Partial differential equations discovery with EPDE framework: application for real and synthetic data

Mikhail Maslyaev, Alexander Hvatov*, Anna V. Kalyuzhnaya

ITMO University, 49 Kronverksky Pr. St. Petersburg, 197101, Russian Federation

Abstract

Data-driven methods provide model creation tools for systems where the application of conventional analytical methods is restrained. The proposed method involves the data-driven derivation of a partial differential equation (PDE) for process dynamics, helping process simulation and study. The paper describes the methods that are used within the EPDE (Evolutionary Partial Differential Equations) partial differential equation discovery framework [1]. The framework involves a combination of evolutionary algorithms and sparse regression. Such an approach is versatile compared to other commonly used data-driven partial differential derivation methods by making fewer assumptions about the resulting equation. This paper highlights the algorithm features that allow data processing with noise, which is similar to the algorithm's real-world applications.

This paper is an extended version of the ICCS-2020 conference paper [2]

Keywords: data-driven modelling, PDE discovery, evolutionary algorithms, sparse regression, spatial fields, physical measurement data

1. Introduction

The ability to simulate complex processes, neglecting a lack of knowledge about the system's underlying structure, can be vital for developing models in such spheres of science as biology, medicine, materials technology, and metocean studies. In contrast to the deterministic physics-based models, developed

*Corresponding author: alex_hvatov@itmo.ru

by application of conservation laws to the studied process, data-driven modeling (DDM) involves developing complete models from various fields of measurements, describing the process, using means of statistics and machine learning algorithms. Moreover, in some occasions, DDM can enhance the existing physics-based models with supplementary expressions or refined weight values [3]. In fluid dynamics science and hydrometeorology, surrogate models' development is the most common application of data-driven algorithms.

In the current paper's scope are the methods of data-driven differential equation discovery. Differential equations, in some cases, are interpretable by the expert either in the application field or in the differential equations. Moreover, the well-developed mathematical physics methods for the differential equations analysis may interpret the equations. In most cases, actual algorithms utilize the sparse regression in a prescribed differential terms library [4, 5]. The second popular case of the study is the neural network's algorithms for differential equations discovery [6, 7, 8].

We consider discovered models as the surrogate models that could be applied to the hydrometeorological examples. Various approaches to surrogate modeling are described below, including differential equations discovery.

The modern surrogate models tend to belong to one of 3 major groups [9]:

- Data-driven empirical approximations of the deterministic model outputs. These models use conclusions obtained with the statistical or machine learning tools (response surfaces, kriging) applied to the data.
- Reduced-order models are based on the projection of the model's main equations to the subspace with the reduced dimensionality, using various orthogonal decompositions.
- Multifidelity models: simplifications of representing the complex physics of the model's process by omitting the less significant subprocesses or increasing the model's scale. In some cases, the experimental setup requires applying models with different fidelity levels to evaluate multiple scales of processes or modeling ensemble [10, 11].

In this research, we are interested in developing a new approach that belongs to the first class of models. However, natural sciences applications require robustness of the model and should work in high-dimensional space to handle spatio-temporal and other types of variability. Transferring from one spatial dimension usually considered in references to higher spatial dimensions requires the algorithm to handle exponentially growing noise levels.

In the previous works [12] we have described the EPDE (Evolutionary Partial Differential Equations)[1] approach, that can provide a flexible, yet efficient tool for data-driven equation derivation. This work increases the problem's difficulty by introducing higher-dimensional cases and high-magnitude noise in the data.

This version extends conference paper [2] and introduces a series of experiments that allow comparing EPDE framework with the analogs in a better way. The module system of the PDE algorithm that is briefly described in Sec. 6 allows to, as an example, use different from the finite-difference differentiation scheme. We show it using neural networks and automatic derivatives in Sec. 7.

This paper is organized as follows: Sec. 2 briefly introduces the existing surrogate modeling approaches. Sec. 3 describes the problem of the data-driven PDE discovery and Sec. 4 describes the practical realization. In Sec. 5, numerical examples of the synthetic data and the real data are shown. Sec. 6 presents the additions to the method described in the previous article [12], which allows dealing with the higher-dimension data-driven PDE discovery. Sec.7 is dedicated to illustrating the module structure and experiments with replacement of differentiation model with neural network approximation. Sec. 8 concludes the paper.

2. Related work

The first examples of the data-driven surrogate modeling in hydrometeorology have appeared in its earliest stages with the understanding, that the

[1]The approach described in the article is available as stand-alone EPDE-framework in GitHub [1].

contemporary full-scale models required computational powers, inaccessible for many research teams. The original approaches were based on the pattern scaling - the extension of the present trend, obtained from the ensemble of full-scale models [13, 14]. The statistical emulation on the base of an ensemble of pre-computed deterministic models has been developed in [15]. The recent advancements have been achieved in the area of deep learning methods [16]. While being relatively successful in their forecasting abilities, the models above do not consider any knowledge about the processes' physics, and due to a large number of assumptions, it may lead to substantial errors.

Furthermore, the proposed method could be applied to the unstudied systems as a way to model them. Many systems across all spheres of science lack the study to be adequately described by analytical models. The proposed equation-based method may provide a surrogate model to simulate the system and an insight into its dynamics.

This article describes the first step of the creation of the differential equation-based surrogate modeling method. Here we propose only the element of the equation derivation, avoiding the problem of forecasting.

The problem of data-driven discovery of partial differential equations, which plays a significant role in our modeling scheme, has seen an increasing relevance and research interest in recent years. The sparse regression presents the first class of the developed algorithms of data-driven partial differential equation derivation. It is applied to the libraries of possible equation terms to approximate the time derivative with the selected terms, required to describe the examined process, and calculate real-valued coefficients for them. The notable examples of this approach are presented in [17, 18]. In [19], the same idea was extended to the discovery of an equation with non-constant (time-dependent) coefficients.

The concept of numerical Gaussian processes, developed in [20], views the discretized equation as the Gaussian process and obtains the equation's unknown coefficients with maximum likelihood estimation. However, the class of the equations explored in the research is limited by the linear partial differential

equations.

Artificial neural networks provide a more versatile tool. This method is based on the approximation of time derivative with combinations of spatial derivatives and other functions. The ANN applications' examples to the problem of partial differential equation discovery were presented in [8, 21, 22, 7, 6]. While artificial neural networks can discover non-linear equations, they still rely on approximating a determined term (time derivative of the first order), limiting their flexibility.

