$

where $\mathbf{w}^\lambda$ is the sparse coefficient vector obtained with the threshold value $\lambda$ and $\mathbf{w}^{LS}$ is the original unconstrained least squares solution. In the latter term, $\|\mathbf{w}^\lambda\|_0$ denotes the number of non-zero coefficients (the so-called $L_0$ norm), and $K$ is the total number of terms in the candidate library. Thus, the loss function balances the model's relative residual error (the first term) and its complexity (the second term). The coefficient vector $\mathbf{w}^{\hat{\lambda}}$ corresponding to the optimal threshold value $\hat{\lambda}$ is chosen as the final coefficient vector $\hat{\mathbf{w}}$.

**Algorithm 4** WSINDy Test Function Creation (Adopted from [29])

**Inputs:** Discrete support $m$, real-space decay tolerance $\tau$, computational grid $X$, maximum derivative order $D$
**Output:** Test function analytical derivatives $(\phi^{(d)}(Y))_{0\le d\le D}$
$N = \text{length}(X)$, $\Delta x = \text{gridwidth}(X)$
**if** $m > \frac{N-1}{2}$ **or** $m \le 1$ **then**
    **return** Error: invalid support size
**end if**
$p = \min\left\{p' \ge D+1 : \phi_{p'}\left(1 - \frac{1}{m}\right) \le \tau\right\}$ ▷ Determine required polynomial degree
$n_{\text{grid}} = \left\{-1 + \frac{n}{m} : n \in \{0, \dots, 2m\}\right\}$ ▷ Scaled reference grid
**for** $d = 0$ **to** $D$ **do**
    $A_d = \phi_p^{(d)}(n_{\text{grid}})$ ▷ Compute analytical derivatives of the scaled test function
    $\phi^{(d)}(Y) = \frac{1}{(m\Delta x)^d} A_d$ ▷ Scale back to original physical dimensions
**end for**
**return** $(\phi^{(d)}(Y))_{0\le d\le D}$

### 3.4.4 Ensemble Weak SINDy

Although it significantly improves the robustness of model identification against noise through integration by parts, WSINDy may still be sensitive to cross-correlation between terms in the candidate library, especially with large libraries and high-order derivatives. To address this challenge, Fasel et al. [30] introduced the Ensemble SINDy (E-SINDy) method, which utilises techniques familiar from ensemble learning. In this work, the method is combined with the WSINDy algorithm, resulting in the ensemble weak SINDy (E-WSINDy) method.

Our implementation utilises a two-stage ensemble learning strategy (Algorithm 7) that combines library bagging and traditional data bragging.

First, at each iteration $m$, a random subset of terms is selected from the candidate library (for example, $\rho_{\text{lib}} = 0.8$, so that 80 % of the columns are included). The MSTLS algorithm of WSINDy

**Algorithm 5** WSINDy for Partial Differential Equations (Adopted from [29])

**Inputs:** Data matrix $\mathbf{U}$, grid $(X, t)$, highest polynomial power $P$, maximum spatial derivative order $D$, test function parameters $\mathbf{m}$, $\mathbf{s}$, $\tau$, threshold search space $\Lambda$
**Output:** Sparse model coefficients $\hat{\mathbf{w}}$ and optimal threshold $\hat{\lambda}$
$\phi_x^{(d)}(Y_x) = \text{WSINDy Test Function Creation}(m_x, \tau, X, D)$
$\phi_t^{(d)}(Y_t) = \text{WSINDy Test Function Creation}(m_t, \tau, t, 1)$ ▷ Time derivative only requires order 1
Compute physical scale factors $\{\gamma_u, \gamma_x, \gamma_t\}$ and assemble diagonal matrix $M$
Subsample query points $\{(x_q, t_q)\}_{q=1}^{Q} \subset (X, t)$ using frequencies $\mathbf{s} = (s_x, s_t)$
Compute target vector $\tilde{\mathbf{b}} = \tilde{\Psi}^{(0)} * \tilde{\mathbf{U}}$ over query points using FFT and separability of $\psi$
$K = (P+1)(D+1)$ ▷ Total number of candidate terms
**for** $n = 0$ **to** $P$ **do**
    Compute non-linear candidate base $\tilde{f}_n(\tilde{\mathbf{U}}) = \tilde{\mathbf{U}}^n$
    **for** $d = 0$ **to** $D$ **do**
        Compute column index $j = d(P+1) + (n+1)$
        $\tilde{\mathbf{G}}_{:,j} = \tilde{\Psi}^{(d)} * \tilde{f}_n(\tilde{\mathbf{U}})$ using FFT and separability of $\psi$
    **end for**
**end for**
$\hat{\mathbf{w}}, \hat{\lambda} = \text{MSTLS}(\tilde{\mathbf{G}}, \tilde{\mathbf{b}}; \mathcal{L}, \Lambda, K)$
**return** $\hat{\mathbf{w}}, \hat{\lambda}$

**Algorithm 6** Modified Sequential-Thresholding Least-Squares (MSTLS) (Adopted from [29])

**Inputs:** Gram matrix $\mathbf{G} \in \mathbb{R}^{Q \times K}$, target vector $\mathbf{b} \in \mathbb{R}^{Q}$, search space $\Lambda$, total candidate terms $K$
**Output:** Sparse coefficients $\hat{\mathbf{w}}$ and optimal threshold $\hat{\lambda}$
$\mathbf{w}^{LS} = (\mathbf{G}^\top \mathbf{G})^{-1} \mathbf{G}^\top \mathbf{b}$ ▷ Compute initial unconstrained least-squares
**for** $\lambda \in \Lambda$ **do**
    **for** $i = 1$ **to** $K$ **do**
        $L_i^\lambda = \lambda \max\left\{1, \frac{\|\mathbf{b}\|_2}{\|\mathbf{G}_i\|_2}\right\}$ ▷ Establish lower bound
        $U_i^\lambda = \frac{1}{\lambda} \min\left\{1, \frac{\|\mathbf{b}\|_2}{\|\mathbf{G}_i\|_2}\right\}$ ▷ Establish upper bound
    **end for**
    $\mathbf{w}^0 = \mathbf{w}^{LS}$
    $l = 0$
    **repeat**
        $\mathcal{I}^l = \{1 \le i \le K : L_i^\lambda \le |w_i^l| \le U_i^\lambda\}$ ▷ Identify valid active set
        $\mathbf{w}^{l+1} = \arg\min_{\text{supp}(\mathbf{w}) \subset \mathcal{I}^l} \|\mathbf{G}\mathbf{w} - \mathbf{b}\|_2^2$ ▷ Refit active set via least-squares
        $l = l + 1$
    **until** $\mathcal{I}^l \setminus \mathcal{I}^{l-1} = \emptyset$
    $\mathbf{w}^\lambda = \mathbf{w}^l$
    $\mathcal{L}(\lambda) = \frac{\|\mathbf{G}(\mathbf{w}^\lambda - \mathbf{w}^{LS})\|_2}{\|\mathbf{G}\mathbf{w}^{LS}\|_2} + \frac{\|\mathbf{w}^\lambda\|_0}{K}$ ▷ Evaluate objective loss
**end for**
$\hat{\lambda} = \min\{\lambda \in \Lambda : \mathcal{L}(\lambda) = \min_{\lambda' \in \Lambda} \mathcal{L}(\lambda')\}$
$\hat{\mathbf{w}} = \mathbf{w}^{\hat{\lambda}}$
**return** $\hat{\mathbf{w}}, \hat{\lambda}$

is applied to a subset of terms, yielding the coefficient vector $\mathbf{w}^{(m)}$, which is stored in the coefficient matrix $\mathbf{W}_{\text{lib}}$. This is repeated $M$ times, resulting in a collection of sparse models fitted with different library subsets. The significance of each term $i$ is assessed by calculating its inclusion probability $P_{\text{inc},i}$, that is, the proportion of models in which the term was selected as

$$P_{\text{inc},i} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}(\mathbf{W}_{\text{lib}}[m, i] \neq 0), \tag{29}$$

where $\mathbb{1}(x)$ is the indicator function which evaluates to 1 if the condition $x$ is true, otherwise 0.

An inclusion threshold probability $\tau_{\text{inc}}$ is defined (for example $\tau_{\text{inc}} = 0.6$). Terms with a probability below the threshold are removed entirely from the final active set $\mathcal{I}_{\text{active}}$. Our E-WSINDy implementation performs a second phase with the active terms, which utilises data bragging. That is, data points are bootstrapped, and a model is inferred for each sample. More precisely, the library is locked to the set $\mathcal{I}_{\text{active}}$, and the algorithm performs MSTLS using $M$ bootstrap samples that have the size of all data points, with replacement. To ensure strict sparsity, a final inclusion probability is calculated for the data bragging models. The final, stable model coefficients $\hat{\mathbf{w}}$ are obtained by taking the median values of the samples' model coefficients after the second stage thresholding.

**Algorithm 7** Two-step Ensemble Weak SINDy (Adopted from [29])

**Inputs:** Weak Gram matrix $\tilde{\mathbf{G}} \in \mathbb{R}^{Q \times K}$, target vector $\tilde{\mathbf{b}} \in \mathbb{R}^{Q}$, number of ensemble models $M$, library sample ratio $\rho_{\text{lib}}$, inclusion threshold $\tau_{\text{inc, lib}}$ and $\tau_{\text{inc, data}}$ for both phases
**Output:** Robust coefficients $\hat{\mathbf{w}}$
**Phase 1: Library bagging**
**Initialize** coefficient matrix $\mathbf{W}_{\text{lib}} \in \mathbb{R}^{M \times K}$ with zeros
Find optimal threshold $\lambda^*_{\text{lib}}$ using standard MSTLS with full $\tilde{\mathbf{G}}, \tilde{\mathbf{b}}$
**for** $m = 1$ **to** $M$ **do**
    $\mathcal{I}^{(m)}$ = Random sample of $\lfloor \rho_{\text{lib}} \cdot K \rfloor$ terms from library
    $\mathbf{w}^{(m)} = \text{MSTLS}(\tilde{\mathbf{G}}_{:,\mathcal{I}^{(m)}}, \tilde{\mathbf{b}}; \text{fixed } \lambda^*_{\text{lib}})$
    $\mathbf{W}_{\text{lib}}[m, \mathcal{I}^{(m)}] = \mathbf{w}^{(m)}$
**end for**
$P_{\text{inc},i} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}(\mathbf{W}_{\text{lib}}[m, i] \neq 0)$ **for** $i = 1 \dots K$
$\mathcal{I}_{\text{active}} = \{j : P_{\text{inc},j} > \tau_{\text{inc, lib}}\}$ ▷ Ensemble thresholding
**Phase 2: Data bragging on reduced library**
$\tilde{\mathbf{G}}_{\text{reduced}} = \tilde{\mathbf{G}}_{:,\mathcal{I}_{\text{active}}}$
Find optimal threshold $\lambda^*_{\text{data}}$ using standard MSTLS with $\tilde{\mathbf{G}}_{:,\mathcal{I}_{\text{active}}}, \tilde{\mathbf{b}}$
**Initialize** coefficient matrix $\mathbf{W}_{\text{data}} \in \mathbb{R}^{M \times |\mathcal{I}_{\text{active}}|}$
**for** $m = 1$ **to** $M$ **do**
    $\mathcal{R}^{(m)}$ = Random sample of data points $\{1 \dots Q\}$ with replacement
    $\mathbf{w}^{(m)} = \text{MSTLS}(\tilde{\mathbf{G}}_{\text{reduced}}[\mathcal{R}^{(m)}, :], \tilde{\mathbf{b}}[\mathcal{R}^{(m)}]; \text{fixed } \lambda^*_{\text{data}})$
    $\mathbf{W}_{\text{data}}[m, :] = \mathbf{w}^{(m)}$
**end for**
$\hat{\mathbf{w}} = \text{zeros}(K)$
$P_{\text{data},j} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}(\mathbf{W}_{\text{data}}[m, j] \neq 0)$ **for** $j = 1 \dots |\mathcal{I}_{\text{active}}|$ ▷ Second thresholding step
**for** $j = 1$ **to** $|\mathcal{I}_{\text{active}}|$ **do**
    **if** $P_{\text{data},j} > \tau_{\text{inc, data}}$ **then**
        non_zeros = $\{\mathbf{W}_{\text{data}}[m, j] : \mathbf{W}_{\text{data}}[m, j] \neq 0, m \in 1 \dots M\}$
        $\hat{\mathbf{w}}[\mathcal{I}_{\text{active}}[j]] = \text{median}(\text{non\_zeros}[:, j])$ ▷ Aggregate final coefficients
    **end if**
**end for**
**return** $\hat{\mathbf{w}}$

## 3.5 Evaluation metrics

Paramount to our study is to examine the performance of the methods against diverse metrics. There are two main branches in learning a dynamical system from observed data: system identification, where the primary goal is to determine the true underlying equation as precisely as possible, and system prediction, which aims to produce a model that follows the real trajectory as closely as possible. Therefore, the metrics should mirror both objectives. The former goal is addressed by the weighted true positive rate (wTPR) and weighted false positive rate (wFPR), and the latter by valid time and average short-time prediction error metrics.

