# Beyond data subsampling: differentiation as an uncertainty source in equation discovery

**Maria Khilchuk**
AI Institute
ITMO University
Saint-Petersburg, Russia
mdkhilchuk@itmo.ru

**Ilya Markov**
AI Institute
ITMO University
St. Petersburg, Russia, 197101
iomarkov@itmo.ru

**Alexander Hvatov**
AI Institute
ITMO University
Saint-Petersburg, Russia
alex_hvatov@itmo.ru

## Abstract

Data-driven discovery of differential equations typically treats numerical differentiation as a fixed preprocessing step. Existing algorithms improve robustness through data and library subsampling but rarely account for variability in the differentiation method itself. We show that this choice introduces a systematic and reproducible source of uncertainty that alters both the structure of the equation and the coefficients. High-resolution schemes amplify noise, while heavily smoothed derivatives suppress meaningful fluctuations, yielding method-dependent results. We evaluate six differentiation techniques across multiple PDEs and noise levels using SINDy and EPDE, finding consistent shifts in the models discovered. These results establish differentiation method selection as a fundamental modeling decision and a new axis to improve ensemble-based equation discovery.

## 1 Introduction

Four critical components define the framework of any machine learning model: architecture, parameters, features, and the objective function. Similarly, modern approaches to differential equation discovery treat differential equations as machine learning models. This perspective raises key questions about how to assess the quality of the discovered DE and the associated uncertainties, leveraging established evaluation techniques from machine learning and sensitivity analysis (SA). Although uncertainty assessment is important in classical ML, it is particularly vital in differential equation discovery to form ensembles.

Uncertainty can be assessed for each component of either a classical ML model or a differential equation as an ML model:

`Architectural uncertainty` is typically assessed through techniques such as pruning [1] or ensemble methods [2], which quantify the robustness to structural variations.

`Parameter uncertainty` is often analyzed via sensitivity analysis. Local SA methods, such as the one-at-a-time (OAT) approach [3], assess the effect of small perturbations in individual inputs and keep others constant. However, these methods capture non-linearities and interactions poorly. In contrast, global SA methods, such as Sobol indices [4], evaluate the impact of variations across the parameter space and account for input interactions.

`Feature uncertainty` is managed through preprocessing, data augmentation, sampling, feature engineering, and strategies to handle noisy inputs, such as robust normalization techniques [5].

`The objective function`, although often defined by design, can introduce uncertainties when there is misalignment between the target and the model's capacity [6].

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Science.

Recent differential equation discovery methods allow us to treat differential equations as machine learning methods. Therefore, we could also find analogs to machine learning components. Every equation discovery method aims to identify the equation structure and terms likely to appear in the governing equation for the data; this structure is closely related to a neural network architecture, as it describes how features and layers are interconnected.

The second step is to identify the parameters. The parameters are the coefficients within the differential equation that frequently correspond to physical properties. They can be referred to as neural network weights (and are essentially coefficients of a specialized type of linear regression).

Advancements in differential equation discovery techniques have refined the assessment of uncertainty for these components. For example, parameter uncertainty has been addressed using ensemble-based approaches, such as E-SINDy [7], which employs term library ensembling to handle parameter robustness . Structural uncertainty has been explored using methods like multi-objective evolutionary optimization combined with Bayesian networks, as demonstrated by [8]. These approaches enable researchers to more accurately quantify structural robustness and align the DE identified with physical phenomena.

Unlike traditional machine learning, the objective function in DE discovery is more constrained. It is often defined as the discrepancy of the equation, evaluated either in a strong form (e.g., term-by-term residuals) or in a weak form (e.g., weak formulations such as wSINDy [9]). Solver-based methods are also employed to minimize discrepancies between observed data and solutions generated by the identified DE, such as physics-informed criterion (PIC) and others.

The critical distinction between differential equation discovery and machine learning lies in the treatment of features. The sole feature in differential equation discovery is based on observational data; it could be a time series or a field (a multidimensional tensor that contains time as one axis). However, to build an equation, we require differentials with respect to every axis up to the given order. Differentials of the input data are not provided in most cases and must therefore be computed numerically. Thus, from a machine learning perspective, the features are engineered within the algorithm.

Noisy measurements pose a challenge to numerical differentiation, leading to errors in derivative estimates. Stable numerical differentiation techniques (for example, finite differences, polynomial interpolation, or methods based on machine learning [10]) have been proposed to address these problems. However, the choice of differentiation method can significantly impact the quality of the discovered DE model. Variations in derivative computation propagate uncertainty into both the estimated parameters and the structure of the resulting differential equation.

Despite progress in addressing parameter and structural uncertainties in DE discovery, the impact of differentiation methods on feature uncertainty remains underexplored. This paper **aims** to systematically assess how differentiation techniques influence the quality of discovered models, with a particular focus on parameter and structural accuracy under varying levels of data uncertainty.

**Contribution:** - We describe differentiation as a “feature engineering” source of uncertainty in differential equation discovery.
- Experimentally prove the obvious fact that better differentiation quality leads to better discovery, but also the non-obvious fact that different methods should be used for noisy and clean data to achieve better performance.
- We compare several frameworks (SINDy and EPDE) to make the results more reliable.

**Limitations** Not every differential discovery method allows for easy modification of the differentiation method; for example, this is rarely done in RL-based equation discovery, such as DISCOVER [11].

**Data and code** will be available on GitHub in case of acceptance on paper.

## 2 Differential equation discovery background

As noted above, in the context of a differential equation as a machine learning model, we can distinguish the components of such a model: structure/architecture, parameters, features, and objective function.

For differential equations discovery, as input, we have the data placed on a discrete grid $X = \left\{x^{(i)} = \left(x_1^{(i)}, \dots x_{\dim}^{(i)}\right)\right\}_{i=1}^{i=N}$, where $N$ is the number of observations and dim is the dimensionality of the problem. We mention a particular case of time series, for which $\dim = 1$ and $X = \{t_j\}_{i=1}^{i=N}$. It is also assumed that for each point on the grid, there is an associated set of observations $U = \left\{u^{(i)} = \left(u_1^{(i)}, \dots, u_L^{(i)}\right)\right\}_{i=1}^{N}$ to define a grid map $u : X \subset \mathbb{R}^{\dim} \to U \subset \mathbb{R}^L$. This grid and observations can be used as input data or features in the machine learning model.

From differential equation theory, we expect $u$ to represent not only a function but also a jet, which is essentially differential up to a given order $r$ in form:

$$J^r = (x_1, ..., x_{\text{dim}}; u; D_1 u; D_2 u; ...; D_r u) \tag{1}$$

,where $D_r = \bigcup\limits_{|\alpha|=r} \{\frac{\partial^r u}{\partial x_1^{\alpha_1} \dots \partial x_r^{\alpha_r}}\}$ is the set of all partial differentials of order $r$ and $\alpha = \{\alpha_1, ...\alpha_{\text{dim}}\}, |\alpha| = \sum\limits_{i=1}^{i=\text{dim}} \alpha_i$ is just a differential multi-index. Since we usually have a single observation set $u$ we omit it from the notation $J^r(u)$

Any differential equation of an order $r$ is just a surface in a jet space $J^r$. Let $\mathcal{T}$ be a set of basis functions (monomials, compositions) acting on $\bar{J}^r$. Then $S \subset \mathcal{T}$ represents selected terms, and $P$ is the set of admissible coefficients. It could be a function of independent coordinates or just constants. Then the surface has the following form:

$$M(S, P) = \sum_{s \in S} p_s \cdot s(\bar{J}^r) = 0 \tag{2}$$

Having an analytical jet is the best possible case for the discovery of *differential equations*. We look for the surface within the jet space, nothing more. The real-case scenario significantly differs from the ideal "continuous" case, namely: (a) in most cases the jets are restored just from observation data $U$ with numerical differentiation,(b) we cannot look for any surface, we restrict the surface search space, and (c) if we want to find a "governing" law, we need to go beyond simple symbolic regression: the equation may not have unique solution, we may overlook some terms, terms presented in theoretical equation could have small magnitude and appear as numerical noise. In general, we require one to assess the uncertainty in both the structure and the coefficients. In most cases, it is done using ensembles.

**(a) On jets and numerical differentiation** Returning to the discrete setup, we have an approximate jet $\bar{J}^r = \{(x^{(i)}, u^{(i)}, D_h u^{(i)}, ...(D_h)^r u^{(i)})\}_{i=1}^{i=N}$ with $D_h$ denoting an arbitrary numerical differentiation operator. Here is much uncertainty. Generally, we do not know how the discrete observation set $U = \{u^{(i)}\}_{i=1}^{i=N}$ is connected with a true function $u$. For the selected numerical differentiation method $D_h$, we usually only know the order of approximation for a first-order differential, and we apply it several times to have higher-order differentials without any guarantee of higher-order differential approximation.

We expect that if the differentiation method is correct, then in a point-wise manner $\bar{J}^r \xrightarrow[N \to +\infty]{} J^r$. There is an ambiguity. In a real-world scenario, we cannot obtain more observational data. From the other side, for a finite computation starting from a certain number of points, the process becomes ill-posed. We discuss how to approach assessing it below.

**(b) Equation discovery problem statement** Let us assume that the discrete jet $\bar{J}^r$ is already computed for the observation data $U$. Therefore, we can treat items in $\bar{J}^r$ as symbols and formulate a symbolic regression problem.

As stated above, we have to find an explicit surface (2). Any machine learning requires restricting the search space to a finite one. As the first step, we define the loss function $L(M(S, P))$ and formulate the optimization problem.

$$S^*, P^* = \underset{S\in\Sigma, P\in\Pi}{\operatorname{argmin}} L(M(S,P)) \tag{3}$$

The methods of equation discovery differ in the way they determine $L(\cdot)$, the parameterization of the model $M(S,P)$, and the set of restrictions of the structure $\Sigma$, as well as the set of parameters $\Pi$. The structure and parameters are analogous to those of a machine learning algorithm's model architecture. Unlike machine learning $\Sigma$ and $\Pi$ in differential equation discovery, the optimization algorithm determines the results. Below, we briefly outline the main groups of methods.

In equation discovery, in the first place, we care about how $\Sigma$ is built. As a classical algorithm in the area, we consider another algorithm, Sparse Identification of Nonlinear Dynamics (SINDy) [12].

For the SINDy case, we manually determine the longest sentence $\Sigma_{\text{long}}$ possible and fix it. The optimization is performed only by $P$, which is essentially a vector of the numerical coefficients near each word of $\Sigma_{\text{long}}$. We need to make $P$ as sparse as possible, which is done with classical LASSO regression. In SINDy, we compute the loss function by using the discrepancy over the discrete grid.

$$P^* = \underset{P\in\Pi}{\operatorname{argmin}} ||M(\Sigma_{\text{long}}, P)||_2 + \alpha||P||_1 \tag{4}$$

In (4) we denote by $||\cdot||_2$ the mean discrepancy in the computation grid $X$ and by $||\cdot||_1$ is the $l_1$ norm. Since SINDy usually works with constant coefficients, we could use the $l_1$ norm to determine the sparsity of the set of parameters $P$. In some sense, it is a measure of the complexity of the surface in terms of the number of symbols needed to describe it.

Evolutionary approaches and reinforcement learning have their own rules to construct $S$ for a model. Every equation $S_i$ appearing within the optimization process is evaluated using the SINDy approach (4) with discrepancy or, as is done in EPDE, by constructing the Pareto frontier over the discrepancy and complexity criteria. Both discrepancy computation and Patero frontier forming are done as part of the fitness function computation or to form a reward for the reinforcement learning agent.

There are also more robust measures. For a given surface $M(S,P)$, we try to restore the continuous function $u$ that exactly generates the surface and then compare it with observations $U$. It, of course, requires the solution of the equation. We note that in this case, we do not need to consider jets $J^r$; instead, we begin working with the fibers $u$ and no longer need to consider the differentials $D_r$. In that case, all surfaces are single-connected, i.e., the solution of the equation is unique, which is, of course, a limitation, but it is more robust than a discrepancy measure.

There are also some intermediate cases, such as PIC. Here we spatially handle jets, but temporally restore continuous paths. It could be considered as jet factorization and partial fiber projection.

Ultimately, after optimization of equation (3), we obtain a single symbolic expression that represents a relation found in the discrete jet $\bar{J}^r$. We need to assess sensitivity to the data subsampling method, noise, and differentiation.

**(c) Ensemble sensitivity analysis** To assess sensitivity and pick stable appearing equations, we need to form ensembles. To briefly mention, E-SINDy is the first algorithm for the equation discovery method [7] that addresses this problem. We also have different ensembling methods [8].

Unlike what is usually stated in the literature, we do not need to have a single stable equation, but an ensemble that has a common part "in mean". Additionally, we typically consider the sensitivity and optimization stability of data subsampling. However, the sensitivity of the differentiation method, although it is an important part as shown above, is usually omitted.

## 3 Data differentiation problem statement and proposed methods

As seen above, the problem of using derivatives as features and overall differentiation methods is rarely mentioned in equation discovery. The differentials are used as symbols to form a discrete jet. Strictly speaking, discrete jets differ greatly based on the differentiation method used.

In particular, differentials here are considered handcrafted features within symbolic regression from a machine learning perspective. We are interested in the contribution the differentiation algorithm makes to equation discovery and the importance of the differentiation error. The ad hoc solution is that the error should not be too large for most problem statements.

To illustrate the problem, let us consider a straightforward numerical differentiation problem statement. The input to any equation discovery algorithm typically consists of noisy measurements. Denoting the true (noise-free) state by $\overline{u}(t, \mathbf{x})$ and the observed data by

$$u(t, \mathbf{x}) = \overline{u}(t, \mathbf{x}) + \epsilon(t, \mathbf{x}) \tag{5}$$

Typically, one assumes that $\epsilon(t, \mathbf{x})$ arises from additive white Gaussian noise (AWGN). In particular, each measurement is modeled as

$$u(t, \mathbf{x}) \sim \mathcal{N}\big(\overline{u}(t, \mathbf{x}),\, \sigma^2(t, \mathbf{x})\big), \quad \sigma(t, \mathbf{x}) = \kappa\, \big|\overline{u}(t, \mathbf{x})\big| \tag{6}$$

for some proportionality constant $\kappa$. Let us consider the simplest case of the central difference differentiation of a single-variable function:

$$u'(x) \approx \frac{u(x+h) - u(x-h)}{2\,h} \tag{7}$$

Substituting into the central difference noised data (5) yields

$$\begin{aligned} \tilde{u}'(x) = \frac{\big[u(x+h) + \epsilon(x+h)\big] - \big[u(x-h) + \epsilon(x-h)\big]}{2\,h} &= \\ = \underbrace{\frac{u(x+h) - u(x-h)}{2\,h}}_{\text{(deterministic, truncation part)}} &+ \underbrace{\frac{\epsilon(x+h) - \epsilon(x-h)}{2\,h}}_{\eta(x)} \end{aligned} \tag{8}$$

Because $\epsilon(x+h)$ and $\epsilon(x-h)$ are independent Gaussian random variables with variance $\kappa^2\, \big|\overline{u}(t, \mathbf{x})\big|^2$, it follows that

$$\mathrm{Var}\big[\eta(x)\big] \;=\; \mathrm{Var}\Big[\tfrac{\epsilon(x+h)-\epsilon(x-h)}{2\,h}\Big] \;=\; \frac{2\,\kappa^2\,\big|\overline{u}(t, \mathbf{x})\big|^2}{4\,h^2} \;=\; \frac{\kappa^2\,\big|\overline{u}(t, \mathbf{x})\big|^2}{2\,h^2}, \tag{9}$$

hence, the noise-induced standard deviation in the derivative estimate scales like $\kappa\,|u(x)|/(\sqrt{2}\,h)$.

