Solver-Based Fitness Function for the Data-Driven Evolutionary Discovery of Partial Differential Equations

Mikhail Maslyaev
Nature Systems Simulation Laboratory
ITMO University
Saint-Petersburg, 197101, Russia
mikemaslyaev@itmo.ru

Alexander Hvatov
Nature Systems Simulation Laboratory
ITMO University
Saint-Petersburg, 197101, Russia
alex_hvatov@itmo.ru

***Abstract*—Partial differential equations provide accurate models for many physical processes, although their derivation can be challenging, requiring a fundamental understanding of the modeled system. This challenge can be circumvented with the data-driven algorithms that obtain the governing equation only using observational data. One of the tools commonly used in search of the differential equation is the evolutionary optimization algorithm. In this paper, we seek to improve the existing evolutionary approach to data-driven partial differential equation discovery by introducing a more reliable method of evaluating the quality of proposed structures, based on the inclusion of the automated algorithm of partial differential equations solving. In terms of evolutionary algorithms, we want to check whether the more computationally challenging fitness function represented by the equation solver gives the sufficient resulting solution quality increase with respect to the more simple one. The approach includes a computationally expensive equation solver compared with the baseline method, which utilized equation discrepancy to define the fitness function for a candidate structure in terms of algorithm convergence and required computational resources on the synthetic data obtained from the solution of the Korteweg-de Vries equation.**

***Index Terms*—equation discovery, partial differential equation, fitness function selection, data-driven modelling**

I. Introduction

Differential equations are commonly used as mathematical models for continuous physical processes. They describe the interdependence of various derivatives of a studied multivariate function. In addition, differential equations can be used for system state predictions. By integrating the governing partial differential equation with specific initial and correctly stated boundary conditions, the state of the modeled process in the future can be obtained.

The task of partial differential equation derivation for a specific problem in the past involved examining conservation laws that can be applied to the system and using analysis and variational principles to extract the equation from the known properties of the system. While these methods are widely used to derive the equations, they demand significant preliminary study of the process, which in some cases can not be held due to low comprehension of the system. Occasionally, regardless of the dynamical system understanding, the researchers must collect measurements as a preliminary part of the analysis. Thus, the data-driven equation derivation can be introduced as an alternative. Data-driven algorithms can develop a model for a process using measurements data and a few assumptions about the constructed model structure (i.e., the model will take the form of a partial differential equation).

One of the contemporary approaches to data-driven PDE discovery is based on evolutionary algorithms (EA). Here, the objective of EA is the construction of a model in the form of a partial differential equation from a selected set of elementary constituents and with specific conditions that have the best performance on the input data. One of the issues linked with applying evolutionary operators to such ambiguously stated tasks is selecting an objective (fitness) function that shall be optimized during the problem solution.

The computation complexity of the fitness function is a classical problem that is considered for classical optimization benchmark problems [1]. However, in the equation discovery case, the function landscape cannot be known a priori. Thus, only empirical conclusions on the influence of the hardness of computational complexity of a fitness function may be obtained. In this paper, two possible approaches to compute fitness functions for equation discovery evolutionary algorithm are considered. First is the computationally hard complete equation solution. Equation solution should be a more noise-resistant approach to evaluating the quality of evolutionary algorithm candidates due to the independence of a noisy data differentiation error. The ability to handle noisy data is crucial for modeling real-world physical systems since instrumental observational data gathering is usually imposed by various disturbances. Moreover, the solver approach allows considering boundary values, which are not in the scope of previously used computationally more straightforward fitness calculation technique. It involved evaluating differential operator discrepancy from zero and performing poorly on noisy data.

The paper is structured as follows: Section II is dedicated to the analysis of existing approaches of data-driven discovery of

partial differential equations for purposes of modeling dynamical systems; Section III gives a brief formal overview of the solved problem and states the tasks for the research; Section IV provides a description for the evolutionary algorithm of partial differential equation discovery and compares the possible ways of the case-specific fitness function definition; Section V is dedicated to the analysis of algorithm performance on synthetic data, and Section VI outlines the paper and also describes the possible future work directions.

II. Related work

The earliest advances in the data-driven algorithmic derivation of partial differential equations were made with symbolic regression [2]. The main idea of symbolic regression is constructing an expression that describes dependencies in data using a computational graph. The leaves of the computational graph are used for modeled function and its derivatives, while internal nodes are reserved for algebraic operations. Then, the graph is optimized to represent the input data. While symbolic regression can construct an arbitrary form of expression for the final model, it has some disadvantages, such as its tendency to overtrain and vast search space, leading to poor performance of the optimization algorithm. Symbolic regression - based approach has proved to be rather successful in problems of ordinary differential equation discovery. The examples of such algorithm applications are numerous and include [3] and [4]. The former article involves application of numerical tools of differential equations solution to evaluate the quality of a candidate, while the latter is notable for use of multiobjective optimization, considering not only the solution fitting to the problem, but also its complexity.