### 3.5.1 Weighted true positive rate

The standard true positive rate (TPR) is often used to evaluate how well correct terms are discovered. It tracks the number of correctly identified terms against the total number of terms in the true equation, that is:

$$\begin{aligned} \text{TPR} &= \frac{\text{Correct terms in the inferred model}}{\text{Number of terms in the true equation}} \\ &= \frac{\text{True positives}}{\text{True positives + False negatives}}. \end{aligned} \tag{30}$$

However, since in the inferred models the coefficients may vary wildly, the standard true positive rate is not sufficient for comparing models. A poor model for the KS equation, for example, $u_t = -0.2uu_x + 0.07u_x - 0.03u_{xxxx}$ would yield a somewhat reasonable TPR = 0.67 (the true equation being $u_t = -uu_x - u_{xx} - u_{xxxx}$), even though the model itself is far from correct. Additionally, in its standard form, TPR does not account for the coefficient signs, making it possible that a model could obtain a high TPR with coefficients that are not close to the true ones and have wrong signs.

To address these considerations, we introduce the weighted true positive rate (wTPR), which sums the coefficients of the structurally correct terms in a model. Moreover, it takes into account whether the terms have the correct sign. We can define this mathematically as

$$\text{wTPR} = \frac{\sum_{i \in \text{TP},\, \text{sgn}(\hat{w}_i)=\text{sgn}(w_i)} |\hat{w}_i|}{\sum_{j \in \text{true}} |w_j|}, \tag{31}$$

where $w_i$ is the true coefficient, $\hat{w}_i$ is the model's coefficient of the term $i$, and TP (true positives) is the set of all terms that are both in the model and the true KS equation. The denominator adds the absolute values of the KS equation's coefficients. The numerator only counts in the coefficients with the correct sign to ensure a correct term with a wrong sign does not contribute positively to the metric.

wTPR tells directly how much of the real dynamics of the system the model can actually explain. Unlike the standard TPR, wTPR can also exceed 1.0 (or 100%). This is an intentional characteristic of the metric since a value over 1 reveals that the inferred model overestimates the coefficients of the correct terms.

### 3.5.2 Weighted false positive rate

The standard false positive rate is the ratio of incorrectly included terms among all possible incorrect terms. Especially when using this metric to evaluate models inferred from a large number of candidate functions, the number of terms correctly excluded from the model (true negatives) is significant. This results in an FPR value close to 0, even if the model contained multiple incorrect terms. Moreover, as with the standard TPR, the standard FPR does not take into account the magnitude of the incorrect terms' coefficients, and it penalises equally a very small non-zero term as a coefficient with a high value. Therefore, we introduce the weighted false positive rate (wFPR), which is defined mathematically as

$$\text{wFPR} = \frac{\sum_{i \in \text{FP}^*} |\hat{w}_i|}{\sum_{j \in \text{true}} |w_j|}, \tag{32}$$

where, similarly to wTPR, the denominator is the sum of the absolute values of the true coefficients. In the numerator, the absolute values of the coefficients of all incorrect terms in the model are added together. Here, the set FP* also contains correct terms with the wrong sign.

This ratio provides a direct interpretation of the error in the model: if wFPR is 0.20, it means that the combined strength of the erroneous dynamics in the model is 20% of the strength of the real system's dynamics. In addition, because the metric does not divide the error by library size but compares it directly to the system's actual scale, large libraries do not lead to a biased value.

### 3.5.3 Relative coefficient error

The numerical accuracy of individual coefficients is assessed by the relative coefficient error. It measures the percentage deviation between the coefficient $\hat{w}_i$ inferred by the model and the coefficient $w_i$ of the KS equation.

The relative error for a correctly identified term $i$ is defined by

$$E_{\text{rel},i} = \left| \frac{\hat{w}_i - w_i}{w_i} \right| \times 100\,\%. \tag{33}$$

This metric is, naturally, calculated only for those terms that are in the KS equation, which the algorithm has managed to include in the model. The metric is used in tables which report models found by the methods. If the algorithm fails to find a KS equation term, the error cannot be calculated, and it is marked as "inf".

### 3.5.4 Valid time

Valid time measures how long an inferred model's trajectory accurately tracks the true trajectory. During each valid time evaluation, the model is integrated in time, starting from the exact same initial state as the ground truth. The valid time is defined as the duration until the normalised error, $E(t)$, exceeds a predefined threshold $\beta$. The normalised error is computed as

$$E(t) = \frac{||\mathbf{u}(t) - \hat{\mathbf{u}}(t)||_2}{\langle ||\mathbf{u}||_2^2 \rangle^{\frac{1}{2}}}, \tag{34}$$

where $\mathbf{u}(t), \hat{\mathbf{u}}(t) \in \mathbb{R}^m$ are the true and predicted states at time $t$ across $m$ spatial grid points, respectively. To allow for comparison between domains with different magnitudes, the denominator normalises the error by the root-mean-square of the true state's $L_2$ norm, averaged over the entire dataset time horizon.

Formally, the valid time $t_{\text{vt}}$ is given by:

$$t_{\text{vt}} = \min\{t > 0 \mid E(t) \geq \beta\}. \tag{35}$$

We set $\beta = 0.4$, which is consistent with previous studies [49, 51]. Because the initial state of the valid time evaluation influences how rapidly the true and predicted trajectories diverge, the valid time for each inference method and noise level is the average over 20 trials, each with a different initial state drawn from the original noise-free data.

Due to computation time limitations, the maximum evaluated value for valid time is capped at 500, which is empirically observed to be reached only in the travelling wave regime. In the chaotic regime, all valid times across all trials are always less than the maximum in our analyses.

### 3.5.5 Average short-time prediction error

The average short-time prediction error (ASTP error) quantifies an inferred model's ability to predict the system's immediate future states. Typically, this short-time horizon is chosen as one *Lyapunov time* $T_\lambda = 1/\lambda_{\max}$, where $\lambda_{\max}$ is the largest Lyapunov exponent of the system, which is discussed in more detail in Section 2.1.1. However, the Lyapunov exponents of the KS equation are very small: for $L = 22$, for instance, the largest Lyapunov exponent $\lambda_{\max} = 0.043$ [39], which would result in a rather long time horizon of $T_\lambda = 1/0.043 \approx 23.3$. This is especially problematic when models are inferred from noisy data, as some ASTP error calculations may diverge over long time horizons, leaving no information about the model's performance. Hence, we decide to use $\lambda_{\max} = 0.91$, which is the largest Lyapunov exponent of the chaotic Lorenz system in all our ASTP error calculations.

For each trial $i$, the inferred model is integrated forward with $K$ time steps such that, when multiplied by the data time step $\Delta t$, the integration interval corresponds to the closest Lyapunov time $T_\lambda$, that is, $K\Delta t \approx T_\lambda$. The error for a single trial $\varepsilon_i$ is defined as the root mean squared $L_2$ norm of the difference between the true state $\mathbf{u}$ and the predicted state $\hat{\mathbf{u}}$. This is then normalised by the standard deviation $\sigma$ of the entire ground truth dataset to account for the typical fluctuation of the system's states as

$$\varepsilon_i = \frac{1}{\sigma}\sqrt{\frac{1}{K}\sum_{k=1}^{K} ||\mathbf{u}(t_k) - \hat{\mathbf{u}}(t_k)||_2^2}. \tag{36}$$

To ensure the robustness of the metric, the calculation is averaged over $N = 20$ distinct, uniformly spaced initial conditions sampled from the data set. The final ASTP error $\varepsilon$ is defined as the root-mean-square of these individual trial errors as

$$\varepsilon = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \varepsilon_i^2}. \tag{37}$$

If the inferred model diverges numerically and fails to complete the integration before reaching the Lyapunov time limit, the error for that specific trial is penalised by setting it to infinity.

# 4 Empirical results

This section presents the empirical results of the study. After laying out simulation parameters in Section 4.1, the results are divided into two main sections. First, the ability of the methods to identify equations from ideal, noise-free data is assessed in Section 4.2, after which their robustness to varying degrees of observational noise is examined in Section 4.3.

The sections display different evaluation metrics. In the case of noise-free data and large candidate function libraries, we are interested in the structures of the inferred models. In this case, we present the models produced by the algorithms and focus on examining the relative errors of the coefficients of individual terms. When multiple levels of observational noise are added to the data, and multiple methods are used to infer models for each noise level, presenting each inferred equation is not meaningful. Therefore, in evaluating noisy results, the focus shifts to more general metrics. Valid time and average short-time prediction error measure the prediction power of the models. Regarding the system identification perspective, the wTPR and wFPR metrics evaluate the methods' ability to infer the correct equation structure.

## 4.1 Simulation parameters

### 4.1.1 Dynamical system configurations

In all our simulations, the time step is fixed at $\Delta t = 0.25$, and the total simulation time is $T = 256$, so there are $N_t = 1024$ time points for each spatial point. The number of spatial points is chosen as $N_x = 512$; hence, the distance between spatial points, $\Delta x$, varies with the domain size $L$. For example, $L = 22$ yields $\Delta x = \frac{L}{N_x} = \frac{22}{512} = 0.043$. Data of this size is produced after the simulation runs for an initial transient period of $T_{\text{transient}} = 100$. This is to ensure that the system's initialisation phase is skipped, and we can focus on identifying the stationary dynamics. Nonetheless, especially in the travelling-wave regime, the transient period has an effect, and this will be investigated in Section 4.2.2.

In the travelling wave regime, the initial condition is set as $u_0 = \sin((2\pi x)/L) + 0.1$, and in the chaotic regime, the initial condition is $u_0 = \cos((2\pi x)/L) + 0.2 \cdot \cos((4\pi x)/L)$.

### 4.1.2 Inference algorithm configurations

Although in different forms, all investigated methods require a candidate function library as input. Since the candidate terms in this study consist of polynomial cross-terms and spatial derivatives, two hyperparameters determine the size of the library: the polynomial degree $P$ and the derivative order $D$. We investigate the performance of the algorithms with candidate libraries of different sizes. For the smallest candidate library in this study, we select $P = 5$ and $D = 5$. We also test the methods on larger libraries, with $P$ and $D$ up to 20. Whenever results are presented, the values of $P$ and $D$ used are mentioned.