3. Problem statement

The class of problems, which the described EPDE algorithm can solve, can be summarized as follows: the process, which involves scalar field u, is occurring in the area Ω and is governed by the partial differential equation Eq. 1. However, there is no a priori information about the dynamics of the process except that some form of PDE can describe it (for simplicity, we consider temporally varying 2D field case, even though the problem could be formulated for an arbitrary field). In recent developments, we have abandoned the assumption of the constant weights in the partial differential equations, allowing them to be an arbitrary function (logarithmic, trigonometric) and thus expanding the class of possible systems to study.

$$\begin{cases} F(u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, ..., \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_2^2}, ..., \frac{\partial^2 u}{\partial t^2}, ..., \mathbf{x}) = 0; \\ G(\mathbf{x}) = 0, \ \mathbf{x} \in \Gamma(\Omega) \times [0, T]; \end{cases} \tag{1}$$

From the area $\Omega \times [0, T]$ a set of samples $\mathrm{U} = \{u_1, u_2, ..., u_n\}$, where $u_i = u(x_1^{(i)}, x_2^{(i)}, t_i)$ is the function value at the arbitrary point $(x_1^{(i)}, x_2^{(i)}, t_i) \in \Omega \times [0, T]$, is collected. There are no strict limitations for distributing the sample collection points in the area, but the further requirements of the derivative calculations make the case of stationary points located on the grid the most preferable. The main task of the algorithm is the derivation of the Eq. 1, using measurements from the set of discrete measurements U with some externally

defined limitations, including a range of the derivative orders, several terms in the equations, and some factors in the term.

The resulting model Eq. 2 takes form of linear combination of terms, where each of them $t(\mathbf{x}) = \prod_{j=1}^{N_{tokens}} \phi_j$ (with N_{tokens} is pre-defined algorithm hyper-parameter) is constructed as the product of pre-computed elementary operators ϕ_j, selected from a different groups of elementary operators of a same nature. More detailed elementary operator $\phi_i \in \Phi = \cup_j \Phi_j$; Φ_j - a group of elementary operators of a same nature (for example, trigonometric group $\Phi_j = \{\sin(x_1), \cos(x_1), \sin(x_2), ...\}$, or differential operators group $\Phi_j = \{u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, ...\}$).

$$F(\mathbf{x}) = \sum_i c_i t_i(\mathbf{x}) = 0; \tag{2}$$

The addition of different groups Φ_j of terms allows to switch from the differential equation with the constant coefficients to differential equations with the variable coefficients.

The noise in this paper is assumed to be directional. The noise E_{x_j} in the direction x_j can be described as Eq. 3.

$$E_{x_j}(u(x_1, ..., x_n; t)) = u(\bar{x}_1, ..., x_j, ...\bar{x}_n; t) + \epsilon(x_j) \tag{3}$$

With $\bar{x}_i$ "fixed" variables are denoted and $\epsilon(x_j)$ is the noise, which in the paper is assumed to be distributed normally $N(0; \sigma)$ with the expected value of 0 and the variance of σ. It should be emphasized that poly-directional noise forms as the superposition of the unidirectional noise operators, i.e. $E_{x_j,x_k} = E_{x_j} \circ E_{x_k}$. In what follows $\bar{\sigma} = \sigma \max(\mathrm{u})$ is chosen as the multiplier of maximal magnitude of the measured value in this direction. The noise level is defined in the same way. In the text below bar over the variance, σ is omitted.

4. Method description

In this section, the details of the evolutionary method of partial differential equation derivation are described. The proposed method involves a combination of evolutionary algorithms and sparse regression to detect the equation

structure. The sparse regression aims to construct equation terms set, while the evolutionary algorithm is focused on selecting significant terms from the created set and calculating weights that will be present in the resulting equation. At first, we introduce the preprocessing pipeline while later describe the algorithm workflow.

4.1. Data preprocessing

To initialize the algorithm, time and spatial derivatives, which will later form the desired equation, must be calculated. In specific situations, the derivatives by themselves can be measured, and, therefore, this step can be skipped, but often only the raw value of the studied function is available in the research. It can be assumed without losing the generality that the measurements are held on the rectangular (but not necessarily uniform) grid for the more straightforward further computations. The multi-dimensional case requires more nuanced methods of obtaining derivatives, unlike the instances of a single dimension. In most of the one-dimensional experiments, even on moderate noise levels, which can be measured as Eq. 4, the finite-difference method of derivative calculation can lead to satisfactory results. It is important to note that the taken derivatives' quality is crucial for acquiring the equation's correct structure.

$$Q_{noise} = \frac{\|u_0 - \widetilde{u}\|_2}{\|u_0\|_2} * 100\% \tag{4}$$

In general, the calculation of derivatives is the operation that is vulnerable to the data's noise. Also, the convergence of the algorithm and the resulting equation depends on the input derivatives' quality. If they are computed with high errors, the resulting equation's alterations can vary from incorrect coefficients to the entirely wrong structure. For these reasons, several noise-resistant methods of partial derivative calculations have been introduced. Notably, they include such commonly used methods as kernel smoothing [23], derivation of polynomials, fitted over sets of points, and more uncommon ones like Kalman filtering [24].

Therefore, data clearance and noise-resistant derivative calculations have been combined to achieve decent smoothness in the framework. First of all, Gaussian smoothing kernels are applied for the data field on each time frame. This approach can reduce the significant outliers in the data and corresponds to the nature of the studied metocean processes, where the fields tend to be smooth. In the time-dependent multi-dimensional field, the smoothing is applied for each of the time frames. Two-dimensional Gaussian smoothing with selected bandwidth σ has the structure Eq. 5 and kernel Eq. 6, where s is the point, for which the smoothing is done, and s' - point, that value is utilized in smoothing.

$$\tilde{u}(s,t) = \int K_\sigma(s-s')u(s')ds'; \tag{5}$$

$$K_\sigma(s-s') = \frac{1}{2\pi\sigma^2}\exp\left(\frac{1}{2\sigma^2}\sum_{i=1}^{2}(s-s')_i\right); \tag{6}$$

In addition to smoothing, a noise-stable numerical differentiation scheme is applied. The derivative is taken by differentiation of polynomials constructed over the set of points in the selected window. The coefficients of the polynomials utilized in this step are obtained by linear regression. Despite all these measures, as presented on Tab. 1, derivatives of higher orders tend to have significant errors even after smoothing and polynomial derivation.

Table 1: Noise levels (%) for the raw noised data and for the smoothed data

	u	$\frac{\partial u}{\partial t}$	$\frac{\partial^2 u}{\partial t^2}$
Noised function	15.5	260.1	12973.8
Smoothed function	12.3	10.78	458.2

A particular example of the noised function field is shown in Fig. 1. For clarity purposes, only the spatial domain center slice is provided. However, it should be emphasized that the entire spatial field is smoothed out to obtain

the spatial derivative field, with the other spatial dimension processed by the kernel.