The next class of data-driven methods of PDE discovery is based on sparse regression, trained over a defined beforehand library of candidate terms. Examples of works involving sparse regression are numerous and include [5]–[8]. The main advantage of this approach is its low computational demands, but it can achieve such good performance with the cost of a limited expanse of possible equations, discoverable by the framework: the researcher has to manually select all reasonable terms for the equation.

One of the most actively developing methods involves applications of artificial neural networks (ANN) to derive the partial differential equation in the form of first-order time derivative approximations. The most notable examples in this include physics-inspired artificial neural networks, developed by [9], aimed at the representation of both input data and the structure of the PDE with ANNs. Another approach, proposed by [10], views differential operators through the lens of convolution kernels, thus approximating the system's dynamics. The former approach has the downside of requiring knowledge about the general structure of the equation. The latter approach can detect an arbitrary operator for the dynamics, but the classes of learned equations are still limited by necessarily having first-time derivatives.

Evolutionary algorithms of PDE discovery proved to be rather effective, being able to derive equations with complex structures and systems of equations. Conceptually similar approaches were introduced in [11] and [12]. However, they tend to have stability and convergence issues while operating on noisy data. The inaccurate estimations of derivatives, leading to the errors in calculating fitness functions and incorrect operation of selection operator, make their output inconsistent and the constructed models unreliable.

During the equation discovery process, different forms of operators appear. In a classical mathematical physics analysis, many methods allow solving an equation of a given type with a given type boundary conditions. Classical solution approaches require the expert to choose the proper method for the given equation. Such an approach is improper for data-driven discovery. We require a more automated yet maybe less precise equation solver. Such approaches usually involve neural networks [13]. In the paper, we use an alternative realization of a neural network-based solver.

III. Problem statement

This work's primary goal is to check the influence of the more advanced fitness function on the data-driven derivation of ordinary and partial differential equations. As in the majority of the data-driven methods, the model is constructed based on sets of measurements. It has to be expected that a single dynamical process prevails in the entire area so that the derived model can be applied to the complete dataset. The versatility of the approach makes possible the creation of models in forms of non-linear ordinary and partial differential equations with the defined order n in the form of the expression Eq. 1, where t is time, $\mathbf{x}$ is spatial coordinates vector, u is the function to be modeled. The subscript denotes derivative along that axis, e.g., u_t means the first-order time derivative.

$$F(t, x_1, \ \dots, x_k, u_t, u_{x_1}, \ \dots, u_{tt}, u_{tx_1}, \ \dots) \sim Lu = 0 \quad (1)$$

In contrast to the symbolic regression, which does not impose any limitations on the models' structures, the proposed algorithm makes assumptions about the equation form to reduce the search space and avoid unlikely candidates. The candidate model structure is granted a form of Eq. 2, where the factors f_j belong to the set of derivatives $\{u_t, u_{x_1}, \ \dots, u_{tt}, \ \dots\}$. The highest order of the derivative along the axis for the set may vary. In some cases, the dynamical system's details may indicate that the derivatives along a specific axis can be only up to a certain order, or a similar notion can be obtained from an algorithm's launch results. Also, it is possible to assume that the total number of significant (i.e., with nonzero coefficient) terms in the candidate equation and factors in a term will be relatively low due to the majority of existing equations for dynamical systems having this property.

$$\begin{gathered} L'u = \textstyle\sum_i a_i(t, \mathbf{x}) c_i = 0 \ , \ c_i = \prod_j f_j, \\ a_i(t, \mathbf{x}) = a_i'(t, \mathbf{x}) * b_i, \ b \in \mathbb{R} \end{gathered} \quad (2)$$

While the process modeling shall be based on the solution of a single equation, the study [14] shows the benefits of

the multi-objective optimization approach. It allows the researcher to obtain the set of Pareto-optimal solutions in terms of multiple objective criteria, such as quality of the model (zero discrepancies or data reproduction precision computed with solver), its complexity, etc. For the latter criterion, the approach adopts the proposed quantity of the significant terms in the equation, containing derivatives or other variables, that are informative for the research, as shown in Eq. 3. The selection of the former objective function is the main object of this research.

$$C(L'_j u) = \#(L'_j u) \tag{3}$$

The multi-objective optimization operates by a separate EA based on the MOEA/DD algorithm, proposed by [15]. In study [14], the optimization problem was stated for the tasks of differential equations systems, but its approach is applicable for the case of a single equation discovery.