The weak candidate term formulation of WSINDy does not allow as flexible products of polynomials and spatial derivatives as the strong formulation, as explained in Section 3.4.3. Hence, to provide an equal basis for comparison, we match the weak- and strong-form libraries so that they contain the same candidate functions. For example, with the smallest hyperparameters $P = 5$ and $D = 5$ used, the candidate function library then contains 15 candidates, which are

- Constant term: 1
- Polynomial terms: $u, u^2, u^3, u^4, u^5$
- Spatial derivatives: $u_x, u_{xx}, u_{xxx}, u_{xxxx}, u_{xxxxx}$
- Mixed terms: $uu_x, u^2u_x, u^3u_x, u^4u_x$

The different algorithms, moreover, require the setting of their specific hyperparameters. In the case of the RIMSI, the iteration process is controlled by $B = 300$ bootstrap samples, each of which has a sample size of $b = 200$ data points. The stopping conditions for the forward and backward steps are defined by the tolerances $\epsilon_{\text{fwd}}$ and $\epsilon_{\text{bwd}}$, both of which are 0.01.

The execution of the STRidge algorithm, which is in the core of the PDE-FIND method, is controlled by setting the initial tolerance to 5, which corresponds to the parameter $d_{\text{tol}}$ presented in Algorithm 3. This parameter determines not only the initial threshold value (`tol`), but also the step size by which the threshold value is adjusted during the tolerance search. The tolerance search is allowed to perform a maximum of 50 iterations (`maxit` in the pseudocode) to find the optimal threshold value. To ensure numerical stability, the coefficient $\lambda$ of the $\ell_2$ regularisation used in sparse regression is chosen to be $10^{-5}$.

In the case of noisy data, when constructing the spatial derivatives for the strong-form candidate library, the polynomial interpolation uses a fifth-degree Chebyshev polynomial (explained in Section 3.3.1) in a sliding window with a radius of 10 data points.

The number of query points $Q$ for the weak-form methods is determined by dynamic subsampling. Instead of using a fixed step size for subsampling, the step size in dimension $d$ is calculated as $s_d = \lfloor N_d/50 \rfloor$, where $N_d$ is the total number of data points in that dimension. For the data dimensions used in this work ($N_x = 512$, $N_t = 1024$), this corresponds to step sizes of $s_x = 10$ and $s_t = 20$ data points.

In the E-WSINDy method, an ensemble of $M = 1000$ separate models is used. In the library bagging stage, 80 % of the terms are randomly selected from the candidate library for each model, and a term is included for further consideration if it occurs in more than half (threshold value 0.5) of the generated models. In the second stage, which utilises the original data sampling with resampling, the final elimination is done with a threshold value of 0.4.

### 4.1.3 Observational noise levels

The main purpose of the study is to examine the robustness of the methods in the presence of observational noise. Gaussian noise with standard deviations $\sigma_{\text{obs}} \in \{10^{-8}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 10^{-2}, 5 \cdot 10^{-2}, 10^{-1}, 5 \cdot 10^{-1}, 10^{0}, 5 \cdot 10^{0}, 10^{1}\}$, as per Equation (5) are used as noise levels.

## 4.2 Systems without observational noise

### 4.2.1 Moderately chaotic system with large candidate libraries

To test the performance of the methods against very large candidate libraries on noise-free data, we utilise the following libraries. The first library contains terms up to the 15th polynomial degree and the 15th spatial derivative:

$$\mathcal{L}_1 = \{1, u, u^2, \ldots, u^{15}, u_x, uu_x, \ldots, u^{14}u_x, u_{xx}, u_{xxx}, \ldots, u^{(15)}\}, \tag{38}$$

where $u^{(15)}$ refers to the 15th spatial derivative. The results are presented in Table 1.

The second library extends this to the 20th-degree polynomial and the 20th spatial derivative:

$$\mathcal{L}_2 = \{1, u, u^2, \ldots, u^{20}, u_x, uu_x, \ldots, u^{19}u_x, u_{xx}, u_{xxx}, \ldots, u^{(20)}\}, \tag{39}$$

and its results are shown in Table 2.

Both experiments were performed on noise-free data, and the domain size was set to $L = 22$, where the KS system is in a chaotic state. The results show that there are significant differences between the methods when they have to be able to discover the correct model from a very large number of candidates. Although each method is able to infer the correct terms of the KS equation with commendable accuracy (with an error of up to 1%), significant differences arise in the number of incorrect terms.

The strong-form methods RIMSI, PDE-FIND, and PDE-FIND with backvoting produce models that are very close to the correct equation. Of these, PDE-FIND and PDE-FIND with backvoting do not include any incorrect terms. Although RIMSI includes one incorrect higher-order derivative term ($u_{xxxxxxxxx}$ for $\mathcal{L}_1$ and $u_{xxxxxx}$ for $\mathcal{L}_2$), the coefficients of these extra terms are negligible (for example, about $8.1 \cdot 10^{-8}$ for $\mathcal{L}_1$).

The weak-form methods WSINDy and E-WSINDy have difficulties in inferring correct models from such a large set of candidates. With $\mathcal{L}_1$ WSINDy picks up 10 incorrect terms, and with $\mathcal{L}_2$ the number increases to 32 incorrect terms. Even with the correct terms included, WSINDy fails to find a

parsimonious model. E-WSINDy is more efficient in finding a simpler model. With the library $\mathcal{L}_1$, it only includes the correct terms, with a smaller error than PDE-FIND, for instance. However, with the library $\mathcal{L}_2$, it picks up eight incorrect terms, although their coefficients are quite small.

| Method | Discovered equation | Error % |
|---|---|---|
| WSINDy | $u_t = 0.001u^3 - 0.000u^5 + 0.000u^9$ $- 0.000u^{11} + 0.000u^{15} - 1.001uu_x$ $+ 0.003u^5u_x - 0.003u^7u_x + 0.001u^9u_x$ $- 0.000u^{11}u_x + 0.000u^{13}u_x - 1.000u_{xx}$ $- 1.000u_{xxxx}$ | (0, 0, 0) |
| E-WSINDy | $u_t = -1.001uu_x - 1.000u_{xx} - 1.000u_{xxxx}$ | (0, 0, 0) |
| RIMSI | $u_t = -1.002uu_x - 0.999u_{xx} - 0.999u_{xxxx}$ $+ 0.000u_{xxxxxxxx}$ | (0, 0, 0) |
| PDE-FIND | $u_t = -0.998uu_x - 0.994u_{xx} - 0.995u_{xxxx}$ | (0, 1, 1) |
| PDE-FIND+BV | $u_t = -0.998uu_x - 0.994u_{xx} - 0.995u_{xxxx}$ | (0, 1, 1) |

**Table 1:** Kuramoto-Sivashinsky equation with noise-free data with library size $P = D = 15$. Discovered models and relative coefficient errors for each true term, $|(\hat{w}_i - w_i)/w_i|$ as a percentage ("inf" indicates a missed true term).

| Method | Discovered equation | Error % |
|---|---|---|
| WSINDy | $u_t = 0.001u - 0.001u^3 + 0.002u^4$<br>$+ 0.001u^5 - 0.006u^6 - 0.000u^7$<br>$+ 0.007u^8 + 0.000u^9 - 0.004u^{10}$<br>$+ 0.001u^{12} - 0.000u^{14} + 0.000u^{16}$<br>$- 0.000u^{17} - 0.000u^{18} + 0.000u^{20}$<br>$- 1.001uu_x - 0.002u^2u_x + 0.004u^3u_x$<br>$+ 0.010u^4u_x - 0.003u^5u_x - 0.009u^6u_x$<br>$+ 0.000u^7u_x + 0.003u^8u_x - 0.000u^{10}u_x$<br>$+ 0.000u^{16}u_x - 0.000u^{17}u_x - 0.000u^{18}u_x$<br>$- 1.000u_{xx} + 0.001u_{xxx} - 1.000u_{xxxx}$<br>$+ 0.003u_{xxxxx} + 0.002u_{xxxxxxx} + 0.001u_{xxxxxxxxx}$<br>$+ 0.000u_{xxxxxxxxxx} + 0.000u_{xxxxxxxxxxxx}$ | (0, 0, 0) |
| E-WSINDy | $u_t = -0.000u^2 + 0.000u^3 + 0.000u^4$<br>$- 0.000u^5 - 0.000u^6 + 0.000u^7$<br>$+ 0.000u^8 - 1.000uu_x - 0.001u^3u_x$<br>$- 1.000u_{xx} - 1.000u_{xxxx}$ | (0, 0, 0) |
| RIMSI | $u_t = -1.003uu_x - 0.999u_{xx} - 1.000u_{xxxx}$<br>$- 0.000u_{xxxxxx}$ | (0, 0, 0) |
| PDE-FIND | $u_t = -0.998uu_x - 0.994u_{xx} - 0.995u_{xxxx}$ | (0, 1, 1) |
| PDE-FIND+BV | $u_t = -0.998uu_x - 0.994u_{xx} - 0.995u_{xxxx}$ | (0, 1, 1) |

**Table 2:** Kuramoto-Sivashinsky equation with noise-free data with library size $P = D = 20$. Discovered models and relative coefficient errors for each true term, $|(\hat{w}_i - w_i)/w_i|$ as a percentage ("inf" indicates a missed true term).

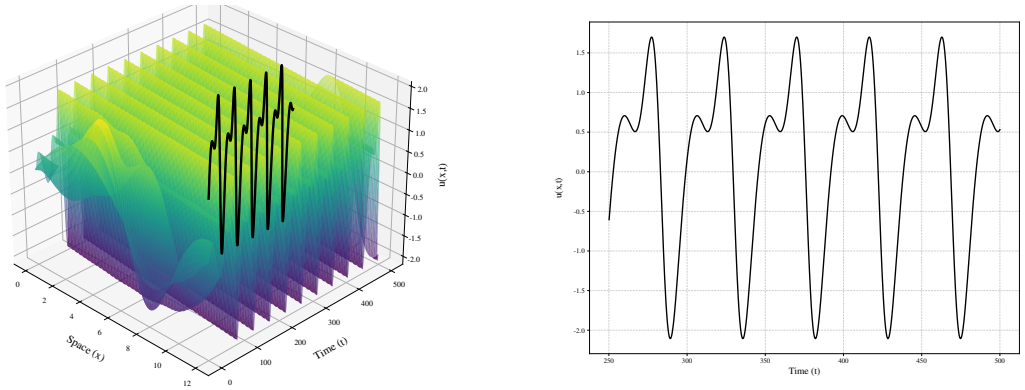
Regarding inference times of the methods, the weak-form methods WSINDy and E-WSINDy are significantly faster than the strong-form methods RIMSI and PDE-FIND, as can be seen from Table 3. When the size of the term library is increased from $P = D = 15$ to $P = D = 20$, the inference time of all methods increases. The slowest is PDE-FIND, with which the computation time jumps to over 300 seconds, while WSINDy completes the same task in less than 20 seconds. E-WSINDy brings only a small time increase compared to the basic model, which makes it a very computationally cost-effective alternative with noise-free data. Although RIMSI is faster than PDE-FIND, it still falls far short of the performance of WSINDy. It is also worth noting that the optional backward voting (BV) that refines the results of PDE-FIND only increases the inference time nominally.

| Method | Inference time (s) ($P = D = 15$) | Inference time (s) ($P = D = 20$) |
|---|---|---|
| WSINDy | 14.23 | 19.77 |
| E-WSINDy | 18.33 | 25.31 |
| RIMSI | 133.38 | 219.59 |
| PDE-FIND | 221.40 | 307.32 |
| PDE-FIND+BV | 222.39 | 308.49 |

**Table 3:** Inference times for each method using candidate libraries of size $P = D = 15$ and $P = D = 20$ on noise-free data.