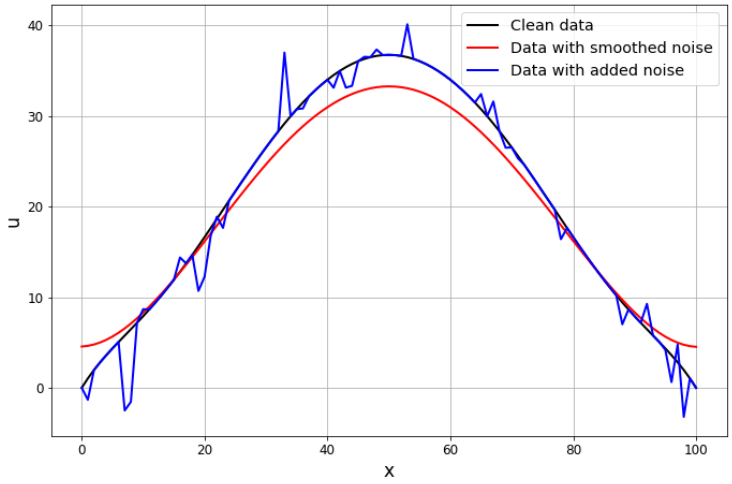


Figure 1: Graph of a section over one spatial dimension for synthetic input function (solution of wave equation with 2 spatial dimensions) in original state, with Gaussian noise, added to a fraction (40%) of points of the domain, and after the noise was smoothed by Gaussian kernel

Differentiation of the three different fields shown in Fig. 1 gives the derivative fields that have values of the different orders. Thus, they are shown in the different graphs in Fig. 2.

In most equations governing the real-world processes, derivatives' orders are limited to the first or second order. Derivatives of the slices shown in Fig. 1 are represented in Fig. 2 and indicate that the proposed algorithm of noise reduction in derivatives not only achieves values close to the values of the derivatives on noiseless data but also preserves the structure of the fields, which is vital for the main evolutionary algorithm due to the normalization of values on each time frame.

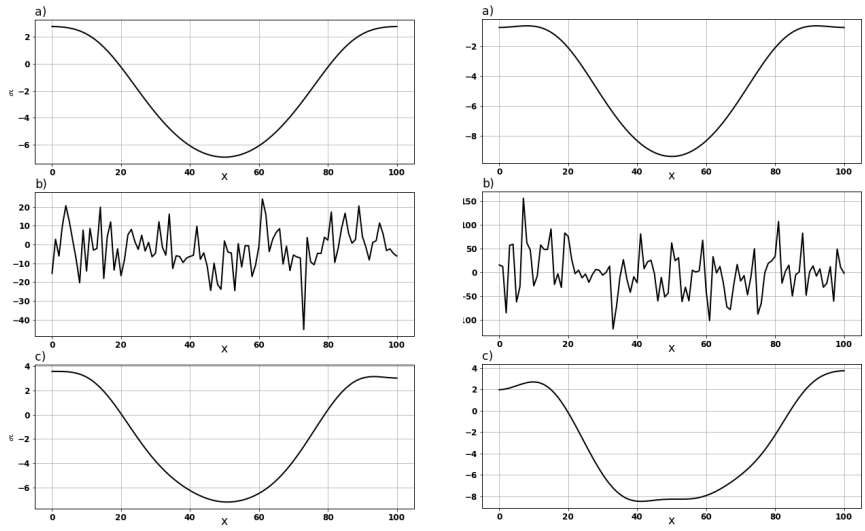


Figure 2: Graph of first (left column) and second (right column) order time derivative, calculated on input function (a)), noisy function (b)), and function with noise, smoothed by Gaussian kernel (c)

4.2. Evolutionary algorithm

After the preprocessing, which involves differentiation of initial field, the evolutionary algorithm is initiated. Here, we split the task of the equation discovery into two subtasks, performed in turns: the detection of the structure (terms) of the equation, and calculation of the real-valued coefficients, that correspond to these terms with the detection of valuable ones. The search of the optimal set of terms is performed with the evolutionary algorithm, while the calculation of intermediate coefficients is done with the regularized regression. During the search process, we use the values of factors ϕ, belonging to the task-specific types Φ, evaluated on the studied domain nodes, that form vectors as in Eq. 7, combinations of which form the terms of the searched equation.

a) Chromosome form and the fitness function. An example of genes in the chromosome is presented in Eq. 8. Here, the vector composed as the elementwise product (that is denoted with $\odot$ symbol) of vectors containing the original

function and its derivative along the x-axis is used as the regression feature set.

$$\mathbf{f}_1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}; \mathbf{f}_2 = \begin{bmatrix} u\left(t_0,\ x_0\right) \\ \vdots \\ u\left(t_i,\ x_j\right) \\ \vdots \\ u\left(t_m,\ x_n\right) \end{bmatrix}; \mathbf{f}_3 = \begin{bmatrix} u_x\left(t_0,\ x_0\right) \\ \vdots \\ u_x\left(t_i,\ x_j\right) \\ \vdots \\ u_x\left(t_m,\ x_n\right) \end{bmatrix}; ... \tag{7}$$

$$\mathbf{F}'_k = \begin{bmatrix} u(t_0, x_0) * u_x(t_0, x_0) \\ \vdots \\ u(t_i, x_j) * u_x(t_i, x_j) \\ \vdots \\ u(t_m, x_n) * u_x(t_m, x_n) \end{bmatrix} = \mathbf{f_2} \odot \mathbf{f_3}; \tag{8}$$

The normalization of terms values is held for each time frame passed into the algorithm for the correct operation of regularized regression during the further weight calculation phases. In this step, we can use arbitrary norm, but most commonly, L_2 norm or L_∞ norms are applied, as it is represented in Eq. 9.

$$\mathbf{F}_k = \begin{bmatrix} \frac{u(t_0,x_0)\cdot u_x(t_0,x_0)}{||f_2(t_0) \odot f_3(t_0)||} \\ \vdots \\ \frac{u(t_i,x_j)\cdot u_x(t_i,x_j)}{||f_2(t_i) \odot f_3(t_i)||} \\ \vdots \\ \frac{u(t_m,x_n)\cdot u_x(t_m,x_n)}{||f_2(t_m) \odot f_3(t_m)||} \end{bmatrix}; \tag{9}$$

The evolutionary part of the algorithm aims to select a set of terms to form the equation. In the set, one of the terms is randomly selected as the target. The target term is approximated with the weighted combination of the other terms in the list. In the beginning, a randomized collection of possible equations, which is called population, is declared. Every individual contains a set of terms with the selected target that can be interpreted as the right part of the equation and features, a linear combination of which composes the left part. In order to perform selection, the fitness function is introduced in Eq. 10 as the

inverse value of L_2-norm. The target term in the "right side" of the equation contains only one randomly selected term. Norm is taken as the differences between target $\mathbf{F}_{target}$ and the selected combination of features $\mathbf{F}$ with weighs α, obtained by the sparse regression (left side of the equation). Therefore, the evolutionary algorithm's task can be reduced to obtain the equation structure with the highest fitness function value.

$$f_{fitness} = \frac{1}{\|\mathbf{F} \cdot \alpha - \mathbf{F}_{target}\|_2} \longrightarrow max \tag{10}$$

The composition of the encoded terms represents the genotype of the individual. These encodings contain the parameters of each token in the term. The evolution of individuals is performed both by mutation and by a crossover in every iteration step. The mutation for an individual is introduced as the random change (addition, deletion, or alteration of factors) in its terms. For example, this can result in shift of equation term u_t to $u_{xx} * u_t$ or $u_x * u_{tt}$ to $u_x * u_t$. The elitism, introduced as the individual's exclusion with the highest fitness value, helps preserve the best-discovered candidate (the one with the highest fitness function value) during the mutation step.

b) Evolutionary operators. Crossover is the part of the evolutionary mechanism, which manifests as the gene exchange between two individuals to produce offspring with higher fitness values. In the task of data-driven equation derivation, it can be introduced as the exchange of terms between equations. In order to produce units with higher fitness values, the crossover should be held between selected individuals. Several tests have proved that the fastest convergence to the desired solution can be achieved with the tournament selection. In this policy, several tournaments, where the unit with the highest fitness value is selected for a further crossover, are held between individuals of the population. After that, parents for the offsprings are randomly chosen between the tournament winners. In contrast to the simple selection of several individuals with the highest fitness function values for reproduction, this approach can let the offsprings take good qualities from the population's less-valuable individuals.