IV. Algorithm Description

This section describes the general details of the proposed partial differential equations discovery approach. The generalized scheme of the developed evolutionary algorithm is presented in the Fig. 1. For the purposes of the fitness evaluation both optimization-based partial differential equation solver and equation discrepancy evaluation procedure can be used.

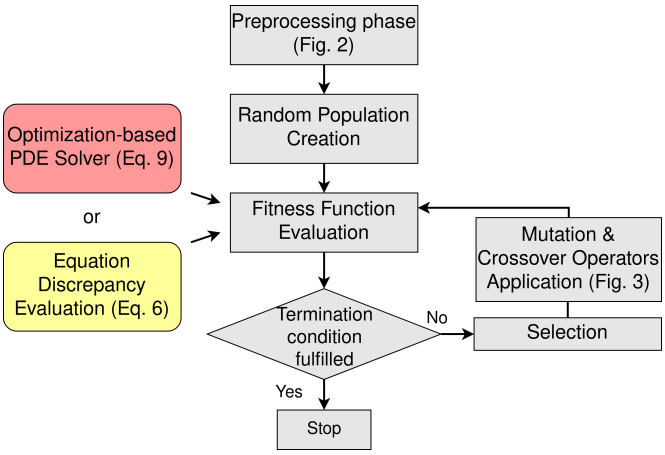

Fig. 1. Scheme of the evolutionary algorithm of partial differential equation derivation.

A. Evolutionary algorithm

The proposed algorithm's main objective is to discover a Pareto-optimal set of equations regarding objective functions, describing various aspects of equation quality. The algorithm uses hyperparameters vectors of the single equation discovery algorithm for multi-objective optimization. The previous works have shown that the evolutionary algorithm, performing the search of an equation, can converge to a single solution or several equivalent equations if provided with enough iterations and correctly selected evolutionary operators. These discovered equations are defined only by data and by the hyperparameters of the algorithm, mainly sparsity parameters, which limit the number of terms in the equation.

Before the fitness evaluation discussion, we provide the general details of the evolutionary algorithm shall be introduced without delving into the details of candidate equation quality evaluations. The search process for the best equation form is divided into several steps. At first, the evolutionary operators are utilized to suggest a set of terms $\{a'_1 c_1,\ ...a'_k c_k\}$, where c_i is the product of selected derivatives or modeled function, while the products a'_i are constructed from an allowed set of case-specific functions. Next, in a loop over the terms of the equation in a set, the following procedure is performed: the term is selected as a "right part of the equation", while other ones are used to approximating it: the sparsity is applied, and then the coefficients are refined. The overview of the EA is presented in the form of Alg. 1.

In each algorithm launch, a total number of terms in the set $\{a'_1 c_1,\ ...a'_k c_k\}$ is fixed for mutation and crossover evolutionary operators. Sparsity operator 4 has to be applied to set zero coefficients to the additional terms in the set, absent in the equation, describing the process. Here, the sparsity-promoting technique, based on LASSO regression, is employed, although other works indicate that the correlation can be used as an indicator of term significance. In the operator description, $\beta*$ and β refer to the sparse vector of real-valued intermediate equation weights. The intermediate status is guided by the nature of sparse regression, where the features and predicted values have. λ is the sparsity constant, which is case of the equation discovery regulates the number of terms with non-zero coefficients. $\mathbf{F}$ is the matrix of left part terms, evaluated on domain grid, while F_{target} is the vector of right part term.

$$\beta* = \arg\min_{\beta} \|\mathbf{F}_k \beta - F_{target,k}\|_2^2 + \lambda \|\beta\|_1 \tag{4}$$

The operation of sparse regression requires calculated libraries of the predictors. Thus the algorithm must have access to the values of all possible equation factors in the grid, placed in the studied domain. While the values of functions $a'_i(t, \mathbf{x})$ can be calculated from the coordinates of the grid points, the evaluation of derivatives has to be performed in the other way to increase the overall speed of the algorithm.

Differentiation prepossessing method phase includes the stage of the input data variable approximated by the fully-connected artificial neural network. Then the outputs of ANN for specific points are used in the finite-difference scheme to evaluate the derivatives. The main advantage of using an intermediate artificial neural network between data assimilation and derivative calculation is its ability to filter out the input noise. Finite-difference differentiation has the property of increasing the magnitudes of noise. Thus to obtain reasonable values of the derivatives, the data has to be smoothed beforehand. Other preprocessing approaches include kernel (Gaussian) filtering and fitting polynomials for the analytical differentiation, as presented in Fig. 2.