### 4.2.2 Travelling wave inference on noise-free data

The Kuramoto-Sivashinsky equation exhibits different regimes depending on the domain size $L$ and the initial condition used to generate data. Although with a large value of $L$, the KS equation results in a chaotic system, the domains with a low value of $L$ are interesting, as well. When $L < 13$, a stable traveling wave emerges [39] as Figure 3 demonstrates for $L = 12$. In this section, we increase the simulation time to $T = 500$ to allow us to see the shape of the wave clearly.

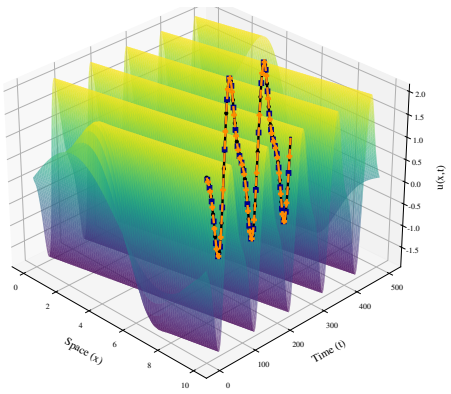



**(a)** Data produced by the KS equation and an example trajectory (in black) of the middle spatial index.

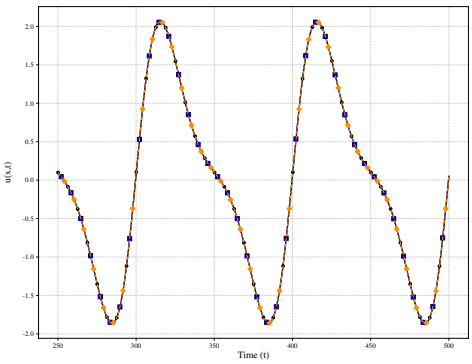
**(b)** Time evolution of the middle spatial index in the stationary period, starting from $t = 250$.

**Figure 3:** Travelling wave regime of the KS equation with domain size $L = 12$ with transient period included.

The travelling wave, with its dynamical simplicity, allows discovery methods to resort to a simpler model than the KS equation. Consider the $L$ values that give rise to a travelling wave. For a lower value, such as $L = 10$, even the best-performing method we have investigated, WSINDy, infers the wave equation $u_t = cu_x$, where $c$ is the wave speed. This is shown in Figure 4 with the transient period included, where both inferred models are simple, but nevertheless follow the ground truth very closely.



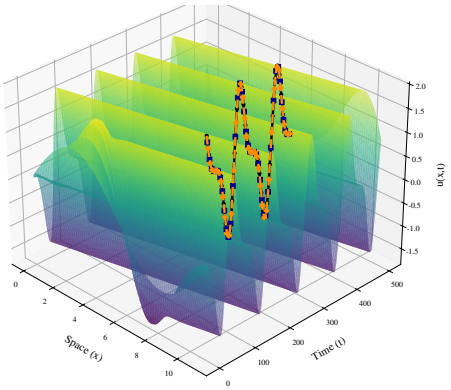

**(a)** Data produced by the KS equation. An example trajectory (in black), and the trajectories of the inferred models RIMSI (in blue) and WSINDy (in orange) of the middle spatial index.



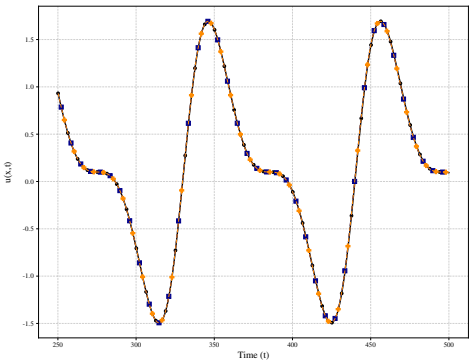

**(b)** Time evolution of the middle spatial index of the KS equation in the stationary period, starting from $t = 250$ (in black), and the trajectories of the inferred models produced by RIMSI ($u_t = 0.100u_x$, in blue) and WSINDy ($u_t = 0.100u_x$, in orange).

**Figure 4:** Travelling wave regime of the KS equation with domain size $L = 10$ with transient period included.

In comparison, for a higher $L$ value, WSINDy may yield the correct KS equation, especially if the transient period of the data simulation is included. This is depicted in Figure 5 for $L = 11$. Because the transient period is distinct from the stationary travelling wave, the correct KS equation terms provide a better fit to the data than the simple wave equation. However, including the transient is insufficient for RIMSI, and it still reverts to the wave equation at this domain size.



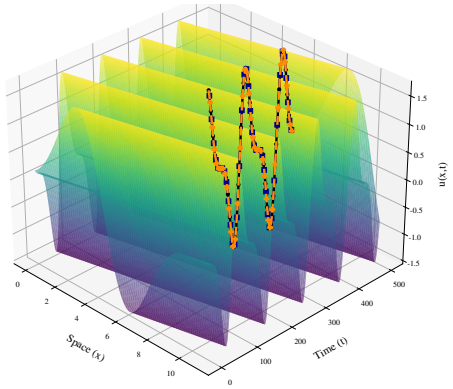

**(a)** Data produced by the KS equation. An example trajectory (in black), and the trajectories of the inferred models RIMSI (in blue) and WSINDy (in orange) of the middle spatial index.



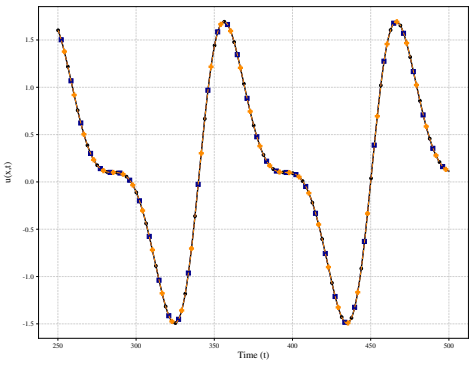

**(b)** Time evolution of the middle spatial index of the KS equation in the stationary period, starting from $t = 250$ (in black), and the trajectories of the inferred models produced by RIMSI ($u_t = 0.100u_x$, in blue) and WSINDy ($u_t = -1.00uu_x - 0.999u_{xx} - 0.999u_{xxxx}$, in orange).

**Figure 5:** Travelling wave regime of the KS equation with domain size $L = 11$ with transient period included.

On the other hand, when the transient period is excluded for $L = 11$, both methods infer the wave equation, which is shown in Figure 6. As the waveform is not very intricate and there is no transient period that would provide additional information about the underlying dynamics, WSINDy is unable to infer the KS equation.




**(a)** Data produced by the KS equation. An example trajectory (in black), and the trajectories of the inferred models RIMSI (in blue) and WSINDy (in orange) of the middle spatial index.




**(b)** Time evolution of the middle spatial index of the KS equation in the stationary period, starting from $t = 250$ (in black), and the trajectories of the inferred models produced by RIMSI ($u_t = 0.100u_x$, in blue) and WSINDy ($u_t = 0.100u_x$, in orange)

**Figure 6:** Travelling wave regime of the KS equation with domain size $L = 11$ with transient period excluded.

To emphasise, although the wave equation is incorrect in the sense that it is not the same as the actual dynamics that produce the data, it mimics the data very closely in the stationary period. This can be seen, for example, from Figure 6, in which both methods infer the wave equation, but nevertheless follow the ground truth accurately.

Added noise, moreover, affects whether the correct underlying equation is found. As with model discovery in general, the higher the noise level, the more likely a simple, incorrect model will be produced. This is especially true in the travelling-wave regime because the wave equation is not only simple but also follows the dynamics closely. Adding even a small amount of noise causes even the weak-form methods to resort to the wave equation, as will be shown in Section 4.3.6 for $L = 12$.

## 4.3 Systems under observational noise

### 4.3.1 Excluding PDE-FIND from the comparison

In the following results, it was decided to exclude the PDE-FIND method. This decision is supported by the method's considerable sensitivity to noise compared to weak-form methods. Since weak-form methods WSINDy and E-WSINDy and the more recent strong-form RIMSI algorithm already provide an interesting comparison set, the poor results of PDE-FIND do not bring significant added value to the work.

The shortcomings of the method become apparent when examining the discovered equations with medium-level noise. To justify the exclusion, Tables 4 and 5 present the PDEs found by the methods for noise levels $\sigma = 10^{-4}$ and $\sigma = 10^{-3}$ for spatial domain size $L = 22$.

| Method | Discovered equation | Error % |
|---|---|---|
| WSINDy | $u_t = -1.001uu_x - 1.001u_{xx} - 1.001u_{xxxx}$ | (0, 0, 0) |
| E-WSINDy | $u_t = -1.001uu_x - 1.001u_{xx} - 1.001u_{xxxx}$ | (0, 0, 0) |
| RIMSI | $u_t = -0.592uu_x - 0.561u_{xx} - 0.579u_{xxxx}$ | (41, 44, 42) |
| PDE-FIND | $u_t = -0.614uu_x - 0.497u_{xx} - 0.576u_{xxxx}$ $+ 0.024u^3 - 0.002u^5$ | (39, 50, 42) |
| PDE-FIND+BV | $u_t = -0.605uu_x - 0.573u_{xx} - 0.592u_{xxxx}$ | (40, 43, 41) |

**Table 4:** Discovered models at noise level $\sigma = 10^{-4}$ and relative coefficient errors for each true term, $|(\hat{w}_i - w_i)/w_i|$ as percentage ("inf" indicates a missed true term).

At a noise level of $10^{-4}$ (Table 4), WSINDy-based methods find the exact structure of the dynamics with coefficients of excellent accuracy. PDE-FIND, on the other hand, picks up physically irrelevant and incorrect terms $u^3$ and $u^5$ into the model. Although the backvoting pruning manages to clean up the wrong terms, the coefficients of the remaining real derivatives are reduced to almost half of the true values.

| Method | Discovered equation | Error % |
|---|---|---|
| WSINDy | $u_t = -1.001uu_x - 1.000u_{xx} - 1.001u_{xxxx}$ | (0, 0, 0) |
| E-WSINDy | $u_t = -1.001uu_x - 1.000u_{xx} - 1.001u_{xxxx}$ | (0, 0, 0) |
| RIMSI | $u_t = -0.064uu_x$ | (94, inf, inf) |
| PDE-FIND | $u_t = -0.148uu_x + 0.176u_{xx} - 0.010u_{xxxx}$ $+ 0.055u + 0.034u^3 - 0.003u^5$ | (85, 118, 99) |
| PDE-FIND+BV | $u_t = -0.132uu_x + 0.168u_{xx} + 0.108u$ | (87, 117, inf) |

**Table 5:** Discovered models at noise level $\sigma = 10^{-3}$ and relative coefficient errors for each true term, $|(\hat{w}_i - w_i)/w_i|$ as percentage ("inf" indicates a missed true term).

When the noise is increased to the level of $10^{-3}$ (Table 5), the performance decreases even more. At this point, the equation structure given by PDE-FIND collapses. The algorithm suggests a model in which the signs and values of the coefficients are entirely incorrect. For example, the coefficient of the diffusion term $u_{xx}$ has an incorrect positive value (+0.1764).

It is true that RIMSI also fails to produce reasonable models at these noise levels. However, RIMSI is included in the following results because it serves as a fresh benchmark for how strong-form methods perform on noisy data.

#### 4.3.2 Moderately chaotic system with normal candidate library

Figure 7 compares the weak-form WSINDy and E-WSINDy methods, the strong-form RIMSI algorithm, and the effect of SG prefiltering on system identification. At low noise levels ($10^{-8} \leq \sigma \leq 10^{-4}$), the weak-form methods perform the task practically flawlessly. WSINDy and E-WSINDy identify the correct equation structure and are able to estimate the coefficients with very high accuracy. The ability of these models to predict the future state of the system is excellent, and the valid times are almost 140.