On the other hand, a standard Taylor remainder analysis shows that the deterministic error (truncation) of the central difference approximation is $\mathcal{O}(h^2)$. Denoting by $k$ the constant that bounds the second derivative term in the remainder, one may write:

$$E_{\text{trunc}}(h) \approx k\,h^2, \qquad E_{\text{noise}}(h) \approx \frac{\kappa\,|u(x)|}{\sqrt{2}\ h}. \tag{10}$$

Therefore, the total error in the central-difference derivative for noisy data can be viewed as

$$E_{\text{total}}(h) \sim k\,h^2 \;+\; \frac{\kappa\,|u(x)|}{\sqrt{2}\,h}. \tag{11}$$

Minimizing $E_{\text{total}}(h)$ with respect to $h$ yields an optimal grid spacing and the corresponding minimal error scales as

$$h^* \;=\; \Big(\tfrac{c}{2\,k}\Big)^{1/3}, \qquad c = \frac{\kappa\,|u(x)|}{\sqrt{2}}, \qquad E_{\min} \sim c^{2/3}\,k^{1/3}\big(2^{-2/3} + 2^{1/3}\big). \tag{12}$$

Thus, although reducing $h$ decreases the truncation error, it amplifies the noise. It implies that an intermediate (non-zero) $h^*$ balances both contributions. In symbolic regression, where derivatives are treated as features, choosing $h$ too small will cause the noise amplification term to dominate, leading to spurious high-frequency artifacts. Choosing $h$ too large will smear genuine gradients and obscure critical dynamics. The differentiation algorithm should be correct to draw a general conclusion for noisy data, but increasing differentiation precision may lead to worse results.

For a general differential equation discovery algorithm, we have only a fixed differentiation method as a hyperparameter and a fixed grid, but the equation is not fixed. So, in some sense, we could only manipulate $k$ from the above equations. We could only assess the variance of the discrepancy distribution for a known equation-" answer ". However, we do not have any guarantees that none of the other equations could eventually achieve a lower discrepancy.

The discrepancy measure is used as the optimization criterion in most equation discovery methods. Since the discrepancy is averaged over the whole grid, we already use the weak form. However, as shown in the paper, the differentiation algorithm and error play a significant role in this process.

## 4 Experiments

We will conduct complex numerical experiments to investigate the influence of the above differentiation methods in DE discovery. DE discovery will be carried out using sparse regression [12] and an evolutionary approach [13].

All experiments were carried out using six differentiation methods in total, which are referred to as `Gradient`, `Adaptive`, `Polynomial`, `Spectral`, and `Total_var`. The detailed description of the methods used is in Appendix A. For the particular algorithms' parameters and realization, please refer to the code.

### 4.1 Experimental setup

Second-order ODE and several types of partial differential equations were chosen, each with different solutions: analytical (KdV), numerical (Burgers, wave, Laplace). Additionally, we use a data-driven model of ocean behavior [14] where the exact equation is not known a priori (referred to as pyqg below).

The workflow includes selecting and generating data; as noted earlier, it is either obtaining an analytical solution in the form of a matrix of values or finding a solution matrix using numerical methods, setting boundary and initial conditions, where necessary, and choosing constants. After that, all the data obtained are differentiated by the described methods, while the derivatives sought are those that, as is known in advance, occur in the equation. These may be derivatives of the form $\frac{\partial u(t,\mathbf{x})}{\partial x}$, $\frac{\partial u(t,\mathbf{x})}{\partial t}$, $\frac{\partial^2 u(t,\mathbf{x})}{\partial x^2}$, etc.

Then, an evolutionary algorithm is applied using the EPDE framework. Data were loaded with the grid and all derivatives, and then we chose a multi-objective mode. The population size is 7 for all equations, and the number of training epochs ranges from 30 to 80, depending on the equation's complexity; the maximum number of terms in each equation is 8. This is done to obtain greater variability in the equations. Then, the algorithm is run; one run yields approximately 5-7 equations per the Pareto frontier. We perform only 50 runs for each equation to minimize variation in the data and more accurately estimate the average approximation for all coefficients.

For each data set, a series of experiments was conducted, resulting in box plots (refer to the appendices) that show the distribution of coefficients preceding the correct terms. The difference between the obtained equations and the true ones was also analyzed using the Structural Hamming Distance (SHD) metric.

For each experiment, noise was added to the data as (6), with $\kappa = \{0, 0.5, 1\}\%$, which is referred to as `noise`. However, we provide results for noise $= 1$ for the particular equations. Other cases are discussed in the corresponding section.

## 5 Results

**Ordinary differential equation** As a simple example, we consider a second-order ODE in the form $mu'' + qu' + ku = 0$ with parameters $m = 1$, $q = 0.25$, $k = 3$, and initial conditions

$u(0) = 1, u'(0) = 0$. The detailed experimental results are placed in Appendix B with a further discussion in Section 6.

**Korteweg – de Vries equation** The Korteweg-de Vries equation is a partial differential equation $u'_t + u'''_{xxx} + 6uu'_x = 0$, which is one of the few that has analytical one-soliton and two-soliton solutions.

We will study its single-soliton solution, presented in the following form $u(x,t) = \frac{2(k^2)}{ch^2(-k(\mathbf{x}-4(k^2)t))}$ ,where $k = 0.7$ is the constant that determines the velocity of the soliton $4k^2$ and the amplitude $2k^2$. The detailed experimental results are placed in Appendix C with a further discussion in Section 6.

**Burger's equation** $u'_t + uu'_x = vu''_{xx}$ ,where $v = 0.05$ is the diffusion coefficient.

The solution was obtained using an implicit numerical scheme for the diffusion term and an explicit numerical scheme for the convective term. An initial condition was set, and the right and left boundaries were fixed at zero. The detailed experimental results are placed in Appendix D with a further discussion in Section 6.

**Wave equation** $u''_{xx} = c^2 u''_{tt}$, where $c = 0.25$ is the propagation speed of the wave. The initial conditions were set as a sinusoidal function, and the boundary conditions were fixed at zero. The finite difference method was then used to solve the problem. The detailed experimental results are placed in Appendix E with a further discussion in Section 6.

**Laplace equation** $u''_{xx} + u''_{yy} = 0$ . The Dirichlet boundary conditions were set, and the problem was solved using the finite difference method. The detailed experimental results are placed in Appendix F with a further discussion in Section 6.

**Quasigeostrophic potential vorticity** Original data were obtained using the pyqg framework [1] for quasi-geostrophic modeling. The maximum number of terms was extended to 15 to capture complex dynamics. Since the exact governing equations are unknown, we evaluate the discovered equations by comparing the discrepancy between the original data and numerical solutions from a Physics-Informed Neural Networks (PINNs) solver. PINNs are necessary due to the high non-linearity that renders conventional FEM inadequate. The general form of the governing equation is $\mathbf{V_g} \cdot \nabla q = 0$, where $V_g$ represents geostrophic velocity and $q$ denotes potential vorticity.

The equations presented were derived using Savitzky-Golay (SG) filtering and spectral domain differentiation methods, respectively, as alternative approaches failed to capture the eddy-driven structure of the derivatives, resulting in suboptimal preprocessing. The solutions to these equations similarly exhibit a lack of regions with pronounced eddy behavior, which may indicate a tendency toward identifying broader-scale features in the data. Visual representations of the original data, numerical solutions, and error maps are provided in Appendix G.

These results demonstrate that, in real-world cases, we cannot consistently achieve results for unknown equations and that we require ensembles that include both data subsampling and differentiation uncertainty.

## 6 Discussion

Our experiments reveal a counterintuitive reality: numerically precise differentiation is not a remedy for equation discovery. As shown in Tab. 1, methods such as Spectral achieve a minimal differentiation error (e.g., $9.988 \cdot 10^{-6}$ at 0% noise), but produce poor structural precision (SHD = $4 \pm 0.13$). In contrast, despite the high differentiation error (1.963 for 1% noise level). Polynomial consistently delivers superior structural recovery (SHD = 3 ± 0.13). This paradox arises because noise amplification from high-precision methods introduces misleading high-frequency artifacts. As we illustrate in a concrete example (see (11)), optimal discovery requires strategic smoothing rather than maximal precision within a single algorithm run.

To illustrate the general dependencies, we plot the SHD and error values for different methods on a scatter plot, as shown in Fig. 1.

[1]`http://github.com/pyqg/pyqg`

Table 1: Differentiation Method Performance Analysis

| | Noise Level | D1/D2/D3 error | Coeff. Error (Mean ± SD) | SHD (Mean ± SD) | Key Insight |
|---|---|---|---|---|---|
| Gradient | 0% | 0.0002/0.00027/0.00050 | 0.7309 ± 0.0515 | 2 ± 0.0782 | Precise but noisy; good for clean data |
| | 0.5% | 0.0017/0.0024/0.0170 | 0.9179 ± 0.0683 | 3 ± 0.1153 | Noise amplifies rapidly |
| | 1% | 0.0065/0.0036/0.0502 | 1.0068 ± 0.2449 | 4 ± 0.1482 | Avoid for noisy PDEs |
| Adaptive | 0% | 0.0016/0.2023/0.0121 | 0.7309 ± 0.0515 | 2 ± 0.0823 | Precise but noisy; good for clean data |
| | 0.5% | 0.0025/0.2079/0.0221 | 0.9179 ± 0.0683 | 3 ± 0.0723 | Noise amplifies rapidly |
| | 1% | 0.0341/0.6116/0.0432 | 1.0068 ± 0.2449 | 4 ± 0.0971 | Avoid for noisy PDEs |
| Polynomial | 0% | 0.0193/0.3344/0.0350 | 0.8971 ± 0.0517 | 2 ± 0.1300 | **Low SHD despite high diff. error** |
| | 0.5% | 0.0236/0.6421/0.0366 | 0.9611 ± 0.0390 | 3 ± 0.1260 | Robust structure recovery |
| | 1% | 0.0302/1.9630/0.0390 | 0.9148 ± 0.1551 | 3 ± 0.1323 | Best SHD-noise tradeoff |
| Spectral | 0% | 0.0683/11.7856/14.1910 | 1.1074 ± 0.0461 | 4 ± 0.1309 | **High precision, poor SHD** |
| | 0.5% | 0.0716/13.0997/14.1966 | 1.1737 ± 0.0462 | 5 ± 0.1558 | Boundary artifacts dominate |
| | 1% | 0.0800/17.8008/14.2271 | 1.1924 ± 0.0466 | 4 ± 0.1498 | Unreliable under noise |
| Inverse | 0% | 1.5583/52.5442/0.4601 | 1.0482 ± 0.0518 | 4 ± 0.1929 | Moderate SHD, high coeff. variance |
| | 0.5% | 1.5655/71.5591/0.4617 | 1.2806 ± 0.0656 | 4 ± 0.1763 | **Worst coeff. error at low noise** |
| | 1% | 1.5676/76.2041/0.4931 | 1.3120 ± 0.0816 | 5 ± 0.1537 | Avoid for high-order terms |
| Total_var | 0% | 1.6218/52.9117/0.3273 | 1.0482 ± 0.0518 | 3 ± 0.1055 | Smoothing improves structural recovery |
| | 0.5% | 1.6292/54.4074/0.3275 | 1.2806 ± 0.0656 | 3 ± 0.1049 | Consistent performance across noise levels |
| | 1% | 1.6296/58.7889/0.3275 | 1.3120 ± 0.0816 | 3 ± 0.1065 | **Good structure despite high error** |

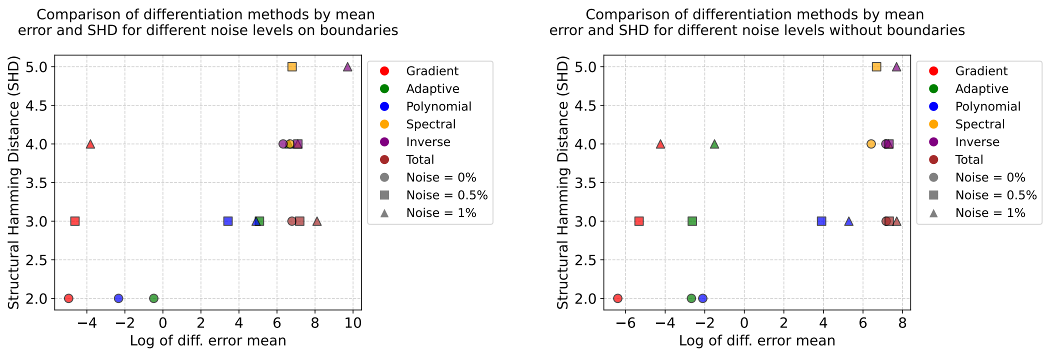

Figure 1: Comparison of differentiation methods by mean differentiation error and SHD for different noise levels on boundaries (left) and without boundaries (right)

## 7 Conclusion

The paper considers another aspect of differential equation discovery as a machine learning method.

The error of the differentiation algorithm, as the "feature engineering" method plays a role in the general uncertainty, is often left out of the scope.

The main results are as follows.

- The differentiation is an important part of every differential equation discovery method
- The differentiation is a reliable uncertainty source and may be used in ensembles to get the models on a different process scale
- Absolute value of differentiation error is less important – very precise methods give poor discovery results in some cases
- Significant differentiation error is allowed in the derivative, and in some cases, it is necessary to accept it to work with noisy data

We also note that the conclusion remains the same regardless of the method used, whether it is LASSO regression-based SINDy or evolutionary EPDE.

## Acknowledgments and Disclosure of Funding

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-15-2025-010.

## References

[1] Blalock, D., J. J. Gonzalez Ortiz, J. Frankle, et al. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[2] Lakshminarayanan, B., A. Pritzel, C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[3] Hamby, D. M. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment*, 32:135–154, 1994.

[4] SoboÍ, I. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp.*, 1:407, 1993.

[5] Werner de Vargas, V., J. A. Schneider Aranda, R. dos Santos Costa, et al. Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. *Knowledge and Information Systems*, 65(1):31–57, 2023.

[6] Gonzalez, S., R. Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *2020 IEEE congress on evolutionary computation (CEC)*, pages 1–8. IEEE, 2020.

[7] Fasel, U., J. N. Kutz, B. W. Brunton, et al. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260):20210904, 2022.

[8] Hvatov, A., R. Titov. Towards true discovery of the differential equations. *arXiv preprint arXiv:2308.04901*, 2023.

[9] Messenger, D. A., D. M. Bortz. Weak sindy: Galerkin-based data-driven model selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, 2021.

[10] Chartrand, R. Numerical differentiation of noisy, nonsmooth data. *International Scholarly Research Notices*, 2011, 2011.

[11] Du, M., Y. Chen, D. Zhang. Discover: Deep identification of symbolically concise open-form partial differential equations via enhanced reinforcement learning. *Physical Review Research*, 6(1):013182, 2024.

[12] Brunton, S. L., J. L. Proctor, J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[13] Maslyaev, M., A. Hvatov, A. V. Kalyuzhnaya. Partial differential equations discovery with epde framework: Application for real and synthetic data. *Journal of Computational Science*, 53:101345, 2021.

[14] Ross, A., Z. Li, P. Perezhogin, et al. Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. *Journal of Advances in Modeling Earth Systems*, 15(1), 2023.

[15] Rahaman, N., A. Baratin, D. Arpit, et al. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[16] Savitzky, A., M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[17] Johnson, S. G. Notes on fft-based differentiation. *MIT Applied Mathematics, Tech. Rep.*, 2011.

[18] Chartrand, R. Numerical differentiation of noisy, nonsmooth, multidimensional data. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 244–248. IEEE, 2017.