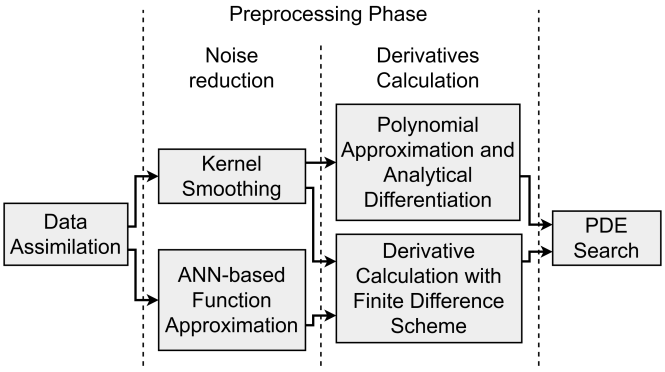

Fig. 2. Scheme of the possible preprocessing phases of the evolutionary algorithm of PDE discovery. The alternative ways indicate alternative steps of the algorithm. The specific choice is dependent on prevalence of noise in data, time restraints, etc.

The final element of the proposed algorithm of equation discovery is the calculation of real-valued coefficients between the terms, selected by the sparsity operator: $\{a_i' c_i \mid \beta *_i \neq 0\}$. For these purposes, linear regression is employed, so the coefficients β are obtained.

During the initialization of the evolutionary algorithm, a random initial population of equations is constructed with derivatives of the modeled function and a set of additional case-specific functions selected by the researcher. While operating with the EA, the "phenotype" of an individual stands for the equation it represents, while the "genotype" under the current encoding paradigm is defined with the string of objects representing equation terms. These terms by themselves consist of a string of factors. Some additional restrictions are imposed on the constructed terms to avoid some undesired behavior of the algorithm. For example, all terms must have the modeled function, or another function, specified as meaningful for the model. In other cases, function type-specific restrictions are necessary, e.g., the higher powers of trigonometric functions must be avoided to prevent trigonometric identity "discovery" that obviously will have lower zero discrepancies than the other candidates and will be preferred by the algorithm.

During the search for the optimal structure of PDE, the evolutionary operators of mutation and crossover are applied to the population, as shown on the scheme of the algorithm. The elitism is introduced to preserve the quality of the best candidate solution. The scheme of evolutionary operators is presented in Fig. 3.

B. Equation discrepancy evaluation

The specifics of the task pose an unusual problem: the question of selecting an objective function, which allows faster and more reliable convergence of the evolutionary algorithm. Previously, the only viable approach was the evaluation of the quality of equation term approximation, i.e., solving the optimization problem, stated on the Eq. 5.

Data: set of tokens $T = \{T_0, T_1, \ ... \ T_n\}$, where T_0 are derivatives and $T_1, \ ... \ T_n$ are user-specified functions, sparsity constant $lambda$
Result: partial differential equations
Randomly generate initial population of candidate equations from tokens from set T;
for *individual in population* **do**
  $right\_part\_idx = 0$;
  $max\_fitness\_val = -inf$;
  for *target_idx = 0 to terms_number* **do**
    Apply sparse (LASSO) regression to find intermediate coefficients β;
    Apply linear regression to find correct coefficients of the equations & **get fitness value** $\mathbf{fit\_val}$;
    if $fit\_val > max\_fitness\_val$ **then**
      $max\_fitness\_val = fit\_val$;
      $right\_part\_idx = target\_idx$;
    else
      pass
for *epoch = 1 to epoch_number* **do**
  population.sort();
  Remove worst solutions to maintain population size;
  Tournament selection of parents for recombination;
  Apply recombination and mutation operators;
  for *new_individual in offsprings* **do**
    Use LASSO operator and linear regression to find the best partition into left & right parts and **calculate fitness** (as above);
  Add new solutions into population;

Algorithm 1: The pseudo-code of a single equation discovery process

$$L = \sum_i a_i^*(t, \mathbf{x}) b_i c_i \longrightarrow \min_{a_i^*; c_i} \tag{5}$$

The objective function, representing the quality of the proposed model, is evaluated as the L2-norm of discrepancy of the right part of candidate equation term approximation with the left part ones on the domain grid points. The overview of the function is portrayed in Eq. 6

$$\begin{aligned} f_{fitness} &= (||L||_2)^{-1} \\ &= (|| \sum_{i \neq i\_rhs} (a_i^*(t, \mathbf{x}) b_i c_i) - a_{i\_rhs}^*(t, \mathbf{x}) c_{i\_rhs} ||_2)^{-1} \end{aligned} \tag{6}$$