The RIMSI algorithm, on the other hand, starts to show signs of poor performance even with the cleanest data. Although the algorithm identifies correct terms at the level of $10^{-8}$, the coefficients it

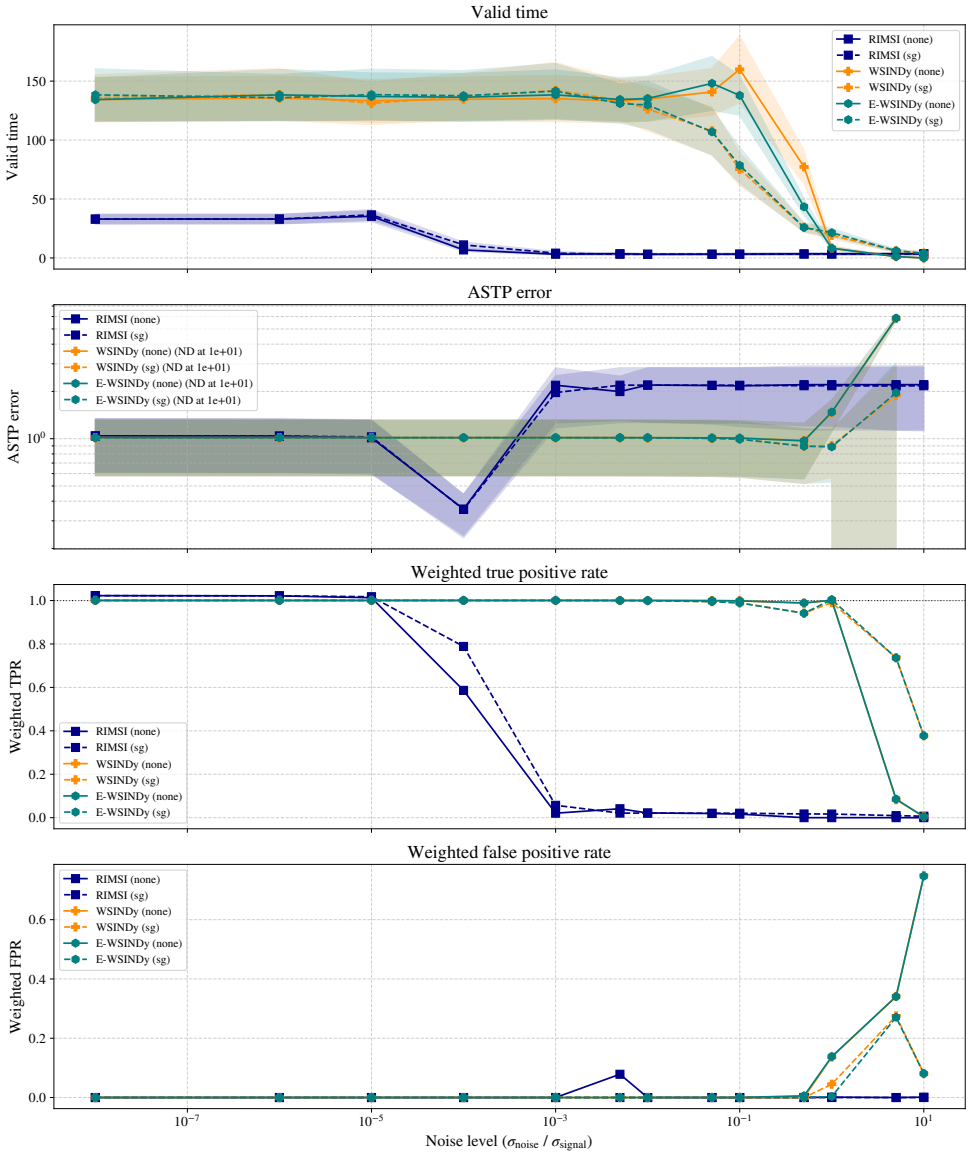



**Figure 7:** Models inferred from Kuramoto-Sivashinsky data of domain size $L = 22$. "sg" denotes results when Savitzky-Golay denoising is used. For Valid time and ASTP error, 95% confidence intervals are provided. In the ASTP error plot, ND denotes numerical divergence at a noise level.

estimates (for example, $-1.037$ for the term $u_{xxxx}$) deviate from the true values much more than with weak-form methods. As a result, the valid time of the model is significantly lower, around 30. As the noise level increases to the relatively small value of $10^{-4}$, the performance of RIMSI really starts to collapse. At this noise level, the algorithm produces a poor model $u_t = -0.602uu_x - 0.569u_{xx} - 0.588u_{xxxx}$, with a valid time of only less than 7.

The performance of the strong-form method degrades significantly at $10^{-3}$ noise, where RIMSI drops the spatial derivatives $u_{xx}$ and $u_{xxxx}$ from the model, and the result is a completely incorrect model of $-0.062uu_x$. From a physical point of view, this means that the model lacks the energy dissipation, which is characteristic of the KS equation. As a result, the model's valid time is close to zero, and the ASTP error increases. This reflects a well-known limitation of strong-form methods: numerical estimation of high derivatives (especially fourth-order $u_{xxxx}$) from noisy data is extremely error-prone. In contrast, the weak-form technique based on integration by parts utilised by WSINDy methods transfers the derivatives to the test functions, which explains the methods' superior noise tolerance.

WSINDy and E-WSINDy, on the other hand, require a much higher noise level when they start to lose their grip on the original dynamics. At the level of $10^{-1}$, the valid times for the unfiltered data even reach their peak (WSINDy reaches almost 160, and E-WSINDy is nearly 140). However, at extreme noise levels, even these algorithms start to interpret the noise as part of the real signal. At a noise level of $\sigma = 1.0$ (without denoising), both WSINDy and E-WSINDy produce a 12-term model, which includes multiple incorrect terms such as $u^2$, $u^3$, $u^4$, and $u_x^5$. This overfitting significantly decreases the predictive power of the models, and the valid times drop to about eight.

Under high noise conditions, RIMSI consistently produces underfitted models. The algorithm heavily penalises erroneous terms, which makes it, at high noise levels, often choose only a constant term or one incorrect term with a very small coefficient (for example, at noise 1.0, the algorithm chooses only the constant $-0.0039$). Both overfitting and underfitting destroy the predictive power, and from noise level 1.0 upwards, the valid times of all unfiltered models are very short.

The results also show that the use of SG denoising in data preprocessing offers somewhat promising results. At moderate noise levels, such as $10^{-1}$, SG filtering actually degrades the results of the weak-form methods significantly since the valid times of WSINDy and E-WSINDy drop to about 75.

However, at the extreme noise level of 1.0, SG denoising shows its power. While unfiltered WSINDy and E-WSINDy lose their ability to model the system, the models inferred from SG-filtered data are able to keep the simulation stable for a longer period. SG filtering increases the valid time of E-WSINDy to 21.4 and WSINDy to 19.2 (from the valid time of around eight without denoising). The filtered models include fewer erroneous terms and manage to keep the coefficients of the correct terms closer to their true coefficient values. Even at the second highest noise level tested, $\sigma = 5.0$, the SG filtering offers a small advantage since while the valid time of the unfiltered models remains at 1.2, the filtered weak-form models still yield valid times of around six.

In the case of RIMSI, however, SG filtering does not solve the fundamental problems of the algorithm. Although filtering may slightly improve the coefficients at certain noise levels, pre-filtering the data alone is not enough to save the strong-form algorithm when faced with high-order spatial derivatives and their noisy approximations, as in the KS equation.

### 4.3.3 Strongly chaotic system with normal candidate library

Figure 8 shows the performance of the weak-form WSINDy and E-WSINDy methods and the strong-form RIMSI method in identifying the structure of the KS equation in a larger ($L = 36$) domain. Increasing the domain size makes the dynamics of the system more complex and chaotic as discussed in Section 2.1.1. The increased chaos is reflected in the results since the predictive ability of all models is weakened. While in the $L = 22$ case the best valid times exceeded 140, in the larger domain, even the best weak-form models only reach valid time values of slightly over 90.

At the very lowest noise levels ($10^{-8} \leq \sigma \leq 10^{-4}$), the relative differences between the methods follow the same pattern as with $L = 22$. The weak-form methods WSINDy and E-WSINDy recognise the structure of the equation excellently and achieve valid times of about 85. RIMSI, on the other hand, has serious difficulties even with the cleanest data. Albeit with small coefficients, the algorithm constantly picks up incorrect terms, such as $u$ and $u^3$. At the same time, the coefficients of the correct

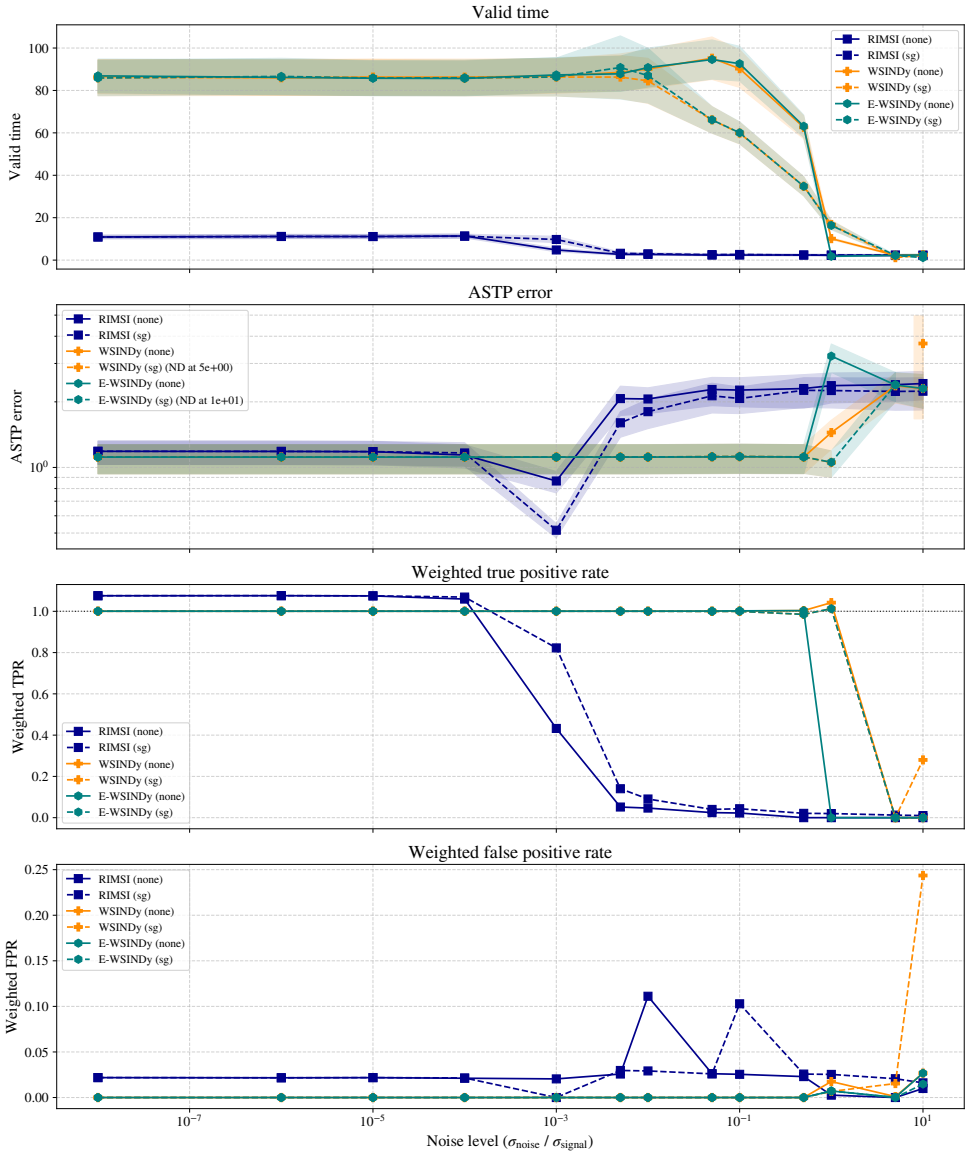



**Figure 8:** Models inferred from Kuramoto-Sivashinsky data of domain size $L = 36$. "sg" denotes results when Savitzky-Golay denoising is used. For Valid time and ASTP error, 95% confidence intervals are provided. In the ASTP error plot, ND denotes numerical divergence at a noise level.

terms differ from the true values. Due to this incorrect structure, even the best valid times of RIMSI models remain at 11.