[19] Schmid, M., D. Rath, U. Diebold. Why and how savitzky–golay filters should be replaced. *ACS Measurement Science Au*, 2(2):185–196, 2022.

[20] Rudin, L. I., S. Osher, E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

## A Differentiation approach formulation

All experiments were carried out using six differentiation methods in total: Gradient - forward finite difference method, second order; Adaptive - numerical derivatives of dense neural network outputs; Polynomial - Savitzky-Golay filtering with interpolating polynomial; Spectral - spectral domain differentiation; Total_var - total variation regularization; Inverse - dense neural network with loss containing the inverse operator to differentiation.

- **Filtering-based approaches:** One of the approaches considered in this work involves approximating the input data with the fully connected artificial neural network (ANN). One of the valuable properties of the artificial neural network is that the low-frequency signal in the data is learned first, while further training approximates the high-frequency components [15]. Thus, by training an ANN representation of the process, we can obtain its low-frequency approximation, which can be further differentiated with a decreased noise component. Moreover, training ANN to represent the process allow us to obtain derivatives via automatic differentiation method.

  Savitzky-Golay (SG) filtering, developed in [16], is a commonly used approach to signal or data filtering, coupled with an opportunity to compute derivatives, involves a least squares-based local fitting of the polynomials to represent the data. For each grid node, the data in its proximity is used to construct a polynomial that can be analytically differentiated.
- **Spectral domain differentiation:** Although the process of differentiation in the spatial domain can be complicated for the data described with an arbitrary function, in the Fourier domain, the derivatives can be estimated on a term-to-term basis [17]. The discrete Fourier transform (DFT) is the basis of our implementation of spectral domain differentiation. In the spectral domain, integration and differentiation can be maintained by multiplication of series terms with an appropriate exponential. This leads to low computational costs, especially if the data are located on a uniform grid, thus allowing the use of the Fast Fourier Transform instead of DFT. The Butterworth filter does the signal filtering, which can preserve the signals with frequencies lower than the cutoff frequency while dampening the high-frequency ones.
- **Total variation regularization:** Variational principles provide an alternative method that incorporates inverse problem solution with the regularization of the gradient variation or its higher-order analogs (e.g., Hessian). One of the main advances in this field was made in [10, 18].

### A.1 Savitzky-Golay filtering

Savitzky-Golay (SG) filtering, developed in [16], is a commonly used approach to signal or data filtering, coupled with an opportunity to compute derivatives, involves a least squares-based local fitting of the polynomials to represent the data. To the set of data samples along an axis, we introduce the window of (commonly, odd) length $N = 2M + 1$, allowing the construction of series of polynomials $P_0(x), P_1(x),$ ... up to (even) order $n$, $n < N$ to approximate the data in the interior of our domain. With the selection of appropriate window size, from which the function values are used for the approximation, and polynomial order, the overdetermined system is constructed. Its solution provides the polynomial coefficients that represent the smoothed signal, without oscillations, caused by the random error. Even though the boundaries of length $M$ can be processed in a separate way, with the finite-difference schema or by a shifted approximation, the quality of results tend to decrease, thus for the equation discovery only the domain interior shall be used.

During the calculation of the partial derivative $u'_j$ for the sample $u(x_i)$, matching the $x_i$ grid node along the $j$-th axis, we select samples $\mathbf{u}_i = (u_{i-M}, u_{i-M+1}, \ ... \ , u_i, \ ... \ , u_{i+M})$ in the aforementioned window. Using the corresponding coordinates $\mathbf{y}_i = (x_{i-M}, \ ... \ , x_i, \ ... \ , x_{i+M})$, we introduce the least-square problem of detecting coefficient vector $\alpha = (\alpha_0, \ ... \ , \alpha_{n-1})$ for the series $P_0, \ ... \ , P_{n-1}$. The representation of data samples is as follows:

$$u_i = \sum_{k=0}^{n-1} \alpha_k P_k(x_i). \tag{13}$$

$$\alpha = \arg\min_{\alpha'} |\mathbf{u}_i - P\mathbf{y}_i|, \tag{14}$$

where matrix $P$ contains values of the polynomials in the grid nodes.

In our case, we utilize orthogonal Chebyshev polynomials of the first kind, where by $C_m^{2k}$ we denote the number of combination of $2k$ elements from the set of cardinality $m$:

$$T_m(x) = \sum_{k=0}^{\lfloor m/2 \rfloor} C_m^{2k}(x^2-1)^k x^{m-2k} \tag{15}$$

Having a series of Chebyshev polynomials with calculated coefficients, differentiation can be held analytically. Using the representation of data as series in 13, we get the derivative as $u_i' = \sum_{k=0}^{n-1} \alpha_k U_k(x_i)$, where $U_k$ is a Chebyshev polynomial of the second kind.

$$U_m(x) = \sum_{k=0}^{\lfloor m/2 \rfloor} C_{m+1}^{2k+1}(x^2-1)^k x^{m-2k} \tag{16}$$

Although the provided approach is capable of filtering the data and stably calculating the derivatives, work [19] suggests that modification of Savitzky-Golay filtering by adding fitting weights or by implementing other filters, such as Whittaker-Henderson filter, can lead to better results in noise suppression.

## A.2 Spectral domain differentiation

Although the process of differentiation in the spatial domain can be complicated for the data, described with an arbitrary function, in the Fourier domain the derivatives can be estimated in term-to-term basis [17]. In general, the series of the derivatives, taken on a term-to-term basis may not converge. However, if we assume that the data represents continuous piecewise smooth function that has piecewise differentiable derivatives, the data can be differentiated term-to-term.

A discrete Fourier transform (DFT) is the basis for our implementation of spectral domain differentiation. Let us examine a case of one-dimensional data, even though the algorithm can operate on multi-dimensional data, with the canonical discrete Fourier transform algorithm replaced by n-dimensional DFT. In data-driven equation discovery problems, one-dimensional data $u(t)$ is viewed from the point of view of samples $u_n = u(nT/N), n = 0, 1, \dots, N-1$, where $T$ is the length of time interval and $N$ - the number of samples, and the corresponding coordinates will be $t_n = nT/N, n = 0, 1, \dots, N-1$. The Fourier coefficients are denoted as $\hat{u}_k$, and they are calculated as:

$$\hat{u}_k = \frac{1}{N} \sum_{n=0}^{N-1} u_n exp(-2\pi i \frac{nk}{N}). \tag{17}$$

In many cases, the data are provided on the regular (even multi-dimensional) grid, thus to improve the algorithm performance a fast Fourier transform can be used. Due to the lower computational complexity, the increase in performance is substantial. The process of data reconstruction, using the obtained Fourier coefficients, is held with an inverse discrete Fourier transform:

$$u_n = \sum_{k=0}^{N-1} \hat{u}_k exp(2\pi i \frac{nk}{N}). \tag{18}$$

Full term-by-term differentiation is performed in the Fourier domain, and the derivatives values are computed by the inverse DFT. For example, an expression for the first-order derivative has form, as in Eq. 19.

$$u'(t_k) = \sum_{0<k<\frac{N-1}{2}} \frac{2\pi i}{T} k \left( \hat{u}_n exp(2\pi i \frac{nk}{N}) - \hat{u}_{N-k} exp(-2\pi i \frac{nk}{N}) \right). \tag{19}$$

Filtering with the desired properties can be done with low-pass filters that pass signals with lower frequencies, while dampen the high-frequency ones. Butterworth filter is a representative of such tools, and is flat for the passband (the frequencies that we do not want to penalize). The latter property prevents distortion of the modeled process by introducing factors, close to 1, to the low-frequency Fourier components. The penalizing factor is introduced with the expression eq. 20:

$$G(\omega) = \frac{1}{1 + (\omega/\omega_{cutoff})^{2s}}, \tag{20}$$

where $\omega$ is the frequency, $\omega_{cutoff}$ is the cutoff frequency, indicating the boundary frequency, from which the damping begins, and $s$ is the filter steepness parameter. The resulting expression is obtained with the introduction of penalizing factors $G(\omega) = G(k/N)$ into the series, representing derivatives:

$$u'(t_k) = \sum_{0<k<\frac{N-1}{2}} G(k/N) \frac{2\pi i}{T} k \left( \hat{u}_n exp(2\pi i \frac{nk}{N}) - \hat{u}_{N-k} exp(-2\pi i \frac{nk}{N}) \right) \tag{21}$$

The derivative of the higher orders can be calculated recursively from the lower order ones with the same filtering-based differentiation procedures, or, preferably, by the further multiplication with the integrating coefficient and IDFT.

### A.3 Total variation regularization

Variational principles provide an alternative method that incorporates inverse problem solution with the regularization of the variation of the gradient or its higher order analogues (e.g. Hessian). Rudin-Osher-Fatemi model [20] in its discrete formulation can be represented by the optimization problem of minimizing functional 22.

$$|D(\nabla \cdot u)|_1 + \frac{\mu}{2} |K(\nabla \cdot u) - u|_2^2 \longrightarrow \min_u, \tag{22}$$

where $\nabla \cdot u = (\frac{\partial u}{\partial t}, \frac{\partial u}{\partial x_1}, \ ...)$ is the gradient of the data field and $K$ and $D = (D_t, D_{x_1}, D_{x_2}, \ ...)$ represent discrete integration operators onf differentiation. Regularization of gradient variation is maintained with term $|D(\nabla \cdot u)|_1 = \sum_\Omega \sqrt{\sum_{i,\, j} \frac{\partial^2 u)}{\partial x_i \partial x_j}}$.

[10, 18]

Although there are multiple approaches to the solution of the problem, we employ an approach, proposed in articles [10, 18], that is designed for a function of one variable. While this approach can be generalized to the problems of higher dimensionality, the computational costs associated with the optimization limit the method's applicability to large datasets. To perform the functional optimization required in Eq. 22, the corresponding Euler-Lagrange equation has to be formed and solved.

## B ODE equation coefficients and Structural Hamming Distances

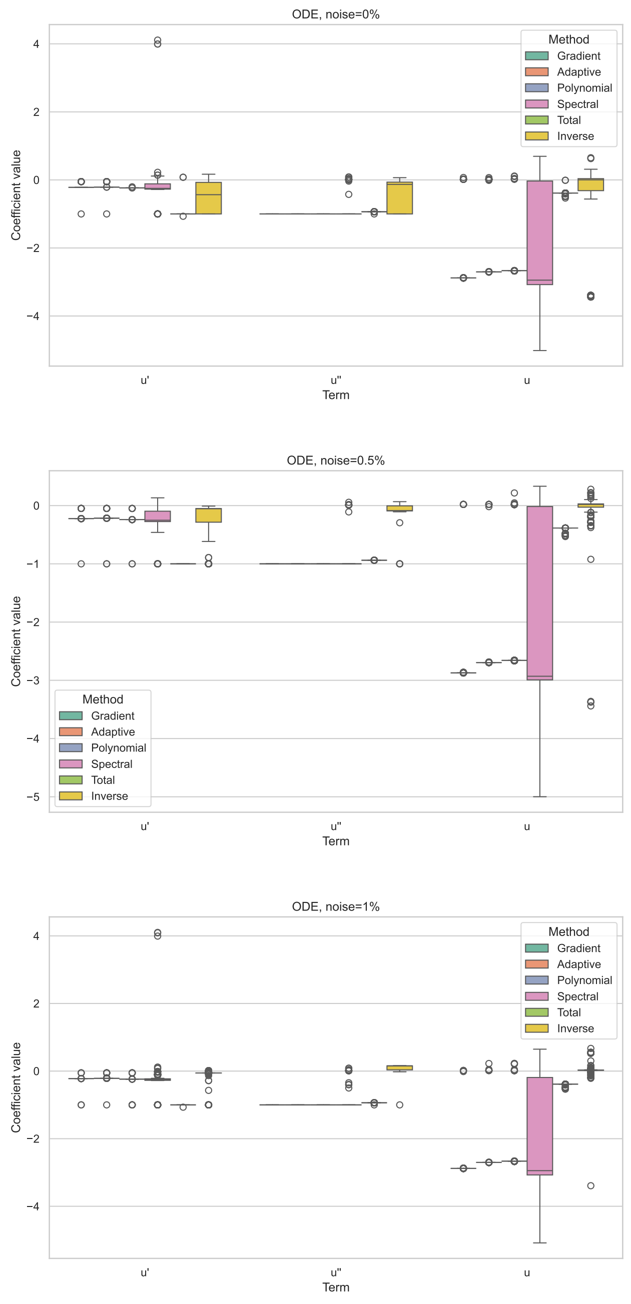

Figure 2: Distribution of coefficients values for different noise level

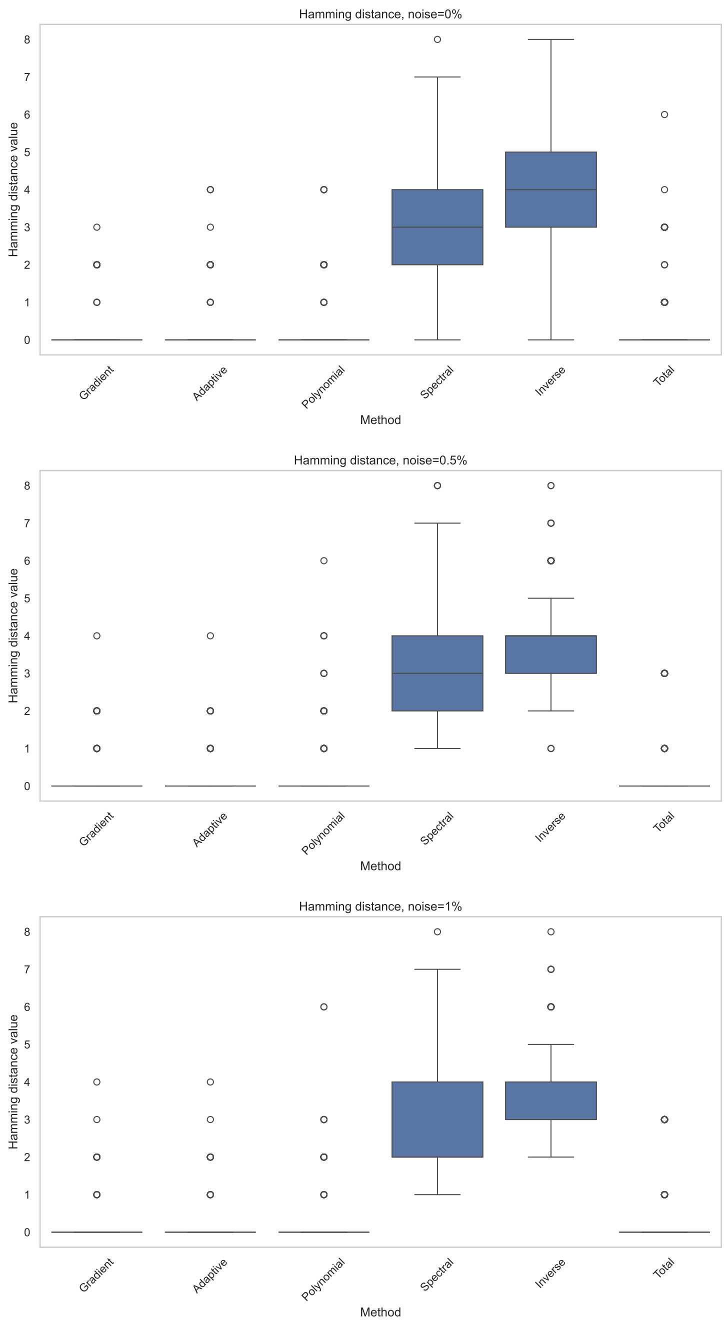

Figure 3: Distribution of coefficients values for different noise level

Table 2: Coefficients values calculated with EPDE, noise = 0%

| Methods/Terms | u | u’ | u” |
|---|---|---|---|
| Gradient | 0.0219 ± 0.0082 | -0.2156 ± 0.0048 | -1 |
| Adaptive | 0.0197 ± 0.0077 | -0.2086 ± 0.0053 | -1 |
| Polynomial | 0.0541 ± 0.0452 | -0.2348 ± 0.0002 | -1 |
| Spectral | -0.0312 ± 0.0412 | -0.3008 ± 0.0392 | -0.8997 ± 0.0460 |
| Inverse | -0.0142 ± 0.0282 | -0.5213 ± 0.0632 | -0.5069 ± 0.1026 |
| Total_var | -0.3901 ± 0.0032 | -0.9952 ± 0.0068 | -0.9353 ± 0.0003 |
| Ground truth | 3 | 0.25 | 1 |