While the equation discovery process, guided by this objective function, on noiseless data can converge to the desired candidate solution, several issues may arise in other cases. First of all, the correct structure of the equation may not be optimal from the sense of the introduced optimization metrics. An example of such an occasion can be found during the process of heat equation discovery Eq. 7.

$$u_t = \nabla(\alpha \nabla u) \tag{7}$$

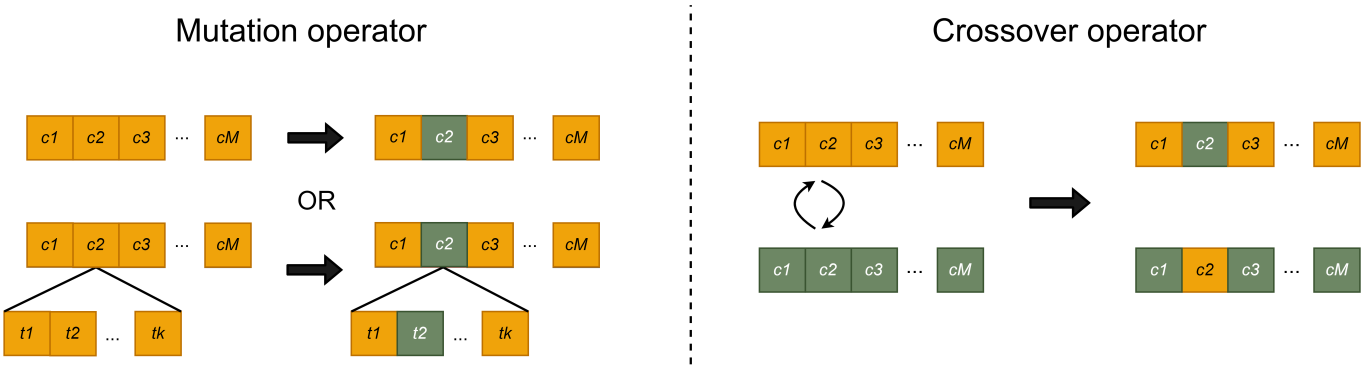

Fig. 3. Scheme of the implemented evolutionary algorithm operators for tasks of partial differential equation discovery. As presented on the left, mutation operator works by altering the existing solutions, while crossover operator, portrayed on the right, combines terms from selected equations.

Due to the properties of the equation, the second time derivative of its solution is close to zero across the domain. On the other hand, the second derivative in the extended heat equation has a physical meaning in some physics applications. The multi-objective optimization algorithm tends to converge to the candidate $u_{tt} = 0$, which has low modeling error on the input data, and complexity of 1 term, thus alone forming Pareto non-dominated set. However, the predictive qualities of this model are low, and integration of the equation will not achieve the result. Such cases should be handled manually.

C. Solver-based approach

Another approach to evaluate the fitness function of a partial differential equation is defined by using an automated technique of PDE solution. In contrast to the conventional algorithms of numerical differential equations solution (finite-difference or spectral methods), we should process an arbitrary candidate proposed by the evolutionary algorithm.

The quality of the evolutionary algorithm individual is defined with the fitness function, stated in Eq. 8, where u represents the data, and $\widetilde{u}$ denotes the ANN approximation of the equation solution.

$$f_{fitness} = (||\widetilde{u} - u||_2)^{-1} \tag{8}$$

In line with the objective for the fitness function evaluation, the equation solution process is performed on the fixed mesh $(t_i, \mathbf{x}_i)$ in the domain Ω, corresponding with the data points. In most problems, the mesh is uniform, but an arbitrary discretization can also be selected.

The PDE-solving algorithm requires correctly (at least "domain-averaged" correctly) posed differential operator and initial/boundary conditions, matching with the type of passed boundary problem, expressed in Eq. 9, where for the modelled function $u(t, \mathbf{x})$, defined in domain $(t, \mathbf{x}) \in \Omega \subset \mathbb{R}^{k+1}$, where k is the number of spatial dimensions. In the studied examples, the cases of two dimensions (time and space) are analyzed, but there are no strict limitations on the dimensionality. L and b are correspondingly arbitrary (possibly, non-linear) differential and boundary operators, with the latter defined on the boundary Γ.

$$\begin{cases} Lu(t, \mathbf{x}) = f; \\ bu(t, \mathbf{x}) = g, \ (t, \mathbf{x}) \in \Gamma \end{cases} \tag{9}$$

To simplify the discovery process, we state only Dirichlet-type boundary conditions. While the choice of boundary conditions not being a severe issue for the equations of up to second order, where the problem statement does not diverge from the conventional ones, this approach may face difficulties on equations of higher orders. As a temporary way of providing a PDE-solving algorithm with sufficient conditions, the field values in additional areas inside the domain are utilized. For example, if the equation has third order, the algorithm is provided with values on boundaries and in the center of the domain.

