An Interpretable Approach to the Solutions of High-Dimensional Partial Differential Equations

Lulu Cao[1,2], Yufei Liu[1,2], Zhenzhong Wang[3], Dejun Xu[1,2], Kai Ye[1,2], Kay Chen Tan[3], Min Jiang[1,2*]

[1] School of Informatics, Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, Xiamen University

[2] Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University

[3] Department of Computing, The Hong Kong Polytechnic University

lulucao@stu.xmu.edu.cn, phenanthra@stu.xmu.edu.cn, zhenzhong16.wang@connect.polyu.hk, xudejun@stu.xmu.edu.cn, yekai@stu.xmu.edu.cn, kaychen.tan@polyu.edu.hk, minjiang@xmu.edu.cn

Abstract

In recent years, machine learning algorithms, especially deep learning, have shown promising prospects in solving Partial Differential Equations (PDEs). However, as the dimension increases, the relationship and interaction between variables become more complex, and existing methods are difficult to provide fast and interpretable solutions for high-dimensional PDEs. To address this issue, we propose a genetic programming symbolic regression algorithm based on transfer learning and automatic differentiation to solve PDEs. This method uses genetic programming to search for a mathematically understandable expression and combines automatic differentiation to determine whether the search result satisfies the PDE and boundary conditions to be solved. To overcome the problem of slow solution speed caused by large search space, we propose a transfer learning mechanism that transfers the structure of one-dimensional PDE analytical solution to the form of high-dimensional PDE solution. We tested three representative types of PDEs, and the results showed that our proposed method can obtain reliable and human-understandable real solutions or algebraic equivalent solutions of PDEs, and the convergence speed is better than the compared methods. Code of this project is at https://github.com/grassdeerdeer/HD-TLGP.

Introduction

A partial differential equation (PDE) is an equation that contains an unknown function and its partial derivatives. The unknown function is a multivariable function, and the independent variables include time and space. Solving PDEs analytically is of great significance for understanding phenomena in physics (Roubíček 2013), engineering (Çayan, Özhan, and Sezer 2022), finance (Acevedo and Lelièvre 2018), and other fields. Analytical solutions of PDEs can demonstrate the interactions between variables affect the results, which provides a clear basis for engineers to understand the underlying physical principles of the problem and make more informed design decisions (Oh et al. 2023a). However, it is challenging to obtain analytical solutions of PDEs in real-world problems. Many PDEs do not have analytical solutions and require numerical methods to solve them (Gugat 2006). Moreover, even if an analytical solution exists, the solving process may be very complex especially when the dimension of the problem increases (Abraham-Shrauner 2018; Gockenbach 2011). Therefore, researchers have been working to find more efficient methods for solving analytical solutions of PDEs.

Traditional numerical methods (Prathap and Rao 2023; Mu and Kim 2023; Zheng and Wang 2023; Du and Cai 2021; Lai et al. 2021; Gao et al. 2019) and deep learning methods (Choi, Kim, and Hong 2023; Tang et al. 2023; Peng, Hu, and Xu 2023) are commonly used to solve PDEs. However, these methods have some limitations compared to solving analytical solutions directly. Numerical methods usually require discretization of the problem, which introduces discretization errors (Li et al. 2020a). In addition, the convergence speed of numerical methods may be slow and requires a lot of computational resources (Karniadakis et al. 2021). Deep learning methods can learn the solution directly from data and can be used to solve PDEs in complex geometries without the need for a structured grid (Ta et al. 2023; Koyamada et al. 2021). However, these learning-based methods require a large amount of training data and need to adjust many hyper-parameters (Lu, Jin, and Karniadakis 2019).

In addition to solving ability and efficiency, interpretability is also indispensable in practical applications of PDE solvers. Compared with a black-box solver, an interpretable one can provide some clues to the process of solving the equation, thus helping users understand the physical laws of the original problem behind the PDEs. Disappointingly, the solutions obtained by both the traditional numerical methods and deep learning methods are not interpretable (Oh et al. 2023b). Therefore, assigning interpretability to efficient PDE solvers is of practical significance and has not yet been tackled.

Symbolic regression has been proven to be an effective tool to achieve interpretability in many fileds, which can find a function that best fits the given data by searching the space of mathematical expressions (Dong et al. 2023; Chang et al. 2023). Genetic programming is often used for sym-

*Corresponding author: Min Jiang, minjiang@xmu.edu.cn

bolic regression by simulating natural selection and genetic mutation (Haider et al. 2023). In (Oh et al. 2023a), genetic programming is introduced to search for a function that satisfies the given PDE and boundary conditions. By designing the search space and fitness function reasonably, genetic programming can effectively solve the analytical solution of PDEs for low-dimensional problems.

However, when the dimension increases, the effectiveness of genetic programming for solving PDE may deteriorate. This is because the search space for high-dimensional problems is very large, and genetic programming may have difficulty finding effective analytical solutions in a limited amount of time. In addition, the complexity of high-dimensional problems also introduces difficulties for the design of genetic programming. Therefore, solving PDEs using genetic programming still faces some challenges for high-dimensional problems. By addressing this problem, we can endow the PDE solver with efficient solving ability and interpretability in high-dimensional problems, which is of great significant in practical applications.

It is worth noting that, the PDEs that describe the same physical law have similar structures in different dimensions. We infer that physical laws have similarities in different dimensions, so the solutions of PDEs that describe physical laws may have correlations between different dimensions. Such correlations can facilitate the handling of high-dimensional problems. In addition, efficient individual selection and simplification in genetic programming can improve the convergence speed and diversity (Liu et al. 2016; Karniadakis et al. 2021).