The next essential element of the proposed data-driven algorithm is sparse regression. Its main application is the detection of the equation structure among the set of possible terms. With no original information about the equation structure and the correct number of terms, it is better to introduce the equation with a higher number of possible term candidates. Therefore some form of filtration has to take place. The main instrument in this phase is the Least Absolute Shrinkage and Selection Operator (LASSO). In contrast to other types of regression, LASSO can reduce the number of non-zero elements of the weights vector, giving zero coefficients to the features that are not significant to the target.

The minimized functional of the LASSO regression Eq. 11 takes the form of the sum of two terms. First is the squared error between vectors of the target, denoted as $\mathbf{F_{target}}$, and vector of predictions, obtained as the inner product of a matrix of features $\mathbf{F}$ and vector of weights α, while second in the L_1-norm of the weights vector, taken with sparsity constant λ:

$$\|\mathbf{F}\alpha - \mathbf{F}_{target}\|_2^2 + \lambda\|\alpha\|_1 \rightarrow \min_{\alpha} \tag{11}$$

The main drawback of the LASSO regression is its disability to acquire the correct values of the coefficients. Final linear regression over discovered effective terms is performed to obtain the resulting PDE's actual coefficients. In the final step, non-zero weights from the LASSO are rescaled with original unnormalized data as features and the target.

The pseudo-code for the resulting algorithm is provided in Appendix A

5. Numerical experiments

5.1. Synthetic data

a) Wave equation. The analysis of the algorithm performance is held on the synthetic data. This simplification can show the result's response to various types and magnitudes of noise, which is generally unknown on the measurement data. As in the previous studies, the solution of the wave equation with two

spatial variables Eq. 14, where t - time , x, y - spatial coordinates, u - studied function (for example, small out-of-plane membrane displacement), and $\alpha_1 = \alpha_2 = 1$ was taken as the synthetic data. The equation was solved, using the finite-difference technique for the domain, comprised of $201 \times 201 \times 201$ points in 2 spatial dimensions & time, and the proposed method was applied to the solution dataset. The grid, which covered the domain, had uniformly distributed nodes with coordinates between 0 and 10. The initial conditions for the equation were Eq. 12 & Eq. 13, and $u = 0$ was the boundary condition for the problem.

$$u = 10000 \sin\left(\frac{1}{100} xy(1 - \frac{1}{10}x)(1 - \frac{1}{10}y)\right)^2 \tag{12}$$

$$\frac{\partial u}{\partial t} = 1000 \sin\left(\frac{1}{100} xy(1 - \frac{1}{10}x)(1 - \frac{1}{10}y)\right)^2 \tag{13}$$

The algorithm has proved to detect the correct structure of the equation with the clean data, while on the noisy data, additional terms or completely wrong structures have been detected.

$$\frac{\partial^2 u}{\partial t^2} = \alpha_1 \frac{\partial^2 u}{\partial x^2} + \alpha_2 \frac{\partial^2 u}{\partial y^2} \tag{14}$$

Several noise addition experiments were held on the synthetic data: first of all, in a fraction of points (40% of total number) the noise of various magnitudes have been added: $(\mu = 0; \sigma = n * ||u(t)||,\ n = 0.1, 0.2, ..., 0.8)$. After that, the algorithm has been applied to this data. The results of the experiment are as follows: the method is successfully able to detect the structure of the equation for the interval of noise levels up to 14.9 %, which corresponds to the standard deviation of Gaussian noise in the interval [0, 0.35], multiplied by a norm of the field in the time frame. The weights errors in this interval are minor, as is shown in the Tab. 2. With higher noise levels (in the interval between 14.9% and 15.67%), the algorithm detects additional terms that are not present in the original equation, resulting in both distortion of equation structure and incorrect

weights calculation. Finally, with high noise levels, the proposed algorithm can lose grasp of the equation's correct structure.

Table 2: Discovered structures of the equations for the specific noise levels

Noise level of input data (%)	equation
0	$\frac{\partial^2 u}{\partial t^2} = 1.00\frac{\partial^2 u}{\partial x^2} + 1.00\frac{\partial^2 u}{\partial y^2}$
8.3	$\frac{\partial^2 u}{\partial t^2} = 1.02\frac{\partial^2 u}{\partial x^2} + 1.01\frac{\partial^2 u}{\partial y^2}$
10.9	$\frac{\partial^2 u}{\partial t^2} = 1.04\frac{\partial^2 u}{\partial x^2} + 0.99\frac{\partial^2 u}{\partial y^2}$
13.1	$\frac{\partial^2 u}{\partial t^2} = 0.96\frac{\partial^2 u}{\partial x^2} + 0.99\frac{\partial^2 u}{\partial y^2}$
14.9	$\frac{\partial^2 u}{\partial t^2} = 0.95\frac{\partial^2 u}{\partial x^2} + 1.2\frac{\partial^2 u}{\partial y^2}$
15.67	$\frac{\partial^2 u}{\partial t^2} = 0.84\frac{\partial^2 u}{\partial x^2} + 0.63\frac{\partial^2 u}{\partial y^2} + 0.12\frac{\partial u}{\partial y}$
16.45	$\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} = 0$
17.88	$\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} = 0$

In Fig. 3, the influence of the noise level added to the measured field on the derivative fields is shown.

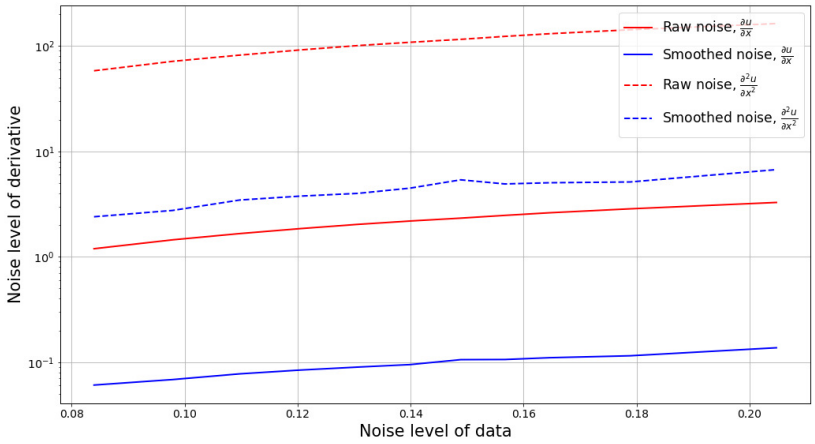


Figure 3: Noise levels of calculated first and second (dashed line) time derivatives, related to noise levels of input data

In the other experiment with the same data set, the noise of relatively high

magnitudes was added to a minor fraction of points (5% of the total number). In this case, the framework has shown similar results to the previous experiment: until the noise level of approximately 15%, the discovered structure was correct. On the data with higher noise magnitudes, errors in the structure of the equation occurred. This experiment has shown that in the studied cases, the main limiting factor for the algorithm's performance with implemented preprocessing for noise reduction is the noise level and not the distribution of noise across the studied field.

b) Korteweg-de Vries equation's solitary solution. To further analyze the synthetic data's algorithm performance, we have conducted additional experiments on the Korteweg-de Vries equation and the heat transfer equation.