As noise increases, the fragility of the strong-form method only increases. At the level of $10^{-3}$, RIMSI's valid time drops to four, and the algorithm loses practically all of its ability to model dynamics. The weak-form methods, on the other hand, tolerate this noise very well and keep their valid times around 85 and even slightly improve their results at a noise level of 0.1, where the valid times are around 90.

Between the noise levels 0.1 and 1, the weak-form methods start to lose their grip on the dynamics. At a noise level of 0.5 (without denoising), the valid times of WSINDy and E-WSINDy drop sharply to just above 60. At level 1.0, the models collapse for good: WSINDy reaches a valid time of only 10, and E-WSINDy crashes almost immediately with a valid time of two.

The benefit of SG filtering appears to be twofold. At moderate noise ($10^{-1}$), denoising actually produces worse results. It drops the valid times of weak-form methods from over 90 to under 60. However, in the case of extreme noise, the stabilising effect of SG filtering becomes evident. At noise level 1.0, it prevents the collapse of weak-form models, yielding valid times of around 15. For RIMSI, SG filtering shows slight improvements for ASTP error at multiple noise levels. For valid time, however, the results improve somewhat only at noise level $10^{-3}$.

#### 4.3.4 Moderately chaotic system with larger candidate library

A premise of ensemble SINDy is that it can, in the case of large candidate libraries, make the library sparser [30]. This allows fewer terms to be considered by the methods, and thus possibly limits the number of wrong terms in the models. The previous results of Figures 7 and 8 indicate that E-WSINDy is on a par with WSINDy with only small differences. Hence, to test the premise, we investigate the methods with a larger candidate library that contains terms up to the 10th polynomial power and spatial derivatives up to order 10. Similarly to the results above, we investigate RIMSI, WSINDy, and E-WSINDy with and without SG denoising. The results are depicted in Figure 9.

At low noise levels ($10^{-8} \leq \sigma \leq 10^{-3}$), the weak-form methods are nevertheless highly robust. Despite the large number of incorrect terms to select from, WSINDy and E-WSINDy manage to find only the correct terms of the KS equation. As a result, their ability to follow the system's trajectory is excellent, and the valid times are around 125.

The strong-form RIMSI algorithm performs well on the cleanest data ($10^{-8}$). Despite a large library of 30 candidate terms, it is able to eliminate all erroneous terms and select exactly the correct equation structure. The lower valid time than with the weak-form models is thus explained by deviating coefficients, as in Section 4.3.2, which causes the true and predicted trajectories to diverge earlier. As the noise level increases to $10^{-3}$, the algorithm's performance breaks down: RIMSI drops everything except the term $-0.062uu_x$ from the equation. As a result, the model collapses and the valid time drops close to three.

As the noise increases to the moderate level ($10^{-2} \leq \sigma \leq 10^{-1}$), the problems introduced by the large library start to degrade the weak-form methods. At the level $5 \cdot 10^{-2}$, a surprising phenomenon is observed, where E-WSINDy performs worse than traditional WSINDy. This is due to the failure of E-WSINDy at that point: while traditional WSINDy finds a 7-term model (with $u^3u_x, u^5u_x, u^7u_x$ and $u^9u_x$ in addition to the correct terms), E-WSINDy overfits the model to include 17 terms.

However, as the noise level increases slightly to $10^{-1}$, we get a converse result. Unfiltered WSINDy finds a similar 7-term model again, and its valid time drops to about 80. Here, E-WSINDy shows its true potential and works as intended. It manages to filter out all the incorrect terms, resulting in the 3-term KS equation, with very accurate coefficients. The ability to eliminate the incorrect terms leads to a high valid time of 117 at this noise level.

At 0.5 noise, WSINDy fed with unfiltered data loses its ability to distinguish signal from noise and selects 24 terms out of 30 possible for the model. The same collapse happens to E-WSINDy also, as its model contains 23 terms.

The role of SG filtering becomes different with a large library. As with a smaller library, SG filtering has a disadvantage with moderate noise. For example, at 0.1 noise, it significantly decreases the valid times of WSINDy and E-WSINDy. At extreme noise levels, moreover, pre-filtering no longer saves the models from a total collapse. With a smaller library, SG filtering managed to stabilise the

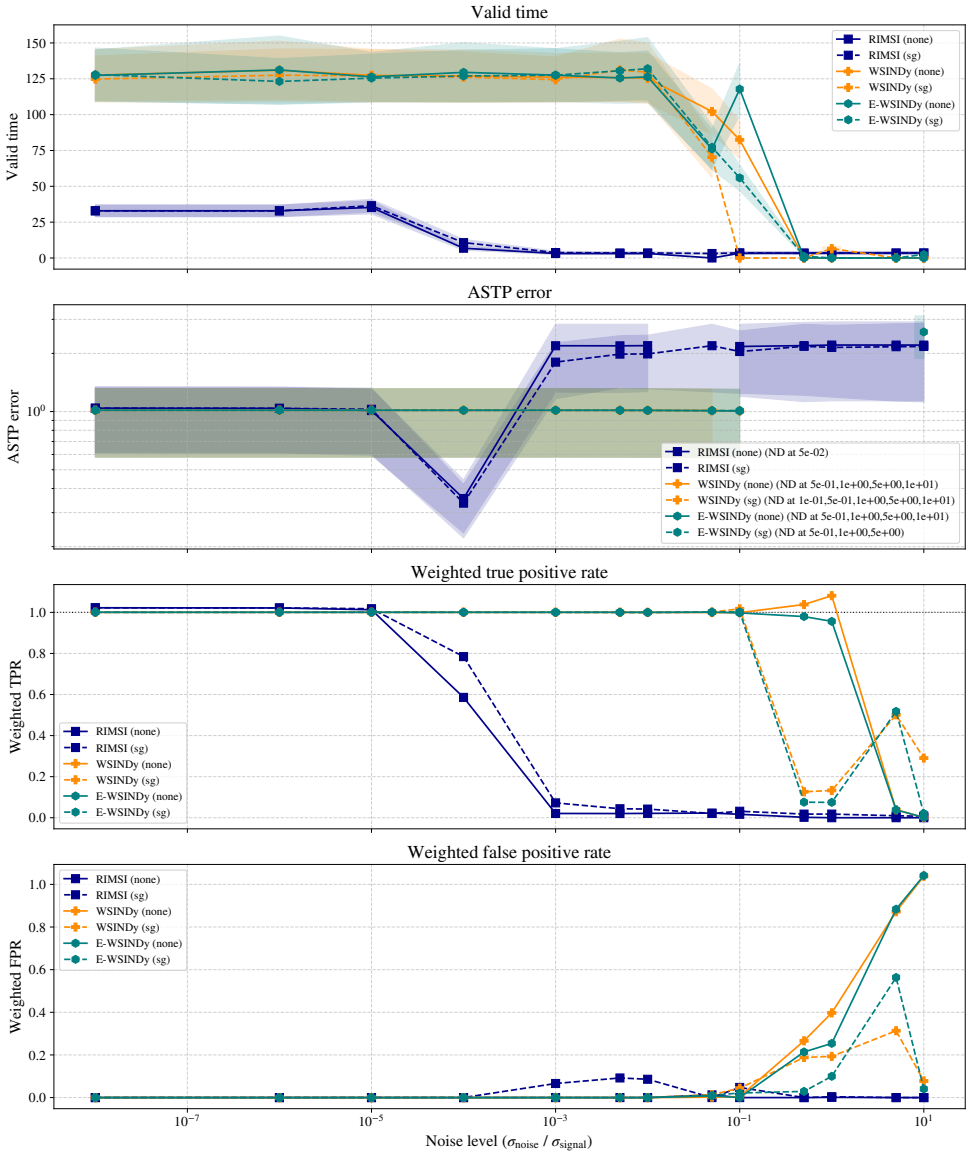



**Figure 9:** Models inferred from Kuramoto-Sivashinsky data of domain size $L = 22$ with a large candidate library. "sg" denotes results when Savitzky-Golay denoising is used. For Valid time and ASTP error, 95% confidence intervals are provided. In the ASTP error plot, ND denotes numerical divergence at a noise level.

models and produce non-zero valid times. However, the same does not apply here. Both WSINDy and E-WSINDy overfit with many incorrect terms. Because the resulting models are completely incorrect, the valid times invariably collapse to zero.

#### 4.3.5 ASTP error anomaly of the RIMSI algorithm

When examining the ASTP error, an anomaly is observed in the RIMSI algorithm, as the error drops visibly at a noise level of $10^{-4}$ for domain size $L = 22$ (Figures 7 and 9) and at a noise level of $10^{-3}$ for $L = 36$ (Figure 8). Interestingly, this seems to occur in both cases at the same noise level where the weighted true positive rate starts to collapse.

At lower noise levels, the ASTP error of RIMSI remains stable. In the case $L = 22$ with the smaller library, when the noise reaches the level $10^{-4}$, however, the error unexpectedly drops to about 0.35. A corresponding, slightly smaller drop occurs for $L = 36$ at a noise level $10^{-3}$. As the noise increases further from this to the next level, the ASTP error rises above two for both.

The reasons for this, and especially for the temporary improvement of the error, can be found in the structure of the equations identified by RIMSI. When the noise is low, the algorithm finds the correct terms in the equation with values close to the true ones. Although the model is structurally close to the KS equation, even small differences accumulate during a small number of time steps due to the equation's chaotic behaviour. Coefficients that are fairly accurate, but vary slightly from the correct values, may produce large changes in the time derivative $u_t$, thus resulting in a relatively high ASTP error. The same holds for the more precise weak-form methods as well: even though their coefficient values are even more accurate than for RIMSI in the very low noise region, their ASTP error still remains at the same level as RIMSI's.

At the turning point of $10^{-4}$ or $10^{-3}$, RIMSI starts to suffer from noise. Although RIMSI still identifies the correct terms $uu_x$, $u_{xx}$ and $u_{xxxx}$, it significantly shrinks the values of the coefficients (for example, from $-1.0$ to $\approx -0.6$ for $L = 22$ in the unfiltered case). Thus, since the model predicts more damped motion than the actual one, the difference between the predicted and the actual state during the first time steps, as measured by the ASTP error, remains smaller. This results in a decrease in the ASTP error. At the same time, naturally, the underlying KS equation is not inferred as well as at lower noise levels, which is revealed when the models' valid time is examined. Although the ASTP error is at its lowest at a noise level of $10^{-4}$ or $10^{-3}$, the valid time of the model produced by RIMSI decreases. This shows that while smaller coefficients minimise the ASTP error, they do not account for the actual dynamics and worsen the model's overall performance.

When the noise is increased even more (for example, to the level $10^{-3}$ with $L = 22$), the algorithm no longer manages to find even the basic structure of the KS equation, but may return just a single incorrect term such as $-0.06uu_x$ in the unfiltered case for $L = 22$ with the smaller library. At this point, the model is structurally incorrect. This results in the models predicting that the state develops in a completely wrong direction already during a small number of steps, which raises the ASTP error to its highest levels.