Table 3: Coefficients values calculated with EPDE, noise = 0.5%

| Methods/Terms | u | u’ | u” |
|---|---|---|---|
| Gradient | 0.0205 ± 0.0000 | -0.2222 ± 0.0043 | -1 |
| Adaptive | 0.0152 ± 0.0112 | -0.2173 ± 0.0055 | -1 |
| Polynomial | 0.0363 ± 0.0331 | -0.2412 ± 0.0058 | -1 |
| Spectral | -0.0288 ± 0.0241 | -0.2694 ± 0.0375 | -0.9048 ± 0.0480 |
| Inverse | -0.0255 ± 0.0177 | -0.2840 ± 0.0533 | -0.1348 ± 0.0804 |
| Total_var | -0.3937 ± 0.0027 | -1 | -0.9387 |
| Ground truth | 3 | 0.25 | 1 |

Table 4: Coefficients values calculated with EPDE, noise = 1%

| Methods/Terms | u | u’ | u” |
|---|---|---|---|
| Gradient | 0.0098 ± 0.0251 | -0.2248 ± 0.0053 | -1 |
| Adaptive | 0.0444 ± 0.0523 | -0.2149 ± 0.0040 | -1 |
| Polynomial | 0.0983 ± 0.0748 | -0.2455 ± 0.0082 | -1 |
| Spectral | -0.0651 ± 0.0643 | -0.3254 ± 0.0391 | -0.9172 ± 0.0400 |
| Inverse | 0.0639 ± 0.0191 | -0.1889 ± 0.0410 | 0.0789 ± 0.0746 |
| Total_var | -0.3949 ± 0.0027 | -1.0002 ± 0.0003 | -0.9371 ± 0.0003 |
| Ground truth | 3 | 0.25 | 1 |

Table 5: Coefficients values calculated with SINDy, noise =0%

| Methods/Terms | u | u’ | u” |
|---|---|---|---|
| Gradient | 2.845 | 0.208 | 1 |
| Adaptive | 2.385 | 0.249 | 1 |
| Polynomial | 2.874 | 0.193 | 1 |
| Spectral | 3.199 | - | 1 |
| Inverse | 2.732 | 0.264 | 1 |
| Total_var | 0.413 | 1.070 | 1 |
| Ground truth | 3 | 0.25 | 1 |

Table 6: Coefficients values calculated with SINDy, noise =0.5%

| Methods/Terms | u | u’ | u” |
|---|---|---|---|
| Gradient | 2.824 | 0.212 | 1 |
| Adaptive | 2.368 | 0.253 | 1 |
| Polynomial | 2.854 | 0.206 | 1 |
| Spectral | 3.180 | - | 1 |
| Inverse | 3.697 | 0.376 | 1 |
| Total_var | 0.409 | 1.066 | 1 |
| Ground truth | 3 | 0.25 | 1 |

Table 7: Coefficients values calculated with SINDy, noise =1%

| Methods/Terms | u | u’ | u” |
|---|---|---|---|
| Gradient | 2.754 | 0.212 | 1 |
| Adaptive | 2.353 | 0.256 | 1 |
| Polynomial | 2.785 | 0.176 | 1 |
| Spectral | 3.197 | - | 1 |
| Inverse | 3.240 | 0.268 | 1 |
| Total_var | 0.414 | 1.072 | 1 |
| Ground truth | 3 | 0.25 | 1 |

## C KdV equation coefficients and Structural Hamming Distances

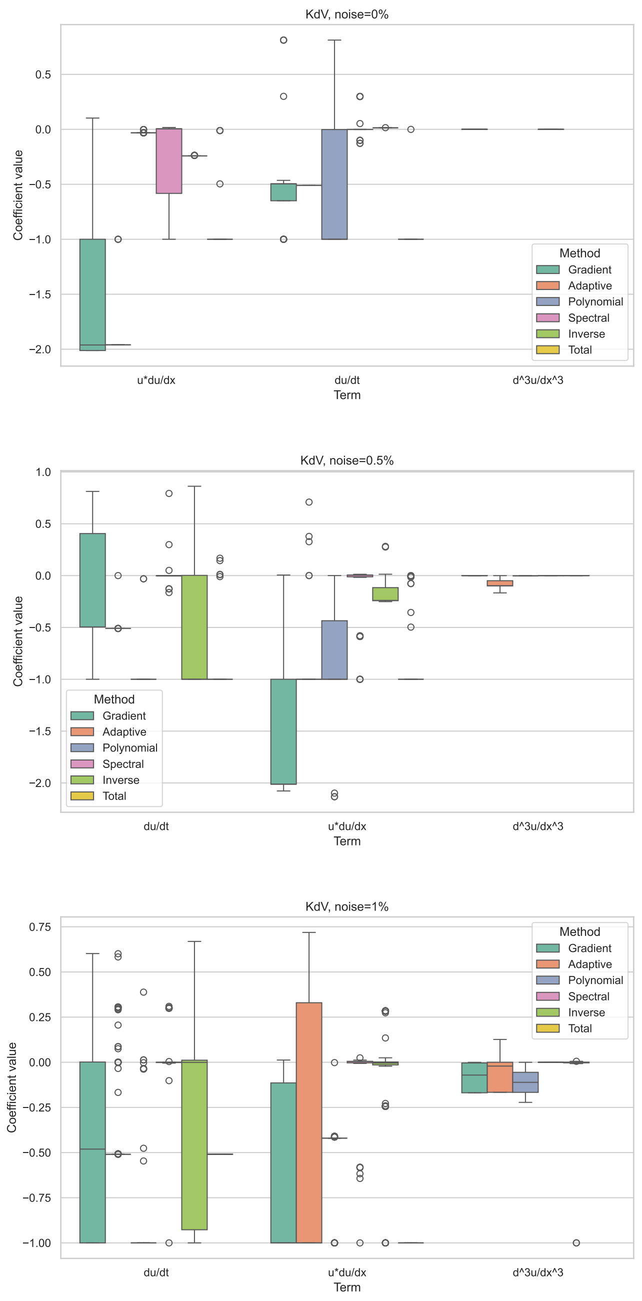

Figure 4: Distribution of coefficients values for different noise level

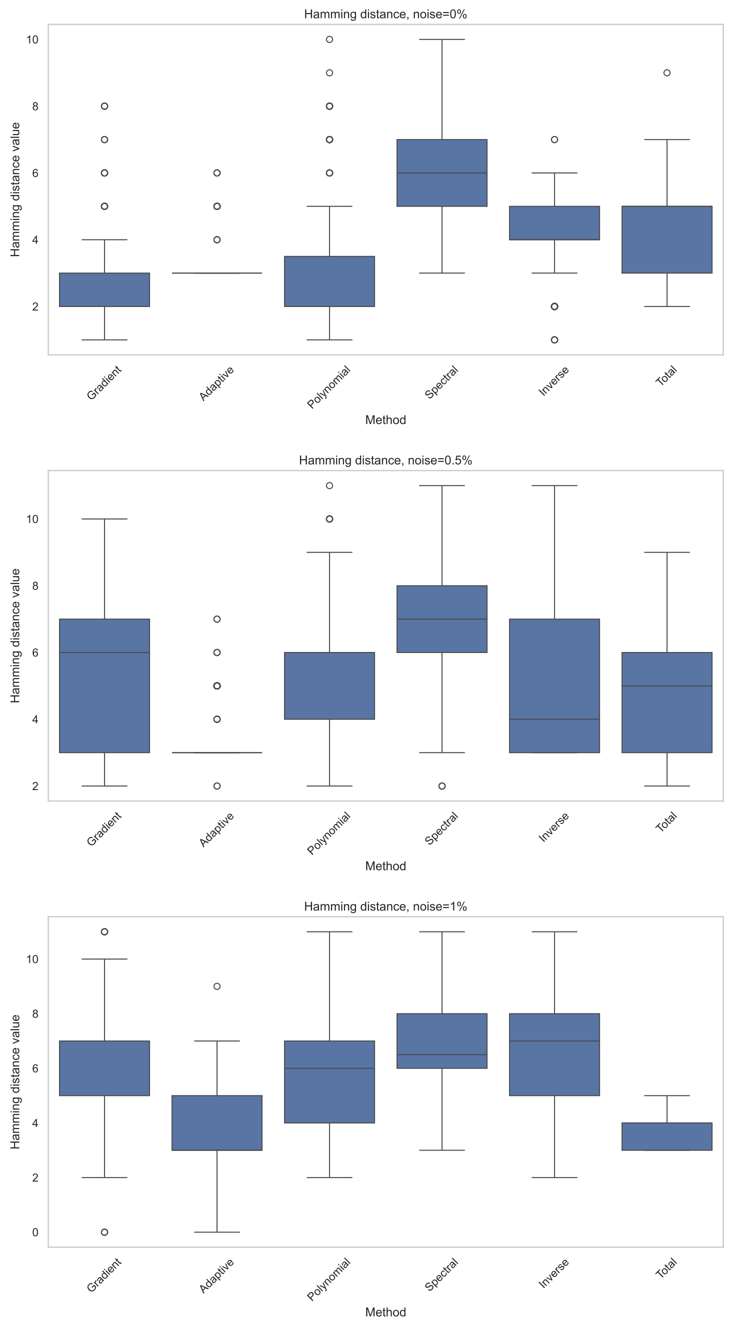

Figure 5: Distribution of coefficients values for different noise level

Table 8: Coefficients values calculated with EPDE, noise =0%

| Methods/Terms | du/dt | d^3u/dx^3 | u*du/dx |
|---|---|---|---|
| Gradient | -0.4565 ± 0.2045 | 0.0008 ± 0.0038 | -1.3444 ± 0.1839 |
| Adaptive | -0.5102 | - | -1.9143 ± 0.0645 |
| Polynomial | -0.5045 ± 0.4228 | - | -0.0303 ± 0.0011 |
| Spectral | 0.0202 ± 0.0401 | 0.0002 ± 0.0000 | -0.2297 ± 0.1165 |
| Inverse | 0.0142 ± 0.0007 | - | -0.2412 ± 0.0004 |
| Total_var | -0.8334 ± 0.4282 | 1.1503 | -0.9770 ± 0.0231 |
| Ground truth | 1 | 1 | 6 |

Table 9: Coefficients values calculated with EPDE, noise =0.5%

| Methods/Terms | du/dt | d^3u/dx^3 | u*du/dx |
|---|---|---|---|
| Gradient | -0.1973 ± 0.2523 | -0.0001 | -1.5692 ± 0.1447 |
| Adaptive | -0.5081 ± 0.0035 | -0.0801 ± 0.0554 | -0.7084 ± 0.2506 |
| Polynomial | -0.9822 ± 0.0248 | -0.0022 | -0.8874 ± 0.2106 |
| Spectral | 0.0191 ± 0.0614 | -0.0000 ± 0.0003 | -0.1706 ± 0.1092 |
| Inverse | -0.4698 ± 0.1770 | 0.0001 ± 0.0003 | 0.0419 ± 0.0954 |
| Total_var | -0.8164 ± 0.1561 | 1.1569 ± 0.1997 | -0.9282 ± 0.0466 |
| Ground truth | 1 | 1 | 6 |

Table 10: Coefficients values calculated with EPDE, noise =1%

| Methods/Terms | du/dt | d^3u/dx^3 | u*du/dx |
|---|---|---|---|
| Gradient | -0.3711 ± 0.1267 | -0.0819 ± 0.0647 | -0.7199 ± 0.3075 |
| Adaptive | -0.4008 ± 0.0431 | -0.0553 ± 0.0512 | -0.3934 ± 0.1230 |
| Polynomial | -0.6898 ± 0.1512 | -0.1111 ± 1.4098 | -0.9766 ± 0.1917 |
| Spectral | 0.0320 ± 0.0667 | -0.0001 ± 0.0006 | -0.1036 ± 0.0779 |
| Inverse | -0.1240 ± 0.0939 | 0.1313 ± 0.1499 | 0.0565 ± 0.0776 |
| Total_var | -0.8394 ± 0.1006 | - | -0.6780 ± 0.1103 |
| Ground truth | 1 | 1 | 6 |

Table 11: Coefficients values calculated with SINDy, noise =0%

| Methods/Terms | du/dt | d^3u/dx^3 | u*du/dx |
|---|---|---|---|
| Gradient | 1 | -0.009 | 0.077 |
| Adaptive | 1 | - | 0.195 |
| Polynomial | 1 | - | 0.595 |
| Spectral | 1 | -0.067 | 2.530 |
| Inverse | 1 | 0.072 | 0.025 |
| Total_var | 1 | -4.011 | -0.599 |
| Ground truth | 1 | 1 | 6 |

Table 12: Coefficients values calculated with SINDy, noise =0.5%

| Methods/Terms | du/dt | d^3u/dx^3 | u*du/dx |
|---|---|---|---|
| Gradient | 1 | 0.158 | 1.295 |
| Adaptive | 1 | 0.146 | 1.320 |
| Polynomial | 1 | - | 0.472 |
| Spectral | 1 | -0.066 | 2.530 |
| Inverse | 1 | - | - |
| Total_var | 1 | -4.010 | -0.639 |
| Ground truth | 1 | 1 | 6 |

Table 13: Coefficients values calculated with SINDy, noise =1%

| Methods/Terms | du/dt | d^3u/dx^3 | u*du/dx |
|---|---|---|---|
| Gradient | 1 | 0.052 | 0.443 |
| Adaptive | 1 | 0.056 | 0.681 |
| Polynomial | 1 | - | 0.841 |
| Spectral | 1 | -0.067 | 2.523 |
| Inverse | 1 | - | - |
| Total_var | 1 | -4.015 | -0.731 |
| Ground truth | 1 | 1 | 6 |