To create a more specific situation for the Korteweg-de Vries equation Eq. 15, we have studied the solitary wave solution of the equation Eq. 16. This solution represents the transfer of a single wave, propagation with speed c from the initial position, specified by the wave crest's location at x_0. The data for the test is obtained from the solution function Eq. 16. The solution is evaluated on the uniform grid of 101 spatial points in the interval $x \in [0, 10]$ and 151 time points in the interval $t \in [0, 15]$.

$$\frac{\partial u}{\partial t} + 6u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0 \tag{15}$$

$$u = -\frac{c}{2} sech^2[\frac{\sqrt{c}}{2}(x - ct - x_0)] \tag{16}$$

The application of the framework to the solution Eq. 16, evaluated on a regular grid, failed to rediscover the initial equation. An improperly discovered model results from the simpler incidental forms in data, such as $u_t = -cu_x$. This equation's simplicity results in a higher probability of its discovery than for the full KdV equation. Additionally, the absence of the high-order derivatives in the structure, which are inevitably calculated with numerical error, may lead to higher fitness function values than in the correct equation. This experiment illustrates that the algorithm is susceptible to discovering "shortcut-equations",

which commonly represent the equality between functions (usually, different derivatives) present in the input functions pool. Similar cases have been studied to analyze the discovery process for ordinary differential equations in the previous works.

c) Heat equation with convection. To provide a more sophisticated test case for the framework, we have utilized the convection-diffusion equation. The equation belongs to the class of parabolic equations and has the structure Eq. 17, where ∇ represents gradient operator, and $\mathbf{v}$ is the velocity vector (field). We have studied example of the equation with transfer in only one direction (meaning,$\mathbf{v} = [v_1, 0]$; $v_1 = -1$, representing constant velocity field along x-axis), and $\alpha = 1$ - thermal diffusivity of the medium; $\nabla = \sum_i e_i \frac{\partial}{\partial x_i}$ - gradient operator; e_i - basis vector for i-th axis. The equation was solved on a grid with $100 \times 100 \times 100$ nodes in domain between 0 and 10 along each axis. The initial and boundary conditions are correspondingly presented in the Eq. 12 and Eq. 13.

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u) - \mathbf{v} \cdot (\nabla u) \tag{17}$$

$$u = 10 \sin\left(\frac{\pi}{100} x(10 - x)\right) \sin\left(\frac{\pi}{100} x(10 - x)\right) \tag{18}$$

$$u = 10 \sin\left(\frac{2\pi}{3} t\right) \sin\left(\frac{\pi}{100} y(10 - y)\right) + 0.05t \tag{19}$$

The evolutionary algorithm detects the correct structure of the equation in the majority of independent runs (out of 15 runs), as is shown in Tab. 3.

In Tab. 3 $c_i \approx 1$, $i = \overline{1,3}$, $c_4 \approx 1.59$, and $c_5 \approx 2.98$. This experiment indicates, that the algorithm is able to detect complex structures of the equation, if there are no "shortcut" solutions of the problem.

Table 3: Equations structures, detected in the experiment.

Equation	Number of experiments, getting the structure
$\frac{\partial u}{\partial t} = c_1 \frac{\partial^2 u}{\partial x^2} + c_2 \frac{\partial^2 u}{\partial y^2} + c_3 \frac{\partial u}{\partial x}$	9
$-11.06 \frac{d^3 u}{dx^3} = \frac{\partial u}{\partial x}$	1
$\frac{\partial u}{\partial t} = c_1 \frac{\partial^2 u}{\partial x^2} + c_2 \frac{\partial^2 u}{\partial y^2}$	3
$\frac{\partial^2 u}{\partial x^2} = -c_4 \frac{\partial u}{\partial y} + c_5 \frac{\partial u}{\partial x}$	2

5.2. Real data example

For the validation of the model, the dynamics of the two-dimensional field of sea surface height (SSH) data from the NEMO ocean model for the Arctic region (center of the Barents sea) for a modeling month a resolution of an hour has been used. The area is known to have strong tides, leading to the discovery of the time-dependent equation. It is necessary to emphasize that despite existing Tidal equations, there is no single analytical equation for the specific case of the SSH dynamics in this region due to the overlapping of different natures' processes. The studied domain was divided into daily intervals to reduce the risks of the deriving equation, which describes multiple processes following each other in the domain. The data's spatial properties are as follows: the intervals between nodes are approximately 5 km, while the domain contained 50×50 nodes.

After the application of the framework to the data, we obtain the equation in the form Eq. 20.

$$\frac{\partial u}{\partial x} = -0.0506 \frac{\partial u}{\partial t} - 0.0053 \frac{\partial^2 u}{\partial t^2} \tag{20}$$

Eq. 20 was solved, and the calculated field was compared with the initial one to validate the result of the algorithm. Since there is a second-order time derivative and first spatial derivative, the initial conditions (two initial time steps to represent the field and its first time derivative for the beginning of the studied period) and the boundary condition on one edge of the studied area are set. The

graphs of daily sea surface height dynamics from reanalysis and equation solution are presented in Fig. 4 and Fig. 5. The quality metrics show that the discovered equation can describe the equation well: $RMSE = 0.0434, MAE = 0.0446$ for the field with values in interval between approximately 0.5 and 0.9.