#### 4.3.6 Travelling wave regime with normal candidate library

As was discussed in Section 4.2.2 with noise-free data, the dynamic simplicity of a travelling wave combined with the absence of a transient phase may drive the methods to infer a simple wave equation. When investigating the methods in a domain size of $L = 12$ (without a transient period), increasing the noise intensity reveals a trend in the structure of the models the methods infer. The results are depicted in Figure 10.

In the low noise region ($10^{-8} \leq \sigma \leq 10^{-6}$), the weak-form methods WSINDy and E-WSINDy perform commendably. They are able to find the KS equation even in this dynamically simple regime. The valid times of these models reach the maximum evaluation time of 500. RIMSI, on the other hand, is unable to construct the KS equation and infers the wave equation $u_t = cu_x$ with a coefficient of $c \approx 0.258$, resulting in lower valid times than the weak-form methods of around 100.

As the results verify, finding the KS equation in this regime is challenging when the noise level increases. At a noise level of $10^{-5}$, E-WSINDy still manages to discover the correct equation structure, but traditional WSINDy resorts to the wave equation, which reduces its valid time to around 100. As

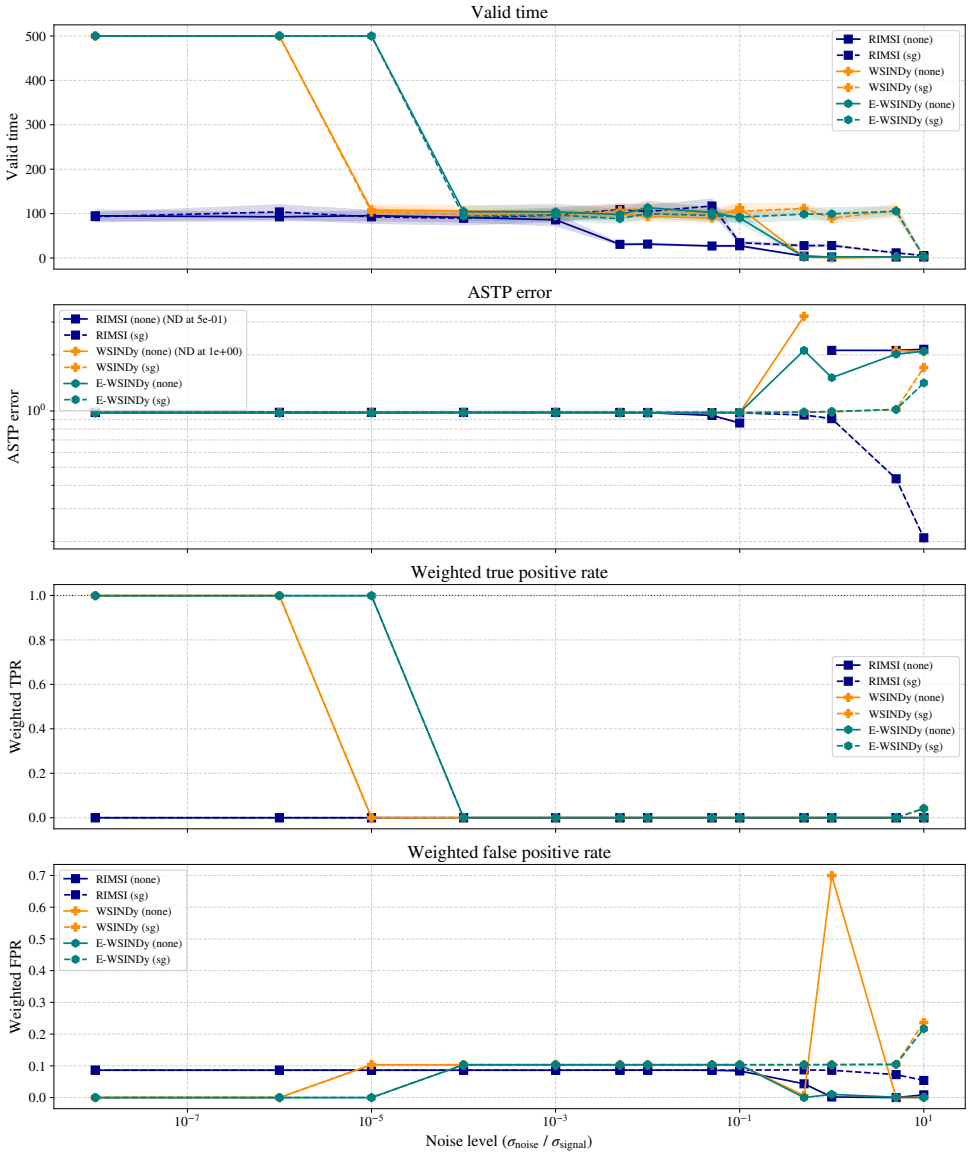



**Figure 10:** Models inferred from Kuramoto-Sivashinsky data of domain size $L = 12$. "sg" denotes results when Savitzky-Golay denoising is used. For Valid time and ASTP error, 95% confidence intervals are provided. In the ASTP error plot, ND denotes numerical divergence at a noise level. The maximum evaluation time of Valid time is capped at 500.

the noise level increases a bit more, to the level of $10^{-4}$, E-WSINDy also abandons the KS equation for the simpler model.

However, resorting to the wave equation is not surprising. It is a direct consequence of the simple dynamics of the observed data. Since the solution with this attractor takes the form $u(x,t) = v(x-ct)$, the wave equation, it makes perfect sense for the methods to infer a simple model. As the methods promote sparsity, the algorithms choose the most optimal sparse model based on the data.

When moving to extreme noise levels ($\sigma \geq 0.5$), unfiltered WSINDy begins to overfit. At noise level 0.5, it picks up an unphysical term $5u^4u_x$, and at level 1.0, it infers a six-term equation (including $u^2$, $u^3$, $u_{xxx}$, and $u_{xxxxx}$ incorrectly). This results in numerical instabilities, and the valid time drops to zero starting from 0.5 noise.

E-WSINDy tries to control the overfitting, but at level 0.5, it ends up returning a completely empty, zero-coefficient model. At level 1.0, it restricts the model to a single incorrect term ($3u^2u_x$). RIMSI, moreover, shows its tendency to underfit again. At level 1.0, it only includes a near-zero constant term ($-0.0057$). Its ASTP error is reasonable, 2.11, because the model predicts a practically static, flat state and thus does not diverge. However, the model is physically inferior, as its zero valid time indicates.

In this regime, using SG denoising is more justified than in the chaotic state. In chaotic regimes, SG filtering could degrade the results. However, the data for a travelling wave attractor is nowhere near as complex, and the algorithms look for a clean waveform. Therefore, data denoising is beneficial even in the presence of extreme noise. For example, in very high 1.0 noise, when the raw data produces numerical collapses, SG filtering rescues the models. Both WSINDy and E-WSINDy find the wave equation ($u_t \approx 0.259u_x$), which allows their valid times to achieve almost 100. RIMSI also benefits slightly from this. At 1.0 noise, it contains the advection term $u_x$ with a coefficient (0.255) close to the speed of the wave, although it also includes two extra terms with small coefficients. In the dynamic simplicity of travelling waves, smoothing of the data is therefore a critical step in rescuing the signal from heavy noise.

### 4.3.7 Deterministic error growth on the traveling wave attractor

In our evaluation of the KS equation on the spatial domain $L = 12$, the plotted confidence intervals for both the valid time and ASTP error metrics are not discernible for several noise levels. This is a direct consequence of the stationary non-chaotic state of the ground truth and the shape of the inferred models.

On sufficiently small spatial domains, the KS system does not exhibit spatiotemporal chaos. Instead, after a brief transient period, it collapses onto a stable travelling wave attractor. The ground truth, therefore, consists only of a waveform moving at a constant velocity. As the methods promote sparsity, they often discard the true, complex PDE structure $-uu_x - u_{xx} - u_{xxxx}$ and instead fit a simple advection model $u_t = cu_x$.

During metric calculation, we integrate the inferred model to compare it against the ground truth. Since both describe waves moving at constant, but slightly mismatched, speeds, the physical phase shift between the two trajectories grows at a constant rate. As a result, because the valid time and ASTP error simply track the distance between the state vectors of the true data and inferred model at each time point, both metrics always yield the same result. The predefined error threshold for valid time is breached at the same time step for every trial, regardless of where on the attractor the initial condition was located. Similarly, the ASTP error will produce the same result for each trial. These will drive the variance across our valid time and ASTP error trials to zero. Consequently, the confidence intervals vanish from the plots.

### 4.3.8 On running times in noisy domains

Tables 6 and 7 and 8 summarize the inference times of the methods at two example noise levels $10^{-5}$ and $10^{-1}$ (without SG denoising) for the results of Sections 4.3.2, 4.3.4 and 4.3.6, respectively. As with the inference times of noise-free data presented in Table 3, the weak-form methods WSINDy and E-WSINDy are significantly faster than the strong-form RIMSI. While WSINDy-based models typically produce results in less than ten seconds, RIMSI consistently requires several minutes of computation.

It is also worth noting that increasing noise does not automatically slow down the solvers. In particular, on the chaotic domain $L = 22$ with a larger term library ($P = D = 10$) (Table 7), the

computation times of WSINDy and E-WSINDy even decrease when the noise is increased. Moreover, although E-WSINDy is based on inferring numerous models through bootstrapping, it increases the inference time only moderately compared to the basic WSINDy.

| Method | Inference time (s) ($\sigma = 10^{-5}$) | Inference time (s) ($\sigma = 10^{-1}$) |
|---|---|---|
| WSINDy | 2.58 | 2.44 |
| E-WSINDy | 4.07 | 5.18 |
| RIMSI | 206.25 | 233.85 |

**Table 6:** Inference times for each method using a candidate library of size $P = D = 5$ for the spatial domain size $L = 22$ at noise levels $\sigma = 10^{-5}$ and $\sigma = 10^{-1}$ (without SG denoising).

| Method | Inference time (s) ($\sigma = 10^{-5}$) | Inference time (s) ($\sigma = 10^{-1}$) |
|---|---|---|
| WSINDy | 9.12 | 3.22 |
| E-WSINDy | 10.46 | 6.23 |
| RIMSI | 286.18 | 264.49 |

**Table 7:** Inference times for each method using a candidate library of size $P = D = 10$ for the spatial domain size $L = 22$ at noise levels $\sigma = 10^{-5}$ and $\sigma = 10^{-1}$ (without SG denoising).

| Method | Inference time (s) ($\sigma = 10^{-5}$) | Inference time (s) ($\sigma = 10^{-1}$) |
|---|---|---|
| WSINDy | 5.79 | 6.65 |
| E-WSINDy | 6.93 | 7.80 |
| RIMSI | 241.64 | 263.88 |

**Table 8:** Inference times for each method using a candidate library of size $P = D = 5$ for the spatial domain size $L = 12$ at noise levels $\sigma = 10^{-5}$ and $\sigma = 10^{-1}$ (without SG denoising).

# 5 Conclusion

This work investigated and compared numerical methods whose objective is to identify the underlying PDE from observational data. The main focus of the study was on evaluating the noise tolerance and system identification accuracy of the algorithms. The test data was produced by the Kuramoto-Sivashinsky equation, to which different levels of random noise were added. The methods under study were divided into two main categories: strong-form methods, RIMSI and PDE-FIND, and weak-form methods, WSINDy and E-WSINDy.

The results clearly show that strong-form methods based on numerical differentiation are very sensitive to noise. The noise range, where these methods are useful, is limited to data with a low level of observational noise (approximately $\sigma_{\text{obs}} \leq 10^{-4}$). At noise levels higher than this, strong-form algorithms either selected terms that were incorrect or underfitted the model by leaving out important terms altogether.