The equation solution task is reduced the optimization problem, stated in Eq. 10, where $||\cdot||_i$ and $||\cdot||_j$ are norms of arbitrary and not necessarily same orders i and j, and λ is a constant.

$$(||L\widetilde{u}(t, \mathbf{x}) - f||_i + \lambda||b\widetilde{u}(t, \mathbf{x}) - g||) \longrightarrow \min_{\widetilde{u}} \tag{10}$$

Due to numerical limitations, the differential operator L is substituted by the approximate one $\overline{L}$. In a generalized approach to the PDE solution, the boundary operator b will be substituted to the approximate operator $\overline{b}$. However, as it was stated earlier, the equation discovery process requires Dirichlet boundary conditions, the definition of which does not cause approximation errors. Calculations of partial derivatives for the operators are done with the finite-difference scheme Eq. 11, where for example, the first time derivative is considered. Δt denotes the time step of the domain grid.

$$\frac{\partial \widetilde{u}(t, \mathbf{x})}{\partial t} = \frac{\widetilde{u}(t + \Delta t, \mathbf{x}) - \widetilde{u}(t - \Delta t, \mathbf{x})}{2 * \Delta t} \tag{11}$$

In the stated problem the task is the selection of function $\widetilde{u}(x, \mathbf{x})$, matching the minimum value of functional on Eq. 10. For these purposes the parameterized function $\widetilde{u}(t, \mathbf{x}, \mathbf{\Theta})$: $\mathbb{R}^{k+1} \rightarrow \mathbb{R}$, where $\Theta = (\theta_1, ..., \theta_{n\_params})$ is the parameter vector for this specific function type, is chosen. Generally,

the class of parameterized function can be arbitrary. The optimization problem can be posed as stated in the Eq. 12.

$$(||L\widetilde{u}(t,\mathbf{x},\mathbf{\Theta}) - f||_i + \lambda||b\widetilde{u}(t,\mathbf{x},\mathbf{\Theta}) - g||) \longrightarrow \min_{\mathbf{\Theta}} \quad (12)$$

In this research, fully-connected artificial neural network is selected to represent the function $\widetilde{u}(t,\mathbf{x},\mathbf{\Theta})$. The question of selecting the best architecture of the ANN to represent the PDE solution is not of interest in this work. The search for parameter vector Θ is held by the training process of the neural network. The details of the equation solver algorithms are presented in the Alg. 2

Data: Encoded equation and boundary conditions, initial NN model *model*
Result: Trained neural network *model* that approximates the solution
Compute *model* Sobolev space norm *min_norm*;
for *NN in cache* **do**
  Train model to repeat NN output;
  Apply differential operator to trained model;
  Compute Sobolev space norm *norm_curr* **if** *norm_curr* < *min_norm* **then**
    model=trained model ;
    min_norm = *norm_curr*
  else
    pass
while *patience* < *threshold* **do**
  Apply differential operator to trained model;
  Compute Sobolev space norm *norm* ;
  if *norm oscilates near the same value* **then**
    patience = *patience* + 1
  if *norm is not improved in improving_patience steps* **then**
    patience = *patience* + 1
  Gradient descent step for *model* with respect to *norm*;

Algorithm 2: The pseudo-code of an equation solver algorithm

V. Validation

The algorithm was tested on the synthetic data to evaluate the effects of the newly introduced PDE solver-based fitness function evaluation technique. The necessity of using synthetic data in the experiment is created by the fact, that for the validation purposes we need to know the true structure of the equation. With this approach the performance of the algorithm on the experimental data can be evaluated with the success rate: fraction of independent launches, that result in the discovery of correct equations. By the correct results we denote the equations with correct structure (i.e. set of terms), and with coefficients, that insignificantly (up to 5%) deviating form the ground truth. For these purposes, the solution of the Korteweg-de Vries equation (Eq. 13) added was utilized. The equation was solved on a grid of 31×31 points with the dimension t representing time and x - space.

$$u_t + 6uu_x + u_{xxx} = f(x,t) \quad (13)$$

Following forcing, initial and boundary conditions as shown in Eq. 14 were applied.

$$\begin{gathered} f(x,t) = \cos t \sin x \\ u(x,0) = 0 \\ [u_{xx} + 2u_x + u]\Big|_{x=0} = 0 \\ [2u_{xx} + u_x + 3u]\Big|_{x=1} = 0 \\ [5u_x + 5u]\Big|_{x=1} = 0 \end{gathered} \quad (14)$$

The data for the numerical solution was obtained using the Wolfram Mathematica 12.3 software to avoid any numerical errors and shown in Fig. 4.