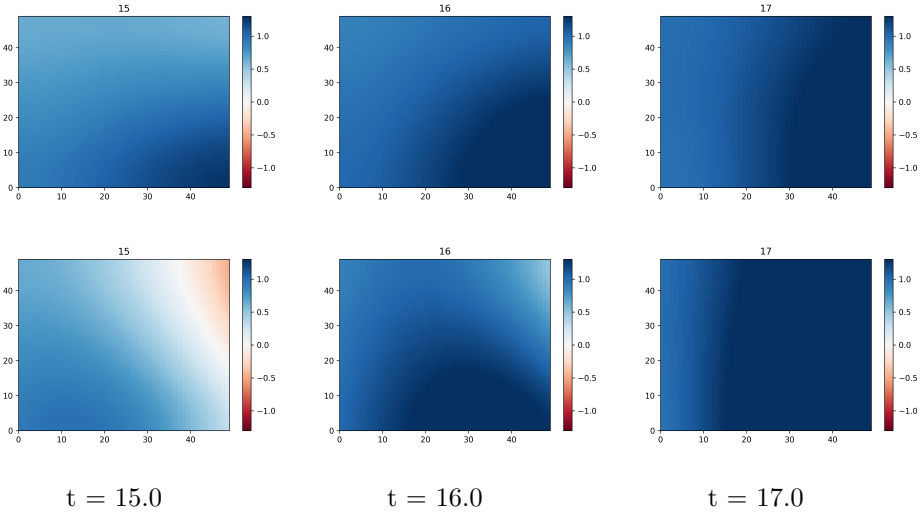


Figure 4: Example of SSH field, obtained from reanalysis (upper row) and the same field from Eq. 20, (lower row) for 3 time frames

5.3. Comparison with other methods

The experiments, similar to the ones in [7], have been performed to compare the proposed algorithm with existing state-of-the-art methods. Due to the framework's limitations, a single equation Eq. 22 is used, instead of the system Eq. 21, that is utilized in [7]. Additional difficulties to the comparison were contributed by unknown initial and boundary conditions in the referenced experiment.

$$\begin{cases} \frac{\partial U}{\partial t} = -U\nabla U + \nu\Delta U, U = (u, v)^T \\ U|_{t=0} = U_0(x, y) \end{cases} \tag{21}$$

$$\frac{\partial u}{\partial t} = (\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + u(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}) \tag{22}$$

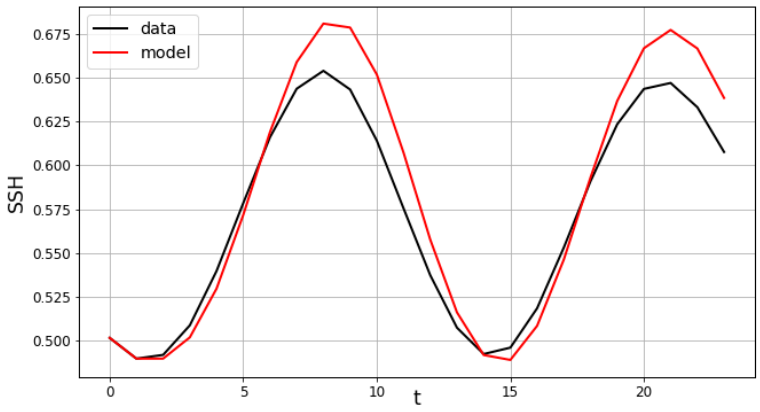


Figure 5: Dynamics of sea surface height for September 18, 2013: reanalysis (denoted as data) and solution of the equation, obtained from framework, denoted as model, for the center of the studied area

Eq. 22 was solved using finite differences, and the noise from normal distribution was added to simulate the previous experiment. The preprocessing phase, described in previous sections and involving smoothing and derivatives calculation, was performed to reduce noise's influence on the resulting equation.

The added noise was created from $k \times \max_{x,y,t} u(x,y,t) \times N(0,1)$. The experiments resulting in the correct structure have been conducted with the value $k = 0.001$, as in the compared study [7]. As the framework output, we will consider the closest to the equation's correct structure, obtained on the grid of sparsity constant values.

These tests show that noise resistance corresponds with other framework applications and is somewhat better than in the compared experiment. Despite the insignificant difference in the coefficient k (0.001 versus 0.00015), the noise level difference is significant (1% versus 6.3%). For the noise levels approximately below 5%, the correct equations were detected. The equation structure deteriorates, which manifests in wrong weights and additional terms or lack of mandatory ones.

Table 4: Discovered structures of the equations for the specific noise levels

k	Noise level of input data (%)	equation
0.0005	0.25%	$\frac{\partial u}{\partial t} = (0.999\frac{\partial^2 u}{\partial x^2} + 1.000\frac{\partial^2 u}{\partial y^2}) + $ $+u(1.001\frac{\partial u}{\partial x} + 1.000\frac{\partial u}{\partial y})$
0.00075	0.49%	$\frac{\partial u}{\partial t} = (1.001\frac{\partial^2 u}{\partial x^2} + 1.001\frac{\partial^2 u}{\partial y^2}) + $ $+u(0.999\frac{\partial u}{\partial x} + 0.999\frac{\partial u}{\partial y})$
0.001	0.97%	$\frac{\partial u}{\partial t} = (1.000\frac{\partial^2 u}{\partial x^2} + 0.999\frac{\partial^2 u}{\partial y^2}) + $ $+u(1.000\frac{\partial u}{\partial x} + 1.000\frac{\partial u}{\partial y})$
0.00125	4.2%	$\frac{\partial u}{\partial t} = (1.001\frac{\partial^2 u}{\partial x^2} + 1.002\frac{\partial^2 u}{\partial y^2}) + $ $+u(0.998\frac{\partial u}{\partial x} + 0.999\frac{\partial u}{\partial y})$
0.0015	6.3%	$\frac{\partial u}{\partial t} = (0.996\frac{\partial^2 u}{\partial x^2} + 0.998\frac{\partial^2 u}{\partial y^2}) + $ $+u(1.000\frac{\partial u}{\partial x} + 1.003\frac{\partial u}{\partial y}) - 0.0034\frac{\partial^2 u}{\partial y^2}\frac{\partial u}{\partial x}$

6. EPDE framework description

The framework, encompassing the described method, is designed to allow the user to customize the algorithm's significant elements while giving the default pipeline and necessary tools for the differential equation discovery. The setup of the equation discovery experiment requires the selection of functions (tokens) that form the pool, from which the algorithm creates the candidate equations. The main element that has to be defined is obtaining the function values on the set of processed points to evaluate further the fitness function in the Eq. 10. For example, the derivatives in the framework's current development are stored as the pre-computed matrices of their values on the grid. In contrast, trigonometric functions' values are calculated during the fitness function calculation due to a frequency parameter. The correct token evaluation method's selection can be viewed as the trade-off between memory storage utilization and computational powers involved in calculating functions during the algorithm run.

The token families, sets of elementary functions, are created with the def-

inition of the tool mentioned above. For every token, the range for function parameters (such as power or frequency) and markers are specified, setting the behavior of functions in the equation structure (i.e., if a function can be present in terms in the left side of the equation, if it is in the right part, or if multiple tokens from a token family can be in a term).

The workflow of the main evolutionary algorithm, which forms the algorithm's cornerstone, is mutable via the evolutionary operator's modifications. To guide the operator's development, we have introduced the builder class, representing the eponymous pattern. The operators, presented in the Sec. 4, are included in its default form, provided by the director class. However, the user can modify all of the significant elements if the specific equation discovery task requires it. The selection of the parameters for the evolutionary operators is made in their definition in the builder class.

7. Neural networks approximation with automatic differentiation

This section is dedicated to changing the differentiation method. The proposed algorithm has a modular structure. Thus, we may replace the differentiation algorithm from finite differences or analytical differentiation of polynomials to the neural network approximation with further automatic differentiation.