## D Burgers equation coefficients and Structural Hamming Distances

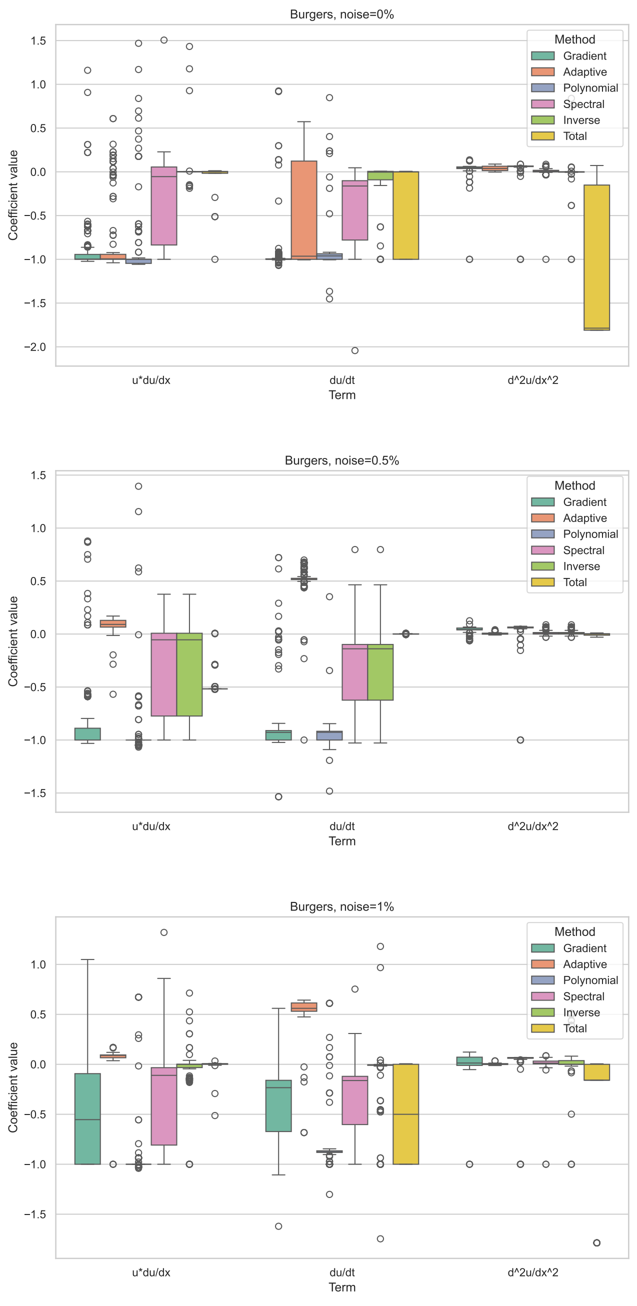

Figure 6: Distribution of coefficients values for different noise level

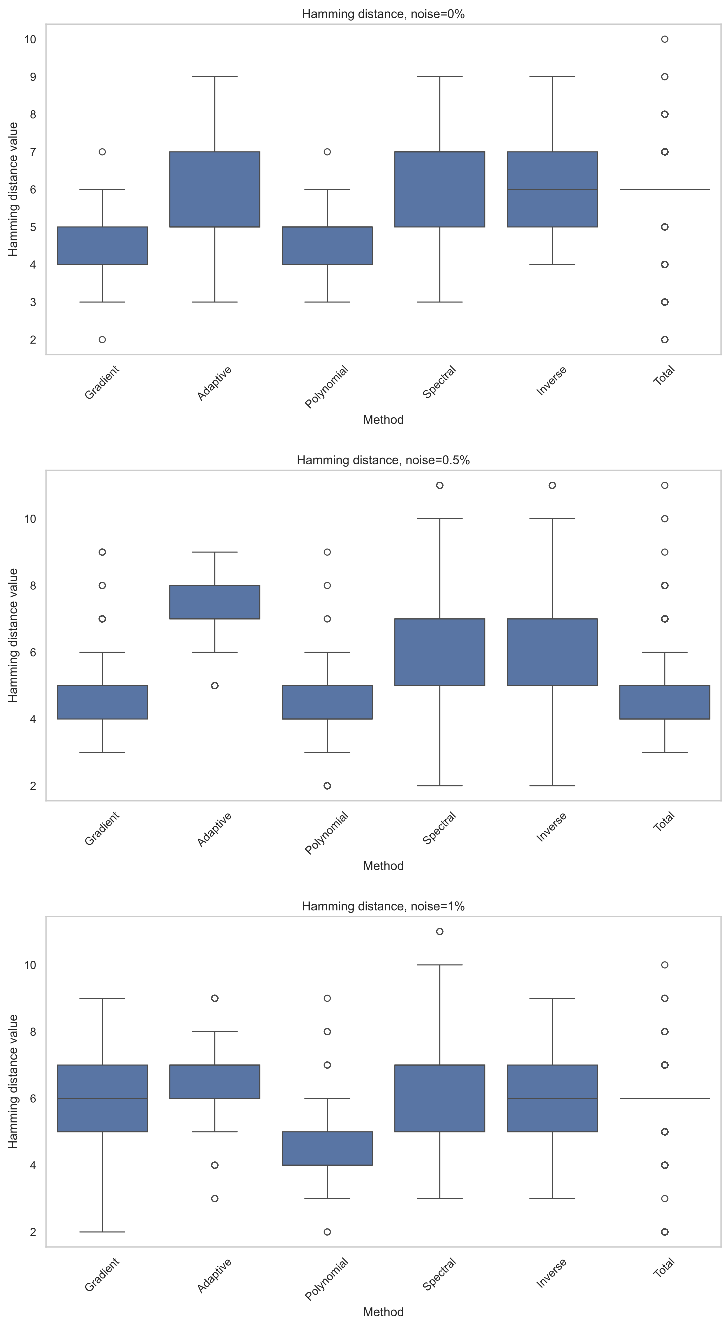

Figure 7: Distribution of coefficients values for different noise level

Table 14: Coefficients values calculated with EPDE, noise =0%

| Methods/Terms | du/dt | d^2u/dx^2 | u*du/dx |
|---|---|---|---|
| Gradient | -0.9454 ± 0.0301 | 0.0346 ± 0.0133 | -0.8945 ± 0.0327 |
| Adaptive | -0.4548 ± 0.6341 | 0.0402 ± 0.026 | -0.7677 ± 0.4413 |
| Polynomial | -0.9283 ± 0.0332 | 0.0439 ± 0.0188 | -0.8931 ± 0.0556 |
| Spectral | -0.4025 ± 0.0549 | 0.0032 ± 0.0171 | -0.3732 ± 0.0849 |
| Inverse | -0.2118 ± 0.1229 | -0.0239 ± 0.1173 | 0.0773 ± 0.1174 |
| Total_var | -0.4373 ± 0.2731 | -1.2180 ± 0.2525 | -0.1324 ± 0.1249 |
| Ground truth | 1 | -0.05 | 1 |

Table 15: Coefficients values calculated with EPDE, noise =0.5%

| Methods/Terms | du/dt | d^2u/dx^2 | u*du/dx |
|---|---|---|---|
| Gradient | -0.8727 ± 0.0398 | 0.0428 ± 0.0035 | -0.8520 ± 0.0479 |
| Adaptive | -0.5185 ± 0.0164 | 0.0069 ± 0.0031 | -0.0718 ± 0.0337 |
| Polynomial | -0.9428 ± 0.0150 | 0.0378 ± 0.0202 | -0.9510 ± 0.0382 |
| Spectral | -0.3064 ± 0.0521 | 0.0138 ± 0.0033 | -0.3226 ± 0.0810 |
| Inverse | -0.3064 ± 0.0521 | 0.0138 ± 0.0033 | -0.3226 ± 0.0810 |
| Total_var | 0.0008 ± 0.0011 | -0.0041 ± 0.0115 | -0.5039 ± 0.0086 |
| Ground truth | 1 | -0.05 | 1 |

Table 16: Coefficients values calculated with EPDE, noise =1%

| Methods/Terms | du/dt | d^2u/dx^2 | u*du/dx |
|---|---|---|---|
| Gradient | -0.4088 ± 0.0448 | 0.0051 ± 0.0269 | -0.5262 ± 0.0840 |
| Adaptive | 0.5498 ± 0.0191 | 0.0033 ± 0.0026 | 0.0577 ± 0.0274 |
| Polynomial | -0.8245 ± 0.0360 | 0.0384 ± 0.0208 | -0.9395 ± 0.0375 |
| Spectral | -0.3414 ± 0.0472 | 0.0049 ± 0.0207 | -0.3910 ± 0.0798 |
| Inverse | -0.1569 ± 0.0575 | 0.0238 ± 0.0420 | -0.0533 ± 0.0453 |
| Total_var | -0.4989 ± 0.1903 | -0.3595 ± 0.2082 | -0.0269 ± 0.0465 |
| Ground truth | 1 | -0.05 | 1 |

Table 17: Coefficients values calculated with SINDy, noise =0%

| Methods/Terms | du/dt | d^2u/dx^2 | u*du/dx |
|---|---|---|---|
| Gradient | 1 | -0.044 | 0.952 |
| Adaptive | 1 | -0.045 | 0.951 |
| Polynomial | 1 | -0.058 | 1.057 |
| Spectral | 1 | - | 0.273 |
| Inverse | 1 | -0.134 | 0.205 |
| Total_var | 1 | 1.765 | - |
| Ground truth | 1 | -0.05 | 1 |

Table 18: Coefficients values calculated with SINDy, noise =0.5%

| Methods/Terms | du/dt | d^2u/dx^2 | u*du/dx |
|---|---|---|---|
| Gradient | 1 | -0.044 | 0.955 |
| Adaptive | 1 | -0.045 | 0.951 |
| Polynomial | 1 | -0.055 | 1.039 |
| Spectral | 1 | - | 0.277 |
| Inverse | 1 | - | 0.188 |
| Total_var | 1 | 1.763 | - |
| Ground truth | 1 | -0.05 | 1 |

Table 19: Coefficients values calculated with SINDy, noise =1%

| Methods/Terms | du/dt | d^2u/dx^2 | u*du/dx |
|---|---|---|---|
| Gradient | 1 | - | 0.661 |
| Adaptive | 1 | - | 0.661 |
| Polynomial | 1 | -0.050 | 1.004 |
| Spectral | 1 | - | 0.271 |
| Inverse | 1 | -0.129 | 0.202 |
| Total_var | 1 | 1.736 | -0.014 |
| Ground truth | 1 | -0.05 | 1 |

## E Wave equation coefficients and Structural Hamming Distances

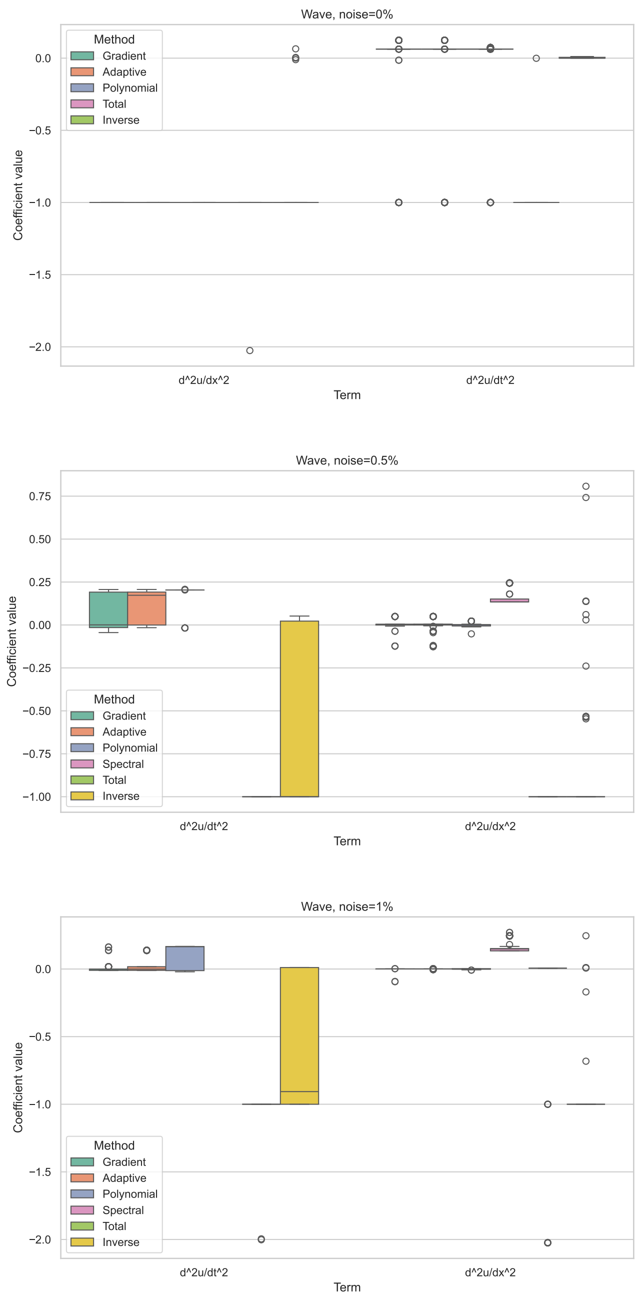

Figure 8: Distribution of coefficients values for different noise level

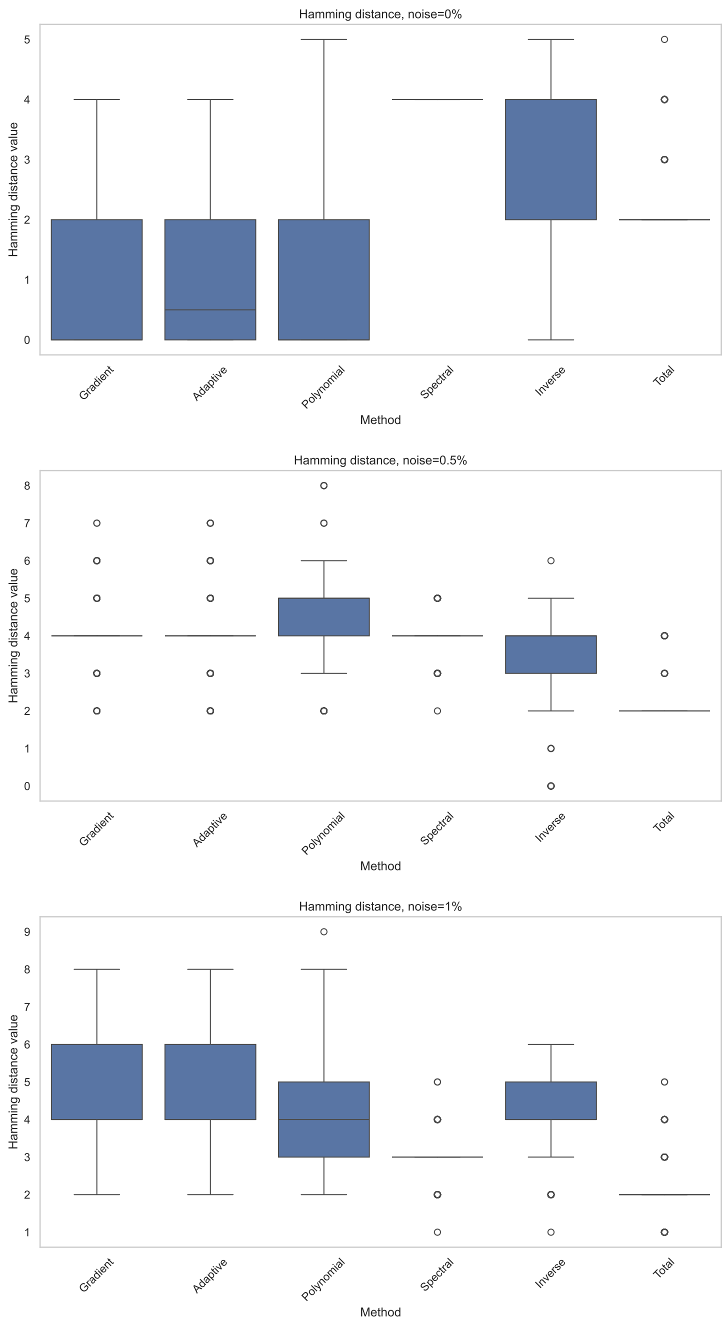

Figure 9: Distribution of coefficients values for different noise level

Table 20: Coefficients values calculated with EPDE, noise =0%

| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 |
|---|---|---|
| Gradient | -1 | -0.0005 ± 0.0338 |
| Adaptive | -1 | -0.0579 ± 0.0408 |
| Polynomial | -1 | -0.0827 ± 0.0430 |
| Spectral | - | - |
| Inverse | -0.9486 ± 0.0502 | 0.0038 ± 0.0162 |
| Total_var | -1.0049 ± 0.0097 | -0.9904 ± 0.0191 |
| Ground truth | 1 | -0.0625 |