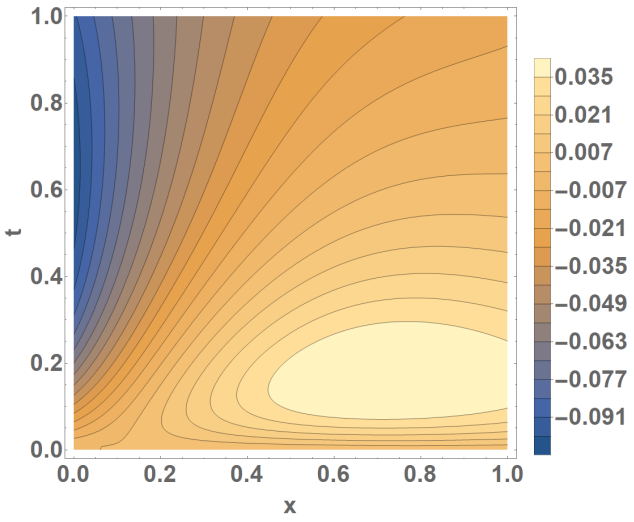

Fig. 4. Visualization of the Korteweg-de Vries equation solution, used during the experiments

While, as it was stated earlier, the convergence on noiseless data was proved to be rather robust, the additional Gaussian noise ($\epsilon \sim \mathcal{N}(\mu = 0; \sigma = n * ||u(t)||,\ n = 0.1, 0.2, ..., 0.9)$) was added into the data for more detailed analysis of algorithm performance. To assess the changes in algorithm behavior, the success rate, manifesting in the fraction of algorithm launches, when it converges to the correct equation (for one point on the Pareto frontier), was utilized. The motivation behind the selection of this metric was driven by the fact that it is consistent even for launches with different fitness function evaluation approaches. The improvements were made only in the selection operator, so no significant changes in the coefficient calculation quality were made. For each of the noise levels, 20 independent algorithm launches were commenced. The examples took forms of Pareto-frontiers, as presented in Fig. 6

From the success rate results, presented on the graph Fig. 5, the interpretation can be made that on high-quality data, the EA performance with both approaches tends to be similar.

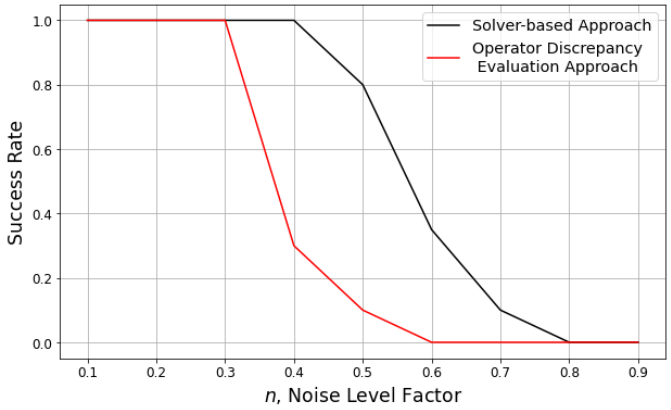

Fig. 5. Dependency of the evolutionary algorithm success rate in response to the noise in data. In the experiments, for each magnitude factor n the Gaussian noise with the standard deviation of $n * ||u(t)|$ was added to the solution of Korteweg-de Vries equation.

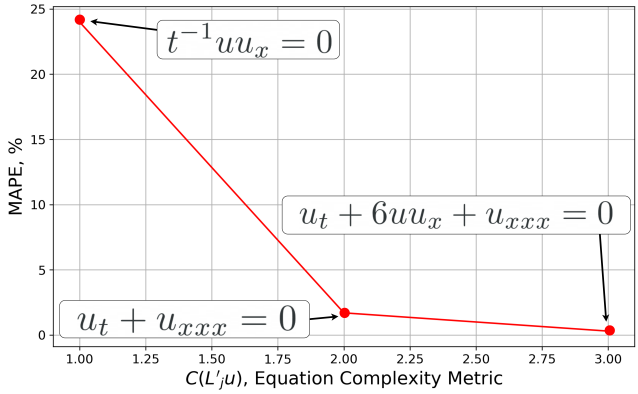

Fig. 6. An example of Pareto frontier, obtained from the multiobjective optimization.