7.1. Application of artificial neural networks to the data preprocessing

Automatic differentiation is a standard tool in deep learning frameworks. The process of neural network training utilizes the backpropagation technique based on calculating the loss function gradient with respect to the weights' values, which is often done via automatic differentiation. With this approach to the derivative calculation problem, the preprocessing stage involves two stages: fitting the artificial neural network to the input data and the automatic differentiation with respect to the spatial coordinates and time.

The multi-layered feed-forward artificial neural network's training process with the sigmoid activation functions was implemented, using the `tensorflow`

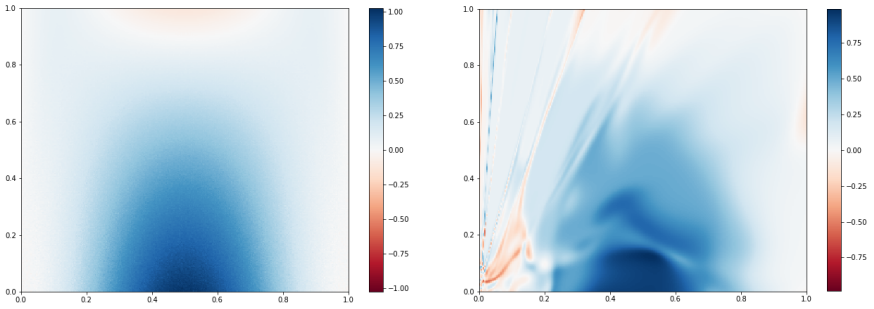
framework. We use the network that contained three fully-connected layers (generally, with 256, 512, and 64 neurons) in the experiments. The selection of architecture was driven by the propositions in [25, 26]. We utilize the mean squared error as the loss function during the artificial neural network training process, performed with Adam stochastic optimization algorithm. A random sample of studied data points (the function we want to describe with the differential equation, which we will obtain later) is used for each epoch's training process.

After the ANN training process with the studied data, automatic differentiation is used to obtain the derivatives. In contrast to the previously mentioned method of calculating derivatives of polynomials fitted only along an axis, the automatic differentiation technique can get mixed partial derivatives. The gradients, hessian and further derivatives are collected from the in-built methods of automatic differentiation.

7.2. The analysis of ANN preprocessing properties

The wave equation's test case with one spatial dimension was examined to compare the noise-reducing properties of proposed derivatives evaluation methods: the kernel smoothing and analytical differentiation of fitted polynomials against the automatic differentiation of artificial neural networks. The noise is added in the way shown previously. However, data is not separated into subdomains. Here, we add the Gaussian noise with $\mu = 0$ and $\sigma = 0.03 * max(u(t))$ to each of the time frames.

The lower quality of the artificial neural field reconstruction of the noisy field presented in Fig. 6, can not be attributed to the overfitting or the underfitting of the network: the same structures are obtained in the initial stages of loss function stabilization and on the latter epochs. Therefore, we can attribute the method's downside to the particular architecture: 3 fully connected layers of the ANN with particular neurons. Nevertheless, the comparison of the fields obtained after the differentiation (see Fig. 7) shows that the default kernel smoothing and the polynomial fitting algorithm can better calculate the fields.



(a) Original noised field

(b) The noised field reconstruction

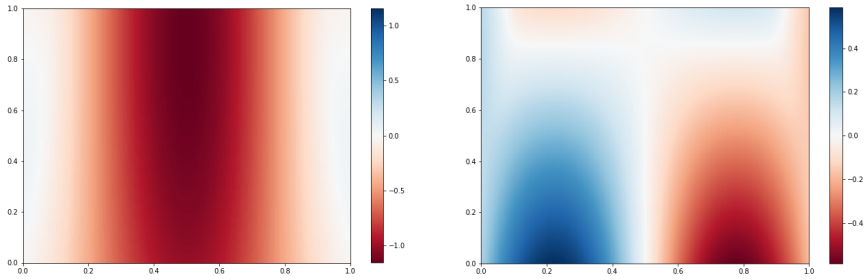
Figure 6: The comparison between the solution of wave equation $x-$ and $y-$ axes are coordinates, color value of the function $u(x, y)$ a) and the ANN, fit to the solution b)

In the discussed experiments, the noise levels, introduced in Eq. 4, in the initial fields are approximately 2.49%. The noise levels in the first time derivatives (Fig. 8) are 31.6% and 1116.89% for default method and ANN accordingly, and the first spatial derivatives (Fig. 9) are 29.7% and 927.14% correspondingly.

8. Conclusion

The proposed method has proven to be suitable for the data-driven derivation of equations that can model various physical processes. The robustness of the algorithm to the noise in the input data provided by improved preprocessing of data allows the framework applicable to real-world problems. Even in the cases of substantial noise in the input data, the resulting equations had the correct structures and, therefore, can correctly describe the studied system. Other notable points about the algorithm operation can be stated:

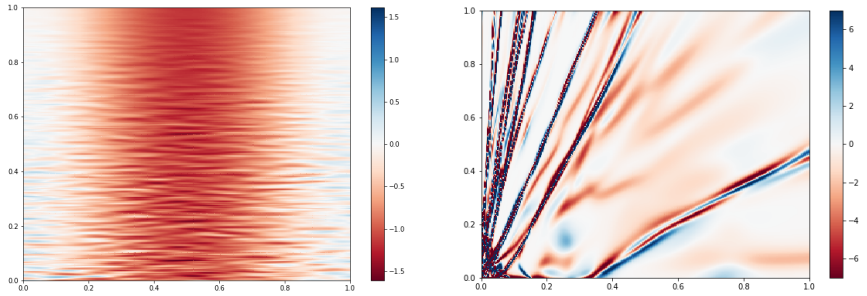
- To achieve a good quality of the resulting processes, the areas, localizing different processes, should be separated and studied independently. It is presented in the case of real-world data processing when the area that is already known to have strong time dependencies of sea surface height was separated, and the equation for it was derived;



(a) Field of the correct first time derivative

(b) Field of the correct spatial time derivative

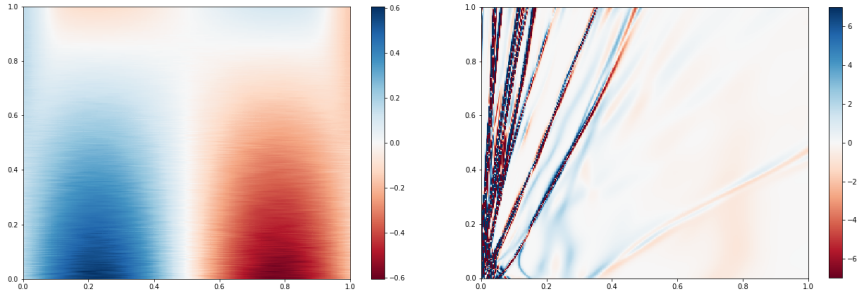
Figure 7: The fields of time and spatial derivatives, calculated from the noiseless data



(a) Kernel filtering & polynomial differentiation

(b) Automatic differentiation result

Figure 8: The comparison between the differentiation methods for first time derivative solution of wave equation



(a) Kernel filtering & polynomial differentiation (b) Automatic differentiation result

Figure 9: The comparison between the differentiation methods for first spatial derivative solution of wave equation

- The meta-parameters of the algorithm have a strong influence on the final result. For example, low values of sparsity constant can lead to additional terms in the equation, while its higher than optimal values can completely distort the equation structure. Therefore, mechanisms of meta-parameter selection should be implemented in the further development of the method;

- The proposed preprocessing technique, that combines kernel smoothing and fitting the Chebyshev polynomial to the data in a specified window, is proven to be an efficient derivative calculation tool. It was shown to perform better than the automatic differentiation of an artificial neural network.