Table 21: Coefficients values calculated with EPDE, noise =0.5%

| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 |
|---|---|---|
| Gradient | 0.0037 ± 0.0075 | 0.0688 ± 0.0199 |
| Adaptive | 0.0008 ± 0.0061 | 0.0972 ± 0.0131 |
| Polynomial | 0.0018 ± 0.0030 | 0.1539 ± 0.0345 |
| Spectral | 0.1549 ± 0.0041 | - |
| Inverse | -0.8171 ± 0.1025 | -0.5975 ± 0.1638 |
| Total_var | -1 | -1 |
| Ground truth | 1 | -0.0625 |

Table 22: Coefficients values calculated with EPDE, noise =1%

| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 |
|---|---|---|
| Gradient | -0.0041 ± 0.0068 | 0.0064 ± 0.0070 |
| Adaptive | 0.0004 ± 0.0003 | 0.0108 ± 0.0056 |
| Polynomial | -0.0001 ± 0.0005 | 0.1041 ± 0.0191 |
| Spectral | 0.1533 ± 0.0039 | - |
| Inverse | -0.8946 ± 0.0792 | -0.6434 ± 0.1901 |
| Total_var | -0.3441 ± 0.1927 | -1.0066 ± 0.0092 |
| Ground truth | 1 | -0.0625 |

Table 23: Coefficients values calculated with SINDy, noise =0%

| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 |
|---|---|---|
| Adaptive | 1 | -0.055 |
| Polynomial | 1 | -0.063 |
| Spectral | 1 | - |
| Inverse | 1 | -0.008 |
| Total_var | 1 | -0.007 |
| Ground truth | 1 | -0.0625 |

Table 24: Coefficients values calculated with SINDy, noise =0.5%

| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 |
|---|---|---|
| Gradient | 1 | -0.193 |
| Adaptive | 1 | -0.163 |
| Polynomial | 1 | -0.049 |
| Spectral | 1 | - |
| Inverse | 1 | -0.221 |
| Total_var | 1 | -0.027 |
| Ground truth | 1 | -0.0625 |

Table 25: Coefficients values calculated with SINDy, noise =1%

| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 |
|---|---|---|
| Gradient | 1 | -0.395 |
| Adaptive | 1 | -0.332 |
| Polynomial | 1 | -0.1 |
| Spectral | 1 | - |
| Inverse | 1 | -4.586 |
| Total_var | 1 | -0.079 |
| Ground truth | 1 | -0.0625 |

## F Laplace equation coefficients and Structural Hamming Distances

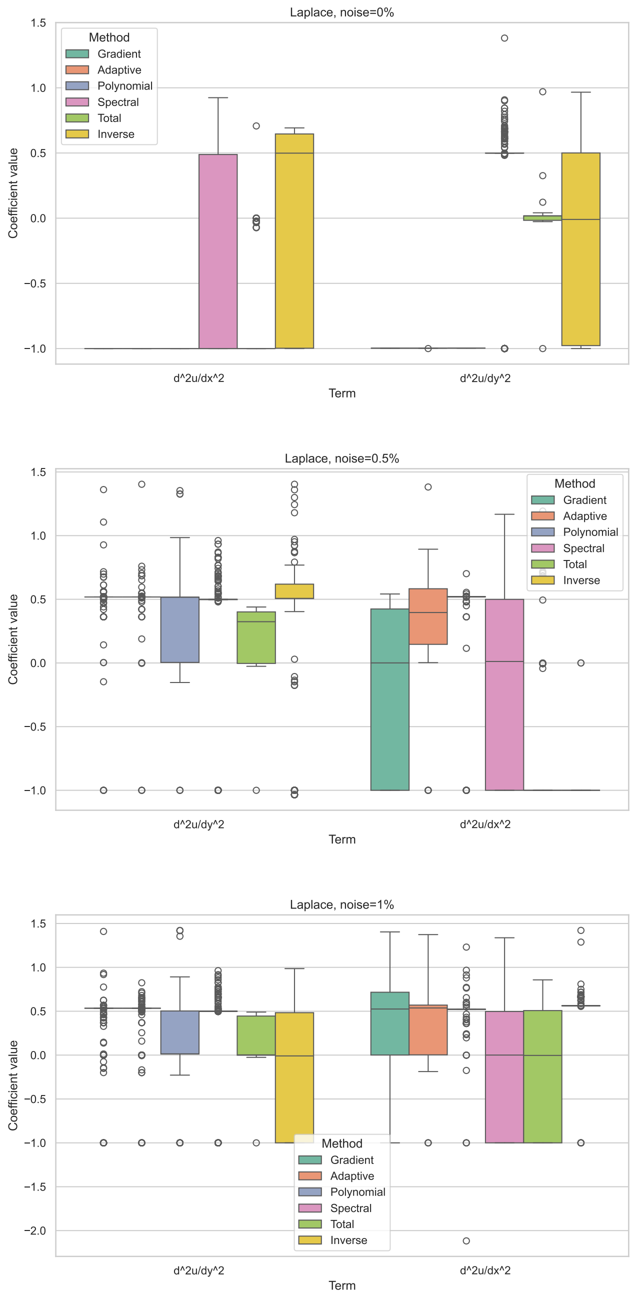

Figure 10: Distribution of coefficients values for different noise level

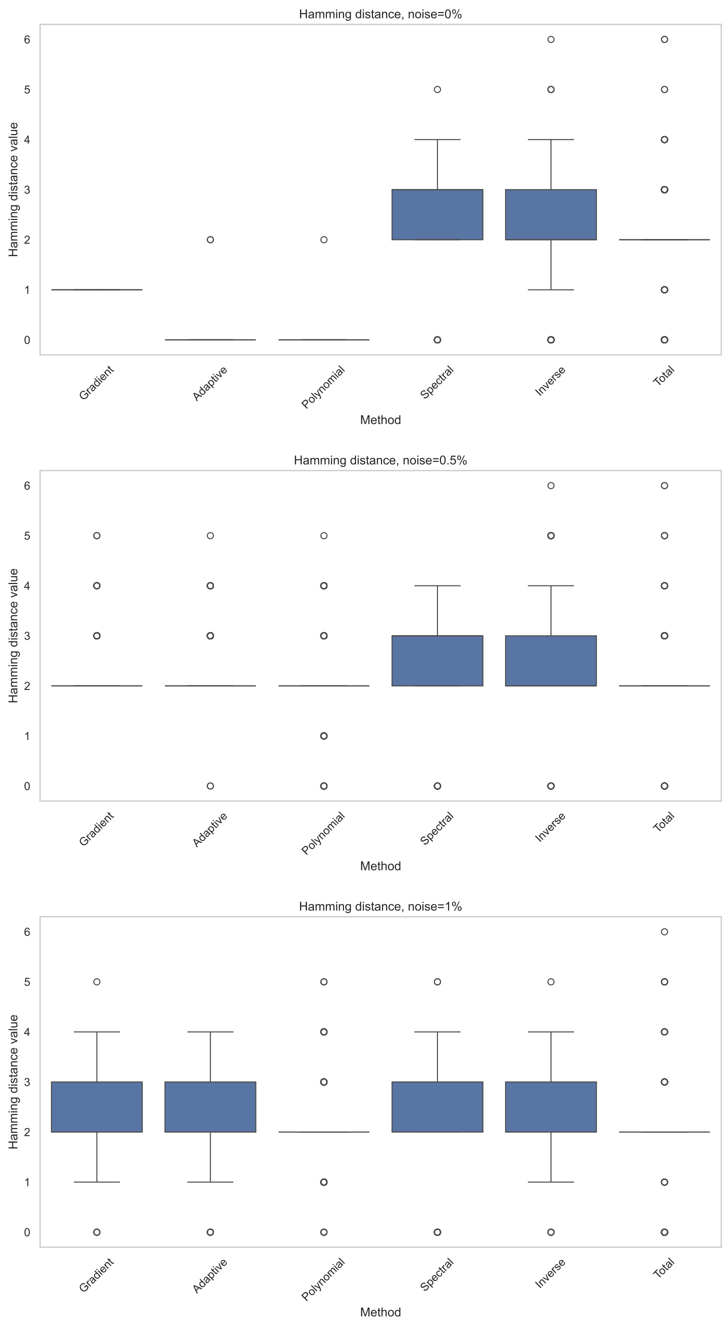

Figure 11: Distribution of coefficients values for different noise level

Table 26: Coefficients values calculated with EPDE, noise =0%

| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 |
|---|---|---|
| Gradient | -1 | -0.997 |
| Adaptive | -1 | -0.997 |
| Polynomial | -1 | -0.9964 |
| Spectral | -1 | -1 |
| Inverse | -0.9985 ± 0.0007 | -0.9955 ± 0.0013 |
| Total_var | -1 | -1 |
| Ground truth | 1 | 1 |

Table 27: Coefficients values calculated with EPDE, noise =0.5%

| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 |
|---|---|---|
| Gradient | 0.2072 ± 0.1540 | 0.5129 ± 0.0095 |
| Adaptive | 0.4755 ± 0.1158 | 0.5150 ± 0.0065 |
| Polynomial | 0.5139 ± 0.0077 | 0.3348 ± 0.0365 |
| Spectral | 0.4728 ± 0.0783 | 0.5393 ± 0.0137 |
| Inverse | -0.0000 ± 0.0025 | 0.5579 ± 0.0239 |
| Total_var | 0.3411 ± 0.1787 | 0.1874 ± 0.0382 |
| Ground truth | 1 | 1 |

Table 28: Coefficients values calculated with EPDE, noise =1%

| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 |
|---|---|---|
| Gradient | 0.4476 ± 0.1022 | 0.4962 ± 0.0192 |
| Adaptive | 0.4047 ± 0.0796 | 0.5056 ± 0.0164 |
| Polynomial | 0.5100 ± 0.0140 | 0.2119 ± 0.0521 |
| Spectral | 0.3388 ± 0.1160 | 0.5333 ± 0.0117 |
| Inverse | 0.5868 ± 0.0061 | 0.3661 ± 0.0714 |
| Total_var | 0.3295 ± 0.0528 | 0.2150 ± 0.0566 |
| Ground truth | 1 | 1 |

Table 29: Coefficients values calculated with SINDy, noise =0%

| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 |
|---|---|---|
| Gradient | 1 | 1.028 |
| Adaptive | 1 | 1.129 |
| Polynomial | 1 | 1.009 |
| Spectral | 1 | - |
| Inverse | 1 | 0.62 |
| Total_var | 1 | - |
| Ground truth | 1 | 1 |

Table 30: Coefficients values calculated with SINDy, noise =0.5%

| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 |
|---|---|---|
| Gradient | 1 | - |
| Adaptive | 1 | - |
| Polynomial | 1 | - |
| Spectral | 1 | - |
| Inverse | 1 | - |
| Total_var | 1 | - |
| Ground truth | 1 | 1 |

Table 31: Coefficients values calculated with SINDy, noise =1%

| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 |
|---|---|---|
| Gradient | 1 | - |
| Adaptive | 1 | -0.457 |
| Polynomial | 1 | - |
| Spectral | 1 | - |
| Inverse | 1 | - |
| Total_var | 1 | - |
| Ground truth | 1 | 1 |

## G Quasigeostrophic potential vorticity equation

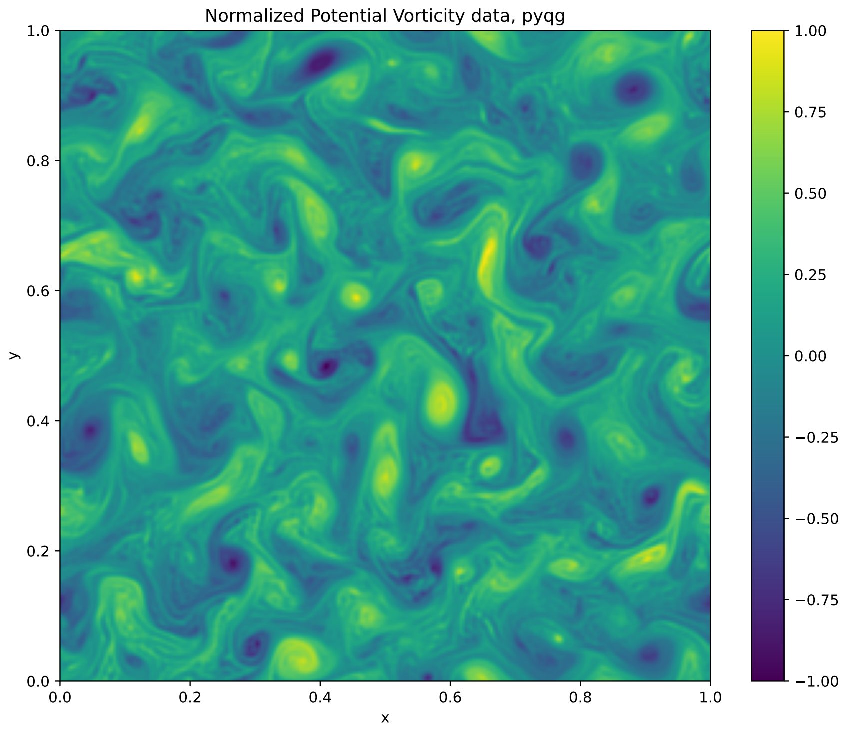

Figure 12: Normalized potential vorticity data, pyqg

Equations discovered:

Via spectral domain differentiation

$$\begin{aligned}3.2602 \times 10^{-6} u_{xx} + 0.0067028u + 0.7095u_y \\ - 0.6485u_y \cos(1.7965y) + 2.5201 \times 10^{-5} uu_{yy} \\ - 0.01010u_x u - 1.2018 \times 10^{-6} u_{xx}u_{yy} \\ + 3.2084 \times 10^{-5} yu_{yy} + 2.4292 \times 10^{-5} u_{xx}u_y \\ + 0.1998u_y \sin(2.7363y) - 0.000223 - yu_y = 0 \end{aligned} \quad (23)$$

Via SG filtering

$$\begin{aligned}0.044994162\, u_x - 5.34527 \times 10^{-5}\, u_{xx} \\ - 0.000760196\, u_x u_{yy} + 0.001192827 - u_x u = 0\end{aligned} \quad (24)$$

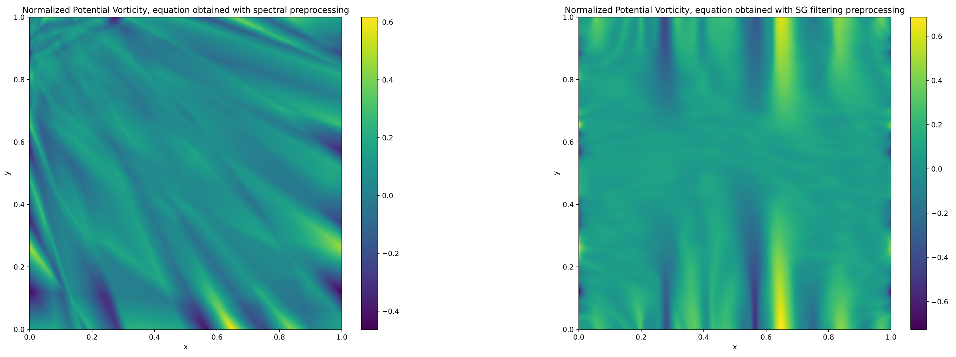

Figure 13: Normalized Potential Vorticity, equation obtained with spectral preprocessing ((23), left) and SG filtering preprocessing ((24), right)