Weak-form methods provided a significant solution to the challenge presented by noise. By moving the derivative operators from the actual observational data to separate, analytical test functions using integration by parts, numerical differentiation is avoided. The weak-form methods investigated were able to reliably discover the correct equation structure of the system even when the magnitude of noise in the data was considerable (up to the level $\sigma_{\text{obs}} \approx 10^{-1}$). E-WSINDy also showed significant robustness in situations where a very large candidate term library was provided. Using ensemble learning, it was able to prevent overfitting more effectively than the basic WSINDy. Moreover, the weak-form methods performed the equation discovery significantly faster than the strong-form algorithms.

The weak-form methods have their own mathematical limitations. The method imposes stricter conditions on the structure of acceptable PDE terms compared to what is allowed by strong-form algorithms, as the terms must be in a form that can be integrated by parts by transferring the derivatives to test functions. Although this in principle limits the possible candidate functions, we emphasise that in the paper introducing the WSINDy method [29], the researchers state that all partial differential equations that were the subject of their research can be represented in the weak form.

Preprocessing the data using SG filtering was found to have twofold effects. At low and moderate noise levels, separate filtering often weakened the results of weak-form algorithms. On the other hand, under extreme noise conditions ($\sigma_{\text{obs}} \geq 1.0$), where the models would otherwise collapse numerically, SG filtering allowed the methods to discover at least partially correct structures. However, for the strong-form RIMSI algorithm, filtering alone was not enough to save the method from the fundamental problems of numerical differentiation.

Furthermore, the work's investigation into travelling wave dynamics highlighted the requirements of data-driven equation discovery. If it is of importance to find out the underlying partial differential equation, and not simply an equation that follows the data closely, the data must have sufficient dynamic complexity. When the dynamics settle into a travelling wave motion, algorithms that enforce sparsity in the model end up with a simpler model that describes only the wave motion.

A natural continuation of this work would be to move from artificially simulated data to the analysis of real, experimental observations. Since the weak-form algorithms studied in this work showed promising noise tolerance, they offer an attractive tool for modelling brain activity, for example. In future, it would be interesting to apply the methods directly to non-invasive brain data, such as EEG or MEG measurements. In this case, the goal could be to identify the partial differential equations that most accurately describe the complex spatiotemporal wave dynamics of the cerebral cortex from actual and possibly very noisy real-world measurements.

## References


[1] L. Edelstein-Keshet. *Mathematical Models in Biology*. 1st edition. SIAM: Society for Industrial and Applied Mathematics, 2005. ISBN: 0898715547.

[2] G. K. Batchelor. *An Introduction to Fluid Dynamics*. 2nd edition. Cambridge University Press, 2000. ISBN: 0521663962.

[3] L. Q. Chen. "Phase-field models for microstructure evolution". In: *Annual Review of Materials Science* 32 (2002), pp. 113–140. DOI: 10.1146/annurev.matsci.32.112001.132041.

[4] A. M. Turing. "The Chemical Basis of Morphogenesis". In: *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 237 (641 1952), pp. 37–72. DOI: 10.1098/rstb.1952.0012.

[5] G. B. Ermentrout and D. H. Terman. *Mathematical Foundations of Neuroscience*. 1st edition. Springer, 2010. ISBN: 038787707X.

[6] J. D. Murray. *Mathematical Biology: I. An Introduction*. 3rd edition. Springer, 2002. ISBN: 0387952233.

[7] G. E. Karniadakis et al. "Physics-informed machine learning". 2021. DOI: 10.1038/s42254-021-00314-5.

[8] J. N. Kutz et al. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. 1st edition. Society for Industrial and Applied Mathematics, 2016. ISBN: 1611974496.

[9] Z. Long et al. "PDE-Net: Learning PDEs from Data". 2018. DOI: 10.48550/arXiv.1710.09668.

[10] J. C. Loiseau and S. L. Brunton. "Constrained sparse Galerkin regression". In: *Journal of Fluid Mechanics* 838 (2018), pp. 42–67. DOI: 10.1017/jfm.2017.823.

[11] J. L. Callaham et al. "Learning dominant physical processes with data-driven balance models". In: *Nature Communications* 12 (1 2021). DOI: 10.1038/s41467-021-21331-z.

[12] L. Zanna and T. Bolton. "Data-Driven Equation Discovery of Ocean Mesoscale Closures". In: *Geophysical Research Letters* 47 (17 2020). DOI: 10.1029/2020GL088376.

[13] E. P. Alves and F. Fiuza. "Data-driven discovery of reduced plasma physics models from fully kinetic simulations". In: *Physical Review Research* 4 (3 2022). DOI: 10.1103/PhysRevResearch.4.033192.

[14] M. Sorokina, S. Sygletos, and S. Turitsyn. "Sparse Identification for Nonlinear Optical Communication Systems: SINO Method". In: *Optics Express* 24 (26 2016). DOI: 10.1364/OE.24.030433.

[15] M. Flaschel, S. Kumar, and L. D. Lorenzis. "Unsupervised discovery of interpretable hyperelastic constitutive laws". In: *Computer Methods in Applied Mechanics and Engineering* 381 (2021). DOI: 10.1016/j.cma.2021.113852.

[16] J. Horrocks and C. T. Bauch. "Algorithmic discovery of dynamic models from infectious disease data". In: *Scientific Reports* 10 (1 2020). DOI: 10.1038/s41598-020-63877-w.

[17] N. M. Mangan et al. "Inferring Biological Networks by Sparse Identification of Nonlinear Dynamics". In: *IEEE Transactions on Molecular, Biological, and Multi-Scale Communications* 2 (1 2016). DOI: 10.1109/TMBMC.2016.2633265.

[18] P. L. Nunez. "The Brain Wave Equation: A Model for the EEG". In: *Mathematical Biosciences* 21 (3–4 1974), pp. 279–297. DOI: 10.1016/0025-5564(74)90020-0.

[19] V. K. Jirsa and H. Haken. "Field Theory of Electromagnetic Brain Activity". In: *Physical Review Letters* 77 (5 1996), pp. 960–963. DOI: 10.1103/PhysRevLett.77.960.

[20] P. A. Robinson, C. J. Rennie, and J. J. Wright. "Propagation and stability of waves of electrical activity in the cerebral cortex". In: *Physical Review E* 56 (1 1997), pp. 826–840. DOI: 10.1103/PhysRevE.56.826.

[21] M. Breakspear. "Dynamic models of large-scale brain activity". In: *Nature Neuroscience* 20 (3 2017), pp. 340–352. DOI: 10.1038/nn.4497.

[22] J. A. Roberts et al. “Metastable brain waves”. In: *Nature Communications* 10 (1 2019). DOI: 10.1038/s41467-019-08999-0.

[23] M. Boda. “Learning Ordinary Differential Equation Models of Nonlinear Dynamics from Noisy Data”. 2026.

[24] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences of the United States of America* 113 (15 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[25] K. Champion et al. “Data-driven discovery of coordinates and governing equations”. In: *Proceedings of the National Academy of Sciences* 116 (45 2019), pp. 22445–22451. DOI: 10.1073/pnas.1906995116.

[26] H. Schaeffer. “Learning partial differential equations via data discovery and sparse optimization”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473 (2197 2017). DOI: 10.1098/rspa.2016.0446.

[27] S. H. Rudy et al. “Data-driven discovery of partial differential equations”. In: *Science Advances* 3 (4 2017). DOI: 10.1126/sciadv.1602614.

[28] S. L. Brunton and J. N. Kutz. “Promising directions of machine learning for partial differential equations”. In: *Nature Computational Science* 4 (7 2024), pp. 483–494. DOI: 10.1038/s43588-024-00643-2.

[29] D. A. Messenger and D. M. Bortz. “Weak SINDy for partial differential equations”. In: *Journal of Computational Physics* 443 (2021). DOI: 10.1016/j.jcp.2021.110525.

[30] U. Fasel et al. “Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 478 (2260 2022). DOI: 10.1098/rspa.2021.0904.

[31] H. Marc. “Robust Interpretable Model for System Identification RIMSI”. 2025. URL: https://urn.fi/URN:NBN:fi:aalto-202508206649.

[32] S. H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. 2nd edition. CRC Press, 2015. ISBN: 0813349109.

[33] P. Holmes et al. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. 2nd edition. Cambridge University Press, 2012. ISBN: 1107008255.

[34] Y. Kuramoto. “Diffusion-Induced Chaos in Reaction Systems”. In: *Supplement of the Progress of Theoretical Physics* (1978), pp. 346–366. DOI: 10.1143/PTPS.64.346.

[35] G. I. Sivashinsky. “Nonlinear analysis of hydrodynamic instability in laminar flames-I. Derivation of basic equations”. In: *Acta Astronautica* 4 (1977), pp. 1177–1206. DOI: 10.1016/0094-5765(77)90096-0.

[36] G. I. Sivashinsky. “On Flame Propagation Under Conditions of Stoichiometry”. In: *SIAM Journal on Applied Mathematics* 39 (1 1980). DOI: 10.1137/0139007.

[37] G. I. Sivashinsky and D. M. Michelson. “On Irregular Wavy Flow of a Liquid Film Down a Vertical Plane”. In: *Progress of Theoretical Physics* 63 (6 1980), pp. 2112–2114. DOI: 10.1143/PTP.63.2112.

[38] Y. Pomeau and S. Zaleski. “The Kuramoto-Sivashinsky equation: A caricature of hydrodynamic turbulence?” In: *Lecture Notes in Physics* 230 (1985), pp. 296–303. DOI: 10.1007/3-540-15644-5_23.

[39] R. A. Edson et al. “Lyapunov Exponents of the Kuramoto-Sivashinsky PDE”. In: *ANZIAM Journal* 61 (3 2019), pp. 270–285. DOI: 10.1017/S1446181119000105.

[40] A. Wolf et al. “Determining Lyapunov exponents from a time series”. In: *Physica D: Nonlinear Phenomena* 16 (3 1985), pp. 285–317. DOI: 10.1016/0167-2789(85)90011-9.

[41] M. Schmidt and H. Lipson. “Distilling Free-Form Natural Laws from Experimental Data”. In: *Science* 324 (5923 2009), pp. 81–85. DOI: 10.1126/science.1165893.

[42] J. Bongard and H. Lipson. “Automated reverse engineering of nonlinear dynamical systems”. In: *PNAS* 104 (24 2007), pp. 9943–9948. DOI: 10.1073/pnas.0609476104.

[43] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. 1st edition. Bradford Books, 1992. ISBN: 9780262111706.

[44] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

[45] R. Chartrand. “Numerical Differentiation of Noisy, Nonsmooth Data”. In: *ISRN Applied Mathematics* 2011 (2011), pp. 1–11. DOI: 10.5402/2011/164564.

[46] I. Knowles and R. Wallace. “A variational method for numerical differentiation”. In: *Numerische Mathematik* 70 (1995), pp. 91–110. DOI: 10.1007/s002110050111.

[47] A. Savitzky and M. J. E. Golay. “Smoothing and Differentiation of Data by Simplified Least Squares Procedures”. In: *Analytical Chemistry* 36 (8 1964), pp. 1627–1639. DOI: 10.1021/ac60214a047.

[48] H. Akaike. “A New Look at the Statistical Model Identification”. In: *IEEE Transactions on Automatic Control* 19 (6 1974), pp. 716–723. DOI: 10.1109/TAC.1974.1100705.

[49] I. Alm. “Data-driven inference of non-linear dynamical systems under observational noise”. 2026.

[50] R. Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society* 58 (1 1996), pp. 267–288. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[51] P. Kärkkäinen and R. Linna. “Dimensional criterion for forecasting nonlinear systems by reservoir computing”. 2022. DOI: 10.48550/arXiv.2202.05159.