Areas of the further development of the framework can include deriving a more generalized class of equations, using similar techniques, not limiting the results in a class of partial differential equations. Additionally, the equations for vector variables or even systems of equations can be the next targets for the work.

Source code is publicity available at GitHub [1].

Acknowledgements

This research is financially supported by The Russian Scientific Foundation, Agreement #19-71-00150.

References

[1] NSS Team, Fedot E* algotirhms, https://github.com/ITMO-NSS-team/FEDOT.Algs (2020).

[2] M. Maslyaev, A. Hvatov, A. Kalyuzhnaya, Data-driven partial differential equations discovery approach for the noised multi-dimensional data, in: International Conference on Computational Science, Springer, 2020, pp. 86–100.

[3] J. Berg, K. Nyström, Neural network augmented inverse problems for pdes, arXiv preprint arXiv:1712.09685.
URL https://arxiv.org/abs/1712.09685

[4] H. Schaeffer, R. Caflisch, C. D. Hauck, S. Osher, Learning partial differential equations via data discovery and sparse optimization, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciencedoi:473(2197):20160446.

[5] S. H. Kang, W. Liao, Y. Liu, Ident: Identifying differential equations with numerical time evolution, arXiv preprint arXiv:1904.03538.

[6] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, Journal of Computational Physics 395 (2019) 620–635.

[7] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: Learning PDEs from data, in: International Conference on Machine Learning, 2018, pp. 3208–3216.

[8] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, The Journal of Machine Learning Research 19 (1) (2018) 932–955.

[9] M. J. Asher, B. F. W. Croke, A. J. Jakeman, L. J. M. Peeters, A review of surrogate models and their application to groundwater modeling, Water Resour. Res. 51 (2015) 5957–5973. doi:10.1002/2015WR016967.

[10] M. P. Rumpfkeil, P. Beran, Multi-fidelity surrogate models for flutter database generation, Computers & Fluids 197.

[11] N. O. Nikitin, P. Vychuzhanin, A. Hvatov, I. Deeva, A. V. Kalyuzhnaya, S. V. Kovalchuk, Deadline-driven approach for multi-fidelity surrogate-assisted environmental model calibration: Swan wind wave model case study, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 1583–1591.

[12] M. Maslyaev, A. Hvatov, A. Kalyuzhnaya, Data-driven partial derivative equations discovery with evolutionary approach, in: International Conference on Computational Science, Springer, 2019, pp. 635–641.

[13] B. D. Santer, T. M. Wigley, M. E. Schlesinger, J. F. Mitchell, Developing climate scenarios from equilibrium gcm results.

[14] M. Cabré, S. Solman, M. Nuñez, Creating regional climate change scenarios over southern south america for the 2020's and 2050's using the pattern scaling technique: validity and limitations., Climatic Change 98 (2010) 449–469. doi:10.1007/s10584-009-9737-5.

[15] S. Castruccio, D. J. McInerney, M. L. Stein, F. Liu Crouch, R. L. Jacob, E. J. Moyer, Statistical Emulation of Climate Model Projections Based on Precomputed GCM Runs*, Journal of Climate 27 (5) (2014) 1829–1844. doi:10.1175/JCLI-D-13-00099.1.

[16] T. Weber, A. Corotan, B. Hutchinson, B. Kravitz, R. Link, Technical note: Deep learning for creating surrogate models of precipitation in earth system models, Atmospheric Chemistry and Physics 20 (2020) 2303–2317. doi:10.5194/acp-20-2303-2020.

[17] K. Kaheman, S. L. Brunton, J. N. Kutz, Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data, arXiv preprint arXiv:2009.08810.
URL https://arxiv.org/abs/2009.08810

[18] L. Zhang, H. Schaeffer, On the convergence of the sindy algorithm, Multiscale Model. Simul., 17(3) (2019) 948–972.
URL https:///arxiv.org/abs/1805.06445

[19] S. H. Rudy, A. Alla, S. L. Brunton, J. N. Kutz, Data-driven identification of parametric partial differential equations, SIAM Journal on Applied Dynamical Systems 18 (2) (2019) 643–660.

[20] M. Raissi, P. Perdikaris, G. E. Karniadakis, Numerical gaussian processes for time-dependent and nonlinear partial differential equations, SIAM Journal on Scientific Computing 40 (1) (2018) A172–A198.

[21] J. Berg, K. Nyström, Data-driven discovery of pdes in complex datasets, Journal of Computational Physics 384 (2019) 239–252.

[22] M. Raissi, P. Perdikaris, G. Karniadakis, Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. arxiv 2017, arXiv preprint arXiv:1711.10566.
URL https://arxiv.org/abs/1711.10566

[23] I. Knowles, T. Le, A. Yan, On the recovery of multiple flow parameters from transient head data, Journal of Computational and Applied Mathematics 169 (1) (2004) 1–15.

[24] R. Piche, Automatic numerical differentiation by maximum likelihood estimation of state-space model, arXiv preprint arXiv:1610.04397.
URL https://arxiv.org/abs/1610.04397v1

[25] Z. Zainuddin, P. Ong, Function approximation using artificial neural networks, International Journal Of Systems Applications, Engineering & Development 1 (4) (2007) 173–178.

[26] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

Appendix A. Pseudo-code of the algorithm

Input: set of elementary tokens T, symbolically representing constant, initial function, and its various derivatives; set of function measurements from the studied field

Parameters: M - number of token combinations in a single individual; k - number of elementary tokens in a combination; n_pop - number of candidate solutions in the population; evolutionary algorithm parameters: number of epochs n_{epochs}, mutation $r_{mutation}$ & crossover rates $r_{crossover}$, part of the population, allowed for procreation a_{proc}, number of individuals, refrained from mutation (elitism) a_{elite}; sparse regression parameter - sparsity constant λ

Result: The structure of the partial differential equation with the corresponding weights, best fitting the input field

Smooth the measurements & calculate the derivatives; Generate population **P** of individuals, representing equation, of size n_pop, with M - random permutations of k tokens to form sets C^j;

for *epoch = 1 to* n_{epochs} **do**

  for *individual in population* **do**

    Apply sparse regression to the individual to calculate weights;

    Calculate fitness function to individual;

  end

  Hold tournament selection and crossover;

  for *individual in population except* $n\_pop \times a_{elite}$ *"elite" ones* **do**

    Mutate individual;

  end

end

Select the individual with the highest fitness function value as the final structure of the solution to the problem;

Calculate the correct weights of the equation using linear regression.

Algorithm 1: The pseudo-code of the algorithm operation