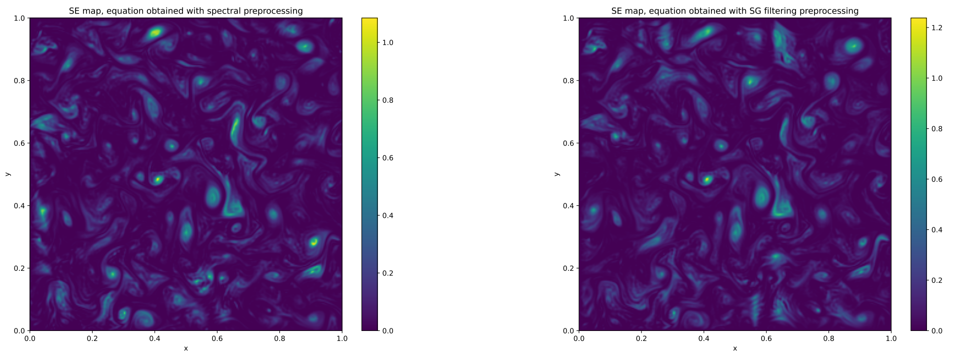

Figure 14: SE map, equation obtained with spectral preprocessing ((23), left, MSE = 0.057) and SG filtering preprocessing((24), right, MSE = 0.065)

## H Structural Hamming Distances

Table 32: SHD for equations calculated with EPDE, noise = 0%

| Methods/Equations | Burgers | KdV | Laplace | ode | Wave |
|---|---|---|---|---|---|
| Gradient | 4 ± 0.091 | 3 ± 0.1302 | 1 | 0 + 0.0491 | 1 ± 0.1206 |
| Adaptive | 6 ± 0.1607 | 3 ± 0.0525 | 0 + 0.0159 | 0 + 0.0636 | 1 ± 0.1189 |
| Polynomial | 4 ± 0.272 | 3 ± 0.1824 | 0 + 0.0112 | 0 + 0.0654 | 1 ± 0.119 |
| Spectral | 6 ± 0.1993 | 6 ± 0.1698 | 3 ± 0.0999 | 3 ± 0.1855 | 4 |
| Inverse | 6 ± 0.3554 | 4 ± 0.1267 | 2 ± 0.1432 | 4 ± 0.2077 | 3 ± 0.1314 |
| Total_var | 6 ± 0.1299 | 4 ± 0.1332 | 2 ± 0.1169 | 0 ± 0.0749 | 2 ± 0.0726 |

Table 33: SHD for equations calculated with EPDE, noise = 0.5%

| Methods/Equations | Burgers | KdV | Laplace | ode | Wave |
|---|---|---|---|---|---|
| Gradient | 5 ± 0.1412 | 5 ± 0.2209 | 2 ± 0.0629 | 0 + 0.0535 | 4 ± 0.0979 |
| Adaptive | 7 ± 0.107 | 3 ± 0.078 | 2 ± 0.0637 | 0 + 0.0458 | 4 ± 0.0671 |
| Polynomial | 4 ± 0.1334 | 5 ± 0.1935 | 2 ± 0.1062 | 0 + 0.0734 | 4 ± 0.1237 |
| Spectral | 6 ± 0.2107 | 7 ± 0.2087 | 3 ± 0.0992 | 3 ± 0.1971 | 4 ± 0.0631 |
| Inverse | 6 ± 0.2107 | 5 ± 0.2632 | 3 ± 0.124 | 4 ± 0.1531 | 3 ± 0.1305 |
| Total_var | 5 ± 0.1413 | 5 ± 0.1394 | 2 ± 0.1084 | 0 + 0.0783 | 2 ± 0.0571 |

Table 34: SHD for equations calculated with EPDE, noise = 1%

| Methods/Equations | Burgers | KdV | Laplace | ode | Wave |
|---|---|---|---|---|---|
| Gradient | 6 ± 0.2134 | 6 ± 0.2824 | 2 ± 0.0843 | 0 + 0.0424 | 5 ± 0.1186 |
| Adaptive | 7 ± 0.1292 | 4 ± 0.1484 | 2 ± 0.0815 | 0 + 0.0433 | 5 ± 0.0833 |
| Polynomial | 4 ± 0.1248 | 6 ± 0.247 | 2 ± 0.102 | 0 + 0.0627 | 4 ± 0.1252 |
| Spectral | 6 ± 0.2096 | 7 ± 0.1827 | 3 ± 0.0974 | 3 ± 0.1904 | 3 ± 0.0687 |
| Inverse | 6 ± 0.2057 | 7 ± 0.2509 | 2 ± 0.0883 | 4 ± 0.1231 | 4 ± 0.1004 |
| Total_var | 6 ± 0.1283 | 5 ± 0.133 | 2 ± 0.1064 | 0 ± 0.0792 | 2 ± 0.0847 |

Table 35: SHD for equations calculated with SINDy, noise = 0%

| Methods/Equations | Burgers | KdV | Laplace | ode | Wave |
|---|---|---|---|---|---|
| Gradient | 0 | 1 | 0 | 0 | 0 |
| Adaptive | 0 | 3 | 0 | 0 | 1 |
| Polynomial | 1 | 3 | 0 | 0 | 1 |
| Spectral | 1 | 2 | 1 | 1 | 1 |
| Inverse | 1 | 1 | 1 | 0 | 1 |
| Total_var | 5 | 4 | 2 | 0 | 2 |

Table 36: SHD for equations calculated with SINDy, noise = 0.5%

| Methods/Equations | Burgers | KdV | Laplace | ode | Wave |
|---|---|---|---|---|---|
| Gradient | 0 | 2 | 2 | 0 | 2 |
| Adaptive | 0 | 2 | 2 | 0 | 2 |
| Polynomial | 1 | 2 | 2 | 0 | 2 |
| Spectral | 1 | 2 | 2 | 1 | 1 |
| Inverse | 3 | 3 | 4 | 0 | 1 |
| Total_var | 5 | 4 | 2 | 0 | 2 |

Table 37: SHD for equations calculated with SINDy, noise = 1%

| Methods/Equations | Burgers | KdV | Laplace | ode | Wave |
|---|---|---|---|---|---|
| Gradient | 2 | 2 | 3 | 0 | 2 |
| Adaptive | 2 | 2 | 1 | 0 | 2 |
| Polynomial | 0 | 5 | 4 | 0 | 2 |
| Spectral | 1 | 2 | 2 | 1 | 1 |
| Inverse | 1 | 4 | 4 | 0 | 2 |
| Total_var | 3 | 4 | 2 | 0 | 2 |

## I Differentiation errors

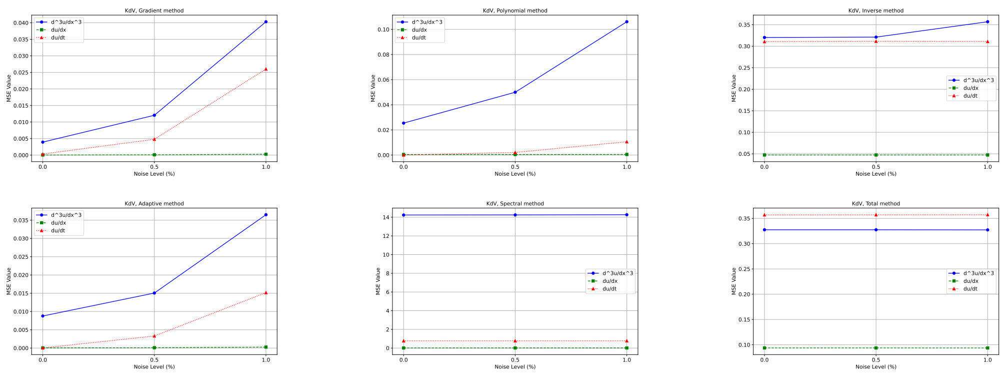

Figure 15: Differentiation errors (MSE) for KdV equation with different noise level

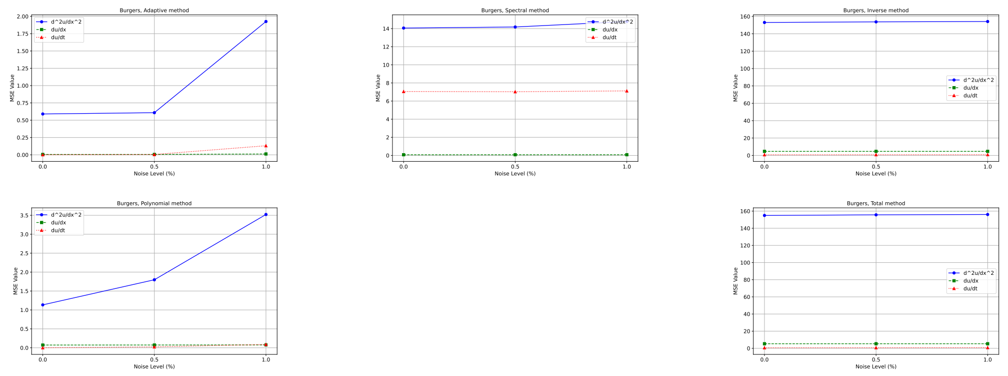

Figure 16: Differentiation errors (MSE) for Burgers equation with different noise level

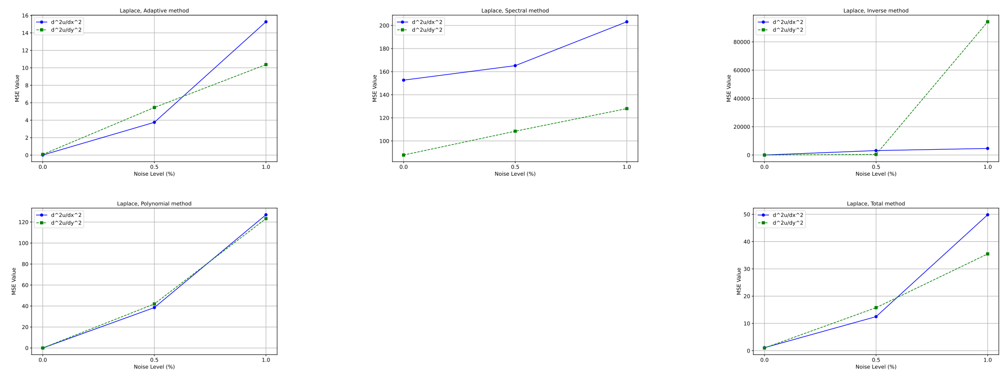

Figure 17: Differentiation errors (MSE) for Laplace equation with different noise level

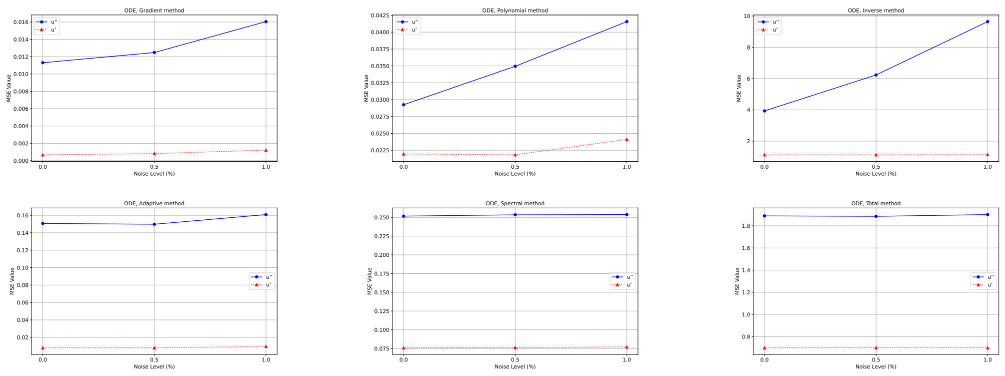

Figure 18: Differentiation errors (MSE) for ODE equation with different noise level

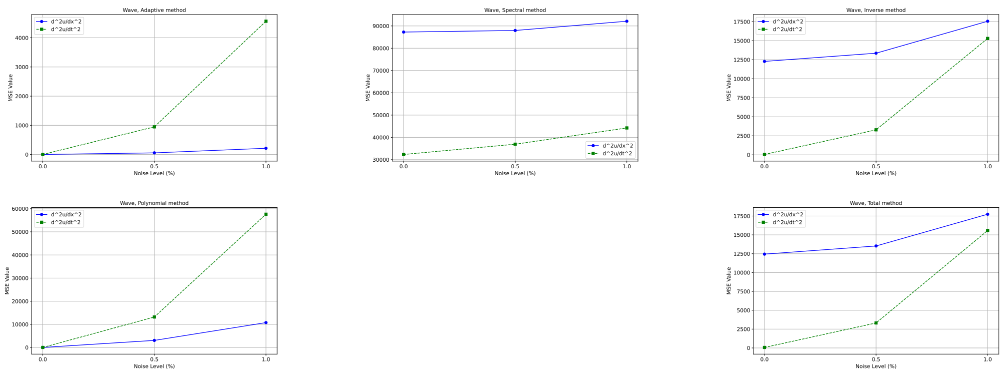

Figure 19: Differentiation errors (MSE) for Wave equation with different noise level

Table 38: Differentiation errors, noise = 0%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0009 | 0.05902 | 0.0053 |
| Polynomial | 0.0012 | 1.1339 | 0.0724 |
| Spectral | 7.0597 | 14.0607 | 0.0757 |
| Inverse | 0.7638 | 153.0405 | 4.6508 |
| Total_var | 0.6760 | 154.9538 | 5.3229 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.000329 | 0.0039 | 0.00004184 |
| Adaptive | 0.00001 | 0.0088 | 0.00005 |
| Polynomial | 0.00007761 | 0.0254 | 0.0004532 |
| Spectral | 0.7798 | 14.2334 | 0.0129 |
| Inverse | 0.3105 | 0.3202 | 0.0473 |
| Total_var | 0.357 | 0.3275 | 0.0937 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 0.0124 | 0.0814 | |
| Polynomial | 0.0058 | 0.0036 | |
| Spectral | 152.6378 | 87.813 | |
| Inverse | 0.0098 | 0.178 | |
| Total_var | 1.0941 | 0.9853 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.00067 | 0.0113 | |
| Adaptive | 0.00801 | 0.1507 | |
| Polynomial | 0.022 | 0.0292 | |
| Spectral | 0.0759 | 0.2517 | |
| Inverse | 1.1231 | 3.9290 | |
| Total_var | 0.6988 | 1.8895 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 2.0602 | 1.2876 | |
| Polynomial | 9.8417 | 0.0057 | |
| Spectral | 87251.4516 | 32328.7991 | |
| Inverse | 12282.1612 | 47.9087 | |
| Total_var | 12444.1383 | 65.6059 | |

Table 39: Differentiation errors, noise = 0.5%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0048 | 0.6084 | 0.0053 |
| Polynomial | 0.0234 | 1.7973 | 0.0737 |
| Spectral | 7.0293 | 14.1771 | 0.0761 |
| Inverse | 0.8013 | 153.7448 | 4.6542 |
| Total_var | 0.7139 | 155.6498 | 5.3271 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.00478 | 0.0120 | 0.0000947 |
| Adaptive | 0.0033 | 0.0151 | 0.0001 |
| Polynomial | 0.0021 | 0.05 | 0.0004592 |
| Spectral | 0.7804 | 14.2418 | 0.0128 |
| Inverse | 0.3114 | 0.3211 | 0.0473 |
| Total_var | 0.3571 | 0.3275 | 0.0937 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 3.7555 | 5.4563 | |
| Polynomial | 38.4966 | 41.9891 | |
| Spectral | 165.1737 | 108.4259 | |
| Inverse | 3138.8939 | 426.1978 | |
| Total_var | 12.4789 | 15.7744 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.00081 | 0.0124 | |
| Adaptive | 0.008 | 0.1498 | |
| Polynomial | 0.0218 | 0.0349 | |
| Spectral | 0.0764 | 0.2534 | |
| Inverse | 1.1220 | 6.2318 | |
| Total_var | 0.6998 | 1.8857 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 56.2421 | 947.3597 | |
| Polynomial | 3047.6637 | 13167.1760 | |
| Spectral | 87963.7107 | 36937.9071 | |
| Inverse | 13359.6413 | 3287.3561 | |
| Total_var | 13524.8049 | 3315.1318 | |