When the algorithm is applied to lesser quality data containing significant noise, the discrepancy-based method tends to converge to incorrect structures, driven by high order derivatives approximation errors. The robustness of the solver-based approach allows it to have a considerably higher operational threshold, which makes. This quality is vital for the algorithm applications to real-world problems, where the significant noise related to the measurement process is unavoidable.

One of the issues linked with the use of the algorithm, solving partial differential equations to evaluate the quality of the proposed candidate, is the significant increase in required computational operations. Previously, to obtain the fitness function value, a single L2 norm of the matrix had to be calculated, while with the solver introduction, the algorithm has to perform additional ANN training procedures during every EA epoch.

VI. Conclusion

This paper proposes a novel candidate solutions quality evaluation method for the evolutionary algorithm of partial differential equation discovery. The improvements were aimed at increasing algorithm applicability to creating data-driven models for dynamical systems. The objective function is evaluated as the norm of PDE solution deviations from the expected input values on some grid in the modeled domain. The optimization-based method is employed to solve the differential equation proposed by the EA. In accordance with the specified differential and boundary operators, which define the boundary problem, representing the dynamical system, it can detect the parameters of function - solution of the equation.

The novel approach achieved better performance on artificially noised synthetic data, indicating that the evolutionary algorithm can model natural processes and dynamical systems based on the measurements datasets.

The direction of the future work on the related topic is guided by the high computational costs of the proposed approach. Its performance may be refined, and the program realization may be optimized To increase the viability of applying the evolutionary algorithm with PDE solver as the fitness function evaluation tool to practical tasks. Another promising approach is the application of techniques that reduce the total number of required PDE solving launches by making guesses of new candidate qualities compared to the previously processed ones.

Data and code avaliability

The numerical solution data and the Python code that partially reproduce the experiments are available at the GitHub repository [1].

Acknowledgements

This work was supported by the Analytical Center for the Government of the Russian Federation (IGK 000000D730321P5Q0002), agreement No. 70-2021-00141.

References

[1] Jun He, Tianshi Chen, and Xin Yao, "On the easiest and hardest fitness functions," *IEEE Transactions on evolutionary computation*, vol. 19, no. 2, pp. 295–305, 2014.

[2] Schmidt Michael and Hod Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 5923, pp. 81–85, 2019.

[3] H. Cao, L. Kang, Y. Chen, et al., "Evolutionary modeling of systems of ordinary differential equations with genetic programming," *Genetic Programming and Evolvable Machines*, vol. 1, pp. 309–337, 2000.

[4] Sébastien Gaucel, Maarten Keijzer, Evelyne Lutton, and Alberto TONDA, "Learning dynamical systems using standard symbolic regression," in *17. European Conference EuroGP 2014*, Grenade, Spain, Apr. 2014, vol. 8599 of *Lecture Notes in Computer Science*, p. np, Springer, Chapitre 3.

[5] H. Schaeffer, R. Caflisch, C. D. Hauck, and S. Osher, "Learning partial differential equations via data discovery and sparse optimization," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 2017.

[6] Hayden Schaeffer, "Learning partial differential equations via data discovery and sparse optimization," *Proc. R. Soc. A*, vol. 473, no. 2197, pp. 20160446, 2017.

[7] Samuel H. Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz, "Data-driven identification of parametric partial differential equations," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 2, pp. 643–660, 2019.

[8] Linan Zhang and H. Schaeffer, "On the convergence of the sindy algorithm," *Multiscale Model. Simul.*,, vol. 17(3), pp. 948–972, 2019.

[1] https://github.com/ITMO-NSS-team/EPDE

[9] M Raissi, P Perdikaris, and GE Karniadakis, "Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10566*, 2017.

[10] Zichao Long, Yiping Lu, and Bin Dong, "Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network," *Journal of Computational Physics*, vol. 399, pp. 108925, 2019.

[11] Mikhail Maslyaev, Alexander Hvatov, and Anna V Kalyuzhnaya, "Partial differential equations discovery with epde framework: application for real and synthetic data," *Journal of Computational Science*, p. 101345, 2021.

[12] Hao Xu, Haibin Chang, and Dongxiao Zhang, "Dlga-pde: Discovery of pdes with incomplete candidate library via combination of deep learning and genetic algorithm," *Journal of Computational Physics*, vol. 418, pp. 109584, 2020.

[13] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis, "Deepxde: A deep learning library for solving differential equations," *SIAM Review*, vol. 63, no. 1, pp. 208–228, 2021.

[14] Mikhail Maslyaev and Alexander Hvatov, "Multi-objective discovery of pde systems using evolutionary approach," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 596–603.

[15] Ke Li, Kalyanmoy Deb, Qingfu Zhang, and Sam Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2014.