Table 40: Differentiation errors, noise = 1%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.1311 | 1.9259 | 0.0121 |
| Polynomial | 0.088 | 3.5219 | 0.0761 |
| Spectral | 7.1244 | 14.7814 | 0.0783 |
| Inverse | 0.8933 | 154.1399 | 4.6562 |
| Total_var | 0.8085 | 156.0558 | 5.3286 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0260 | 0.0403 | 0.0002695 |
| Adaptive | 0.01521 | 0.0365 | 0.0003 |
| Polynomial | 0.0106 | 0.1061 | 0.000499 |
| Spectral | 0.7811 | 14.2637 | 0.0129 |
| Inverse | 0.3108 | 0.3568 | 0.0473 |
| Total_var | 0.3573 | 0.3273 | 0.0936 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 15.2715 | 10.3725 | |
| Polynomial | 126.9283 | 123.2955 | |
| Spectral | 203.1654 | 128.0034 | |
| Inverse | 4647.4401 | 94205.9511 | |
| Total_var | 49.8341 | 35.4761 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.00122 | 0.0160 | |
| Adaptive | 0.0096 | 0.1609 | |
| Polynomial | 0.0241 | 0.0416 | |
| Spectral | 0.0771 | 0.2538 | |
| Inverse | 1.132 | 9.6419 | |
| Total_var | 0.6993 | 1.9013 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 213.6322 | 4567.2138 | |
| Polynomial | 10740.0809 | 57619.2452 | |
| Spectral | 92107.8232 | 44231.8631 | |
| Inverse | 17561.9236 | 15300.6987 | |
| Total_var | 17738.1271 | 15580.1558 | |

Table 41: Differentiation errors without boundaries, noise = 0%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0008 | 0.7898 | 0.0061 |
| Polynomial | 0.001 | 1.3151 | 0.082 |
| Spectral | 0.2112 | 6.3855 | 0.0156 |
| Inverse | 0.8751 | 206.6064 | 5.4783 |
| Total_var | 0.7988 | 208.9618 | 6.2428 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0004 | 0.0050 | 0.00005 |
| Adaptive | 0.0000008 | 0.0121 | 0.00005 |
| Polynomial | 0.000009 | 0.0350 | 0.0006 |
| Spectral | 0.0934 | 14.1910 | 0.0155 |
| Inverse | 0.4196 | 0.4601 | 0.0638 |
| Total_var | 0.357 | 0.3273 | 0.0936 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 0.0018 | 0.0019 | |
| Polynomial | 0.0019 | 0.0018 | |
| Spectral | 26.536 | 14.1521 | |
| Inverse | 0.0017 | 0.0079 | |
| Total_var | 0.4919 | 0.4864 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.0002 | 0.0027 | |
| Adaptive | 0.0012 | 0.0155 | |
| Polynomial | 0.0128 | 0.019 | |
| Spectral | 0.0056 | 0.0687 | |
| Inverse | 0.9548 | 3.5607 | |
| Total_var | 0.617 | 1.7065 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 4.6496e-25 | 3.9716e-25 | |
| Polynomial | 0.0020 | 3.1118e-08 | |
| Spectral | 3975.5779 | 3283.7298 | |
| Inverse | 15075.2208 | 59.1723 | |
| Total_var | 15318.4098 | 68.1121 | |

Table 42: Differentiation errors without boundaries, noise = 0.5%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0009 | 0.7936 | 0.0061 |
| Polynomial | 0.0206 | 1.3864 | 0.0832 |
| Spectral | 0.2275 | 6.4897 | 0.0162 |
| Inverse | 0.9018 | 207.6153 | 5.4904 |
| Total_var | 0.8239 | 209.9772 | 6.2568 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0048 | 0.017 | 0.0001 |
| Adaptive | 0.0041 | 0.0221 | 0.0001 |
| Polynomial | 0.0009 | 0.0366 | 0.0006 |
| Spectral | 0.0934 | 14.1966 | 0.0155 |
| Inverse | 0.4195 | 0.4617 | 0.0638 |
| Total_var | 0.3571 | 0.3275 | 0.0937 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 0.0111 | 0.0115 | |
| Polynomial | 0.5938 | 0.5695 | |
| Spectral | 28.7692 | 17.0705 | |
| Inverse | 72.7557 | 2.4132 | |
| Total_var | 2.932 | 3.0227 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.0002 | 0.0024 | |
| Adaptive | 0.0013 | 0.0154 | |
| Polynomial | 0.0129 | 0.0188 | |
| Spectral | 0.0055 | 0.0694 | |
| Inverse | 0.952 | 3.4523 | |
| Total_var | 0.6166 | 1.6975 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 4.4623e-25 | 4.4207e-25 | |
| Polynomial | 296.3631 | 297.0768 | |
| Spectral | 5250.7934 | 4299.9723 | |
| Inverse | 16376.1646 | 1341.7407 | |
| Total_var | 16622.2915 | 1353.3664 | |

Table 43: Differentiation errors without boundaries, noise = 1%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.1373 | 2.3416 | 0.0146 |
| Polynomial | 0.0474 | 2.5502 | 0.0868 |
| Spectral | 0.2652 | 7.2152 | 0.02 |
| Inverse | 0.9020 | 207.6163 | 5.4917 |
| Total_var | 0.8236 | 209.9705 | 6.2560 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0187 | 0.0502 | 0.0003657 |
| Adaptive | 0.0171 | 0.0495 | 0.0003 |
| Polynomial | 0.0034 | 0.039 | 0.00068 |
| Spectral | 0.0941 | 14.2271 | 0.0154 |
| Inverse | 0.4198 | 0.4931 | 0.0638 |
| Total_var | 0.3573 | 0.3275 | 0.0937 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 0.0458 | 0.0432 | |
| Polynomial | 2.6708 | 2.6109 | |
| Spectral | 38.1629 | 25.7555 | |
| Inverse | 46.8684 | 47.1770 | |
| Total_var | 12.0801 | 11.3885 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.0003 | 0.0036 | |
| Adaptive | 0.0013 | 0.0158 | |
| Polynomial | 0.0128 | 0.0202 | |
| Spectral | 0.0057 | 0.0694 | |
| Inverse | 0.9607 | 3.1547 | |
| Total_var | 0.6155 | 1.7163 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 4.4999e-25 | 4.9763e-25 | |
| Polynomial | 1098.5015 | 1260.9454 | |
| Spectral | 8281.9373 | 9248.8831 | |
| Inverse | 19982.6044 | 6108.153 | |
| Total_var | 20238.9430 | 6130.2836 | |

Table 44: Means of diff errors without boundaries, noise=0%

| Methods | D1. error | D2. error | D3. error |
|---|---|---|---|
| Gradient | 0.0002 | 0.0027 | 0.0050 |
| Adaptive | 0.0016 | 0.2023 | 0.0121 |
| Polynomial | 0.0193 | 0.3344 | 0.0350 |
| Spectral | 0.0683 | 11.7856 | 14.1910 |
| Inverse | 1.5583 | 52.5442 | 0.4601 |
| Total_var | 1.6218 | 52.9117 | 0.3273 |

Table 45: Means of diff errors without boundaries, noise=0.5%

| Methods | D1. error | D2. error | D3.error |
|---|---|---|---|
| Gradient | 0.0017 | 0.0024 | 0.017 |
| Adaptive | 0.0025 | 0.2079 | 0.0221 |
| Polynomial | 0.0236 | 0.6421 | 0.0366 |
| Spectral | 0.0716 | 13.0997 | 14.1966 |
| Inverse | 1.5655 | 71.5591 | 0.4617 |
| Total_var | 1.6292 | 54.4074 | 0.3275 |

Table 46: Means of diff errors without boundaries, noise=1%

| Methods | D1. error | D2. errors | D3. errors |
|---|---|---|---|
| Gradient | 0.0065 | 0.0036 | 0.0502 |
| Adaptive | 0.0341 | 0.6116 | 0.0432 |
| Polynomial | 0.0302 | 1.9630 | 0.039 |
| Spectral | 0.0800 | 17.8008 | 14.2271 |
| Inverse | 1.5676 | 76.2041 | 0.4931 |
| Total_var | 1.6296 | 58.7889 | 0.3275 |

Table 47: Differentiation errors on boundaries, noise = 0%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0010 | 0.4113 | 0.0047 |
| Polynomial | 0.0014 | 0.9714 | 0.0639 |
| Spectral | 13.1979 | 20.9399 | 0.1296 |
| Inverse | 0.6641 | 105.0296 | 3.9091 |
| Total_var | 0.5659 | 106.5467 | 4.4983 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0002 | 0.0030 | 0.00004 |
| Adaptive | 0.00005 | 0.0058 | 0.00005 |
| Polynomial | 0.0001 | 0.0168 | 0.0003 |
| Spectral | 1.3950 | 14.2714 | 0.0105 |
| Inverse | 0.2128 | 0.1948 | 0.0325 |
| Total_var | 0.2416 | 0.2002 | 0.0687 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 0.0219 | 0.1528 | |
| Polynomial | 0.0093 | 0.0052 | |
| Spectral | 265.6624 | 153.8350 | |
| Inverse | 0.0170 | 0.3305 | |
| Total_var | 1.6339 | 1.4325 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.0017 | 0.0302 | |
| Adaptive | 0.0230 | 0.4480 | |
| Polynomial | 0.0422 | 0.0518 | |
| Spectral | 0.2305 | 0.6544 | |
| Inverse | 1.1393 | 4.7208 | |
| Total_var | 0.8787 | 2.2920 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 3.9068 | 2.4417 | |
| Polynomial | 0.0009 | 1.3527e-08 | |
| Spectral | 4330.2841 | 4791.3391 | |
| Inverse | 6552.9687 | 25.7213 | |
| Total_var | 9867.9392 | 663.3597 | |

Table 48: Differentiation errors on boundaries, noise = 0.5%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0082 | 0.4425 | 0.0046 |
| Polynomial | 0.0259 | 2.1655 | 0.0651 |
| Spectral | 13.1257 | 21.0673 | 0.1299 |
| Inverse | 0.7112 | 105.4608 | 3.9048 |
| Total_var | 0.6153 | 106.9564 | 4.4938 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0047 | 0.0076 | 0.00006 |
| Adaptive | 0.0026 | 0.0088 | 0.00007 |
| Polynomial | 0.0031 | 0.0620 | 0.0003 |
| Spectral | 1.3961 | 14.2823 | 0.0105 |
| Inverse | 0.2145 | 0.1951 | 0.0324 |
| Total_var | 0.2418 | 0.2001 | 0.0688 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 7.1115 | 10.3365 | |
| Polynomial | 72.4688 | 79.1133 | |
| Spectral | 287.4326 | 190.3073 | |
| Inverse | 5887.0620 | 806.0344 | |
| Total_var | 21.0357 | 27.2037 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.0021 | 0.0345 | |
| Adaptive | 0.0227 | 0.4455 | |
| Polynomial | 0.0415 | 0.0704 | |
| Spectral | 0.2324 | 0.6584 | |
| Inverse | 1.1366 | 12.4026 | |
| Total_var | 0.8829 | 2.2998 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 106.6517 | 1796.4747 | |
| Polynomial | 83.3851 | 130.6980 | |
| Spectral | 5164.4719 | 5031.0763 | |
| Inverse | 7255.2930 | 433.5565 | |
| Total_var | 10748.5392 | 5073.4548 | |

Table 49: Differentiation errors on boundaries, noise = 1%

| | Burgers equation | | |
|---|---|---|---|
| Methods/Terms | du/dt | d^2u/dx^2 | du/dx |
| Adaptive | 0.0241 | 0.5434 | 0.0048 |
| Polynomial | 0.0740 | 5.2326 | 0.0660 |
| Spectral | 13.2611 | 21.4309 | 0.1311 |
| Inverse | 0.8143 | 106.2819 | 3.9179 |
| Total_var | 0.7134 | 107.7962 | 4.5084 |
| | KdV equation | | |
| Methods/Terms | du/dt | d^3u/dx^3 | du/dx |
| Gradient | 0.0326 | 0.0315 | 0.0002 |
| Adaptive | 0.0134 | 0.0248 | 0.0002 |
| Polynomial | 0.0170 | 0.1662 | 0.0003 |
| Spectral | 1.3969 | 14.2965 | 0.0105 |
| Inverse | 0.2131 | 0.2346 | 0.0326 |
| Total_var | 0.2419 | 0.2002 | 0.0687 |
| | Laplace equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dy^2 | |
| Adaptive | 28.9184 | 19.6306 | |
| Polynomial | 238.2998 | 231.4645 | |
| Spectral | 351.0565 | 219.6478 | |
| Inverse | 8770.9156 | 178600.1116 | |
| Total_var | 83.6729 | 57.0657 | |
| | ODE equation | | |
| Methods/Terms | u’ | u” | |
| Gradient | 0.0033 | 0.0429 | |
| Adaptive | 0.0278 | 0.4711 | |
| Polynomial | 0.0498 | 0.0922 | |
| Spectral | 0.2334 | 0.6601 | |
| Inverse | 1.1606 | 23.7757 | |
| Total_var | 0.8796 | 2.2926 | |
| | Wave equation | | |
| Methods/Terms | d^2u/dx^2 | d^2u/dt^2 | |
| Adaptive | 405.1010 | 8660.7907 | |
| Polynomial | 665.12100 | 472.6915 | |
| Spectral | 6266.4304 | 7554.8468 | |
| Inverse | 8576.6807 | 2610.7530 | |
| Total_var | 15496.6550 | 24050.0412 | |

Table 50: Means of diff errors on boundaries, noise=0%

| Methods | D1. error | D2. error | D3. error |
|---|---|---|---|
| Gradient | 0.0017 | 0.0024 | 0.0030 |
| Adaptive | 0.0058 | 0.2585 | 0.0058 |
| Polynomial | 0.0216 | 0.2594 | 0.0168 |
| Spectral | 2.9927 | 110.2729 | 14.2714 |
| Inverse | 1.1916 | 27.5245 | 0.1948 |
| Total_var | 1.2506 | 27.9763 | 0.2002 |

Table 51: Means of diff errors on boundaries, noise=0.5%

| Methods | D1. error | D2. error | D3.error |
|---|---|---|---|
| Gradient | 0.0023 | 0.0345 | 0.0076 |
| Adaptive | 0.0076 | 4.584 | 0.0088 |
| Polynomial | 0.0272 | 30.7636 | 0.0620 |
| Spectral | 2.9789 | 124.8664 | 14.2823 |
| Inverse | 1.1999 | 1702.7400 | 0.1951 |
| Total_var | 1.2605 | 39.3739 | 0.2001 |

Table 52: Means of diff errors on boundaries, noise=1%

| Methods | D1. error | D2. errors | D3. errors |
|---|---|---|---|
| Gradient | 0.0120 | 0.0429 | 0.0315 |
| Adaptive | 0.0141 | 12.3909 | 0.0248 |
| Polynomial | 0.0414 | 118.7723 | 0.1662 |
| Spectral | 3.7583 | 148.1988 | 14.2965 |
| Inverse | 1.2277 | 44900.2712 | 0.2346 |
| Total_var | 1.2824 | 62.7069 | 0.2002 |