In this paper, we propose a genetic programming symbolic regression algorithm based on structural transfer mechanism and automatic differentiation (HD-TLGP) for solving the analytical solution of PDEs. The main contributions are as follows:

- A structural transfer mechanism is proposed to guide the search direction of the solution of high-dimensional PDEs by transferring the structure of the analytical solution of one-dimensional (1D) problems to high-dimensional problems, thereby accelerating the convergence speed.
- Automatic differentiation is introduced to evaluate whether an individual can be used as a solution to PDEs and speed up evaluation.
- Pruning operators are introduced to simplify individuals and improve population diversity, which improves the quality of interpretable expressions.

Preliminary Studies

Partial Differential Equation

A partial differential equation (PDE) is an equation that contains partial derivatives of an unknown function of several independent variables. It describes the relationship between the independent variables, the unknown function, and its partial derivatives.

Let $\mathcal{L}$ be a differential operator, then a PDE can be formalized as $\mathcal{L}(f(\mathbf{X})) = 0$, where $f(\mathbf{X})$ is the unknown function and $\mathbf{X}$ is the variable vector. Solving a PDE means finding the mathematical expression of the unknown function $f(\mathbf{X})$ that satisfies the PDE.

Related Work

To solve PDE, researchers have proposed many efficient methods in the past few decades. Currently, widely used techniques can be divided into two categories: the traditional numerical methods and the deep learning-based methods.

The Traditional Numerical Methods mainly include finite difference method (FDM), finite element method (FEM), finite volume method (FVM), and spectral method. FDM replaces derivatives in the PDE with finite differences (Thomas 2013). FEM (Felippa 2004) and FVM (Moukalled et al. 2016) discretize the domain and approximate solutions using polynomial basis functions. Spectral methods use Fourier series or Chebyshev polynomials to approximate the solution of PDE (Bernardi and Maday 1997). These numerical methods have their own advantages and disadvantages, and the choice of method depends on the specific problem being solved.

Traditional numerical methods also have some limitations. They may not be able to handle complex geometries and boundary conditions. Numerical methods also require a lot of computing resources and time, so efficiency and accuracy need to be balanced (Karniadakis et al. 2021). Therefore, many scholars turn to the deep learning methods.

Deep Learning-Based Methods for solving PDE have made much progress in recent years. These methods fall into three categories: finite-dimensional operators, physics-informed neural networks (PINNs), and neural operators.

Finite-dimensional operators use deep convolutional neural networks to parameterize the solution operator (Kag and Saligrama 2022; Nikolopoulos, Kalogeris, and Papadopoulos 2022). These approaches are mesh-dependent and different resolutions result in different errors. PINNs integrate the information from both the measurements and PDE by embedding the PDE into the loss function of a neural network using automatic differentiation (Vadyala and Betgeri 2023; Giampaolo et al. 2022). Neural operators are neural networks that are trained to approximate the mapping from PDE to its solution, and then build a fast and resolution-independent solver for solving PDE (Lu, Jin, and Karniadakis 2019; Li et al. 2020b). However, deep learning methods also have some limitations, including their reliance on large amounts of data, and lack of interpretability. In the context of engineering mechanics, lacking interpretability can perpetuate an inherent lack of understanding about the generated model and its prediction capabilities and limitations.

The mentioned methods provide a numerical approximate solution to PDE. However, these numerical methods lack interpretability, which makes it difficult for engineers to understand the underlying physical principles of a problem and make informed design decisions. Genetic programming-based symbolic regression (GPSR) is used to solve PDE thus its inherent interpretability and recent successful application to physics and mechanics problems (Versino, Tonda, and Bronkhorst 2017; Bomarito et al. 2021).

GPSR (Augusto and Barbosa 2000) is a type of symbolic regression that uses genetic programming to evolve mathematical expressions that approximate the solution of a PDE. Burgess *et. al.* (Burgess 1999) first use GPSR to solve differential equations. Cao *et. al.* (Cao et al. 2000) embed a genetic algorithm in genetic programming to discover and optimize the structure of mathematical expressions. Iba *et. al.* (Iba 2008) inferred differential equation by genetic programming (GP) and the least mean square (LMS). Extending beyond previous works, Hongsup Oh *et. al.* (Oh et al. 2023a) augment GPSR with a physics-regularized fitness function, called PR-GPSR.

However, with the increase of dimension, the relationship and interaction between variables become more complex, and it is difficult for existing methods to quickly provide interpretable solutions for high-dimensional PDEs. Therefore, in this work, we focus on how to reduce the difficulty of GPSR to discover the real solution or algebraic equivalent solution of the high-dimensional PDE and accelerate the convergence rate of the algorithm.

Transfer Learning in Genetic Programming

In the genetic programming (GP) realm, transfer learning aims to extract knowledge from solved source optimization problems to improve the efficiency and/or effectiveness of solving the target optimization problem. There are only a small number of studies on Transfer learning in Genetic Programming (TLGP) in the literature. Transfer learning in genetic programming can be classified into two categories: individual-based transfer learning, and instance-based transfer learning.

Individual-based transfer learning was proposed and implemented by transferring a number of good individuals or sub-individuals directly from the source to the target problem (Haslam, Xue, and Zhang 2016; Ardeh, Mei, and Zhang 2021; Iqbal, Zhang, and Xue 2016). Instance-based transfer learning is a type of transfer learning that reuses a part of the data from the source domain to the target domain by reweighting the data (Chen, Xue, and Zhang 2019a,b; Ardeh et al. 2022).

The methods mentioned above are only valid when the source domain and target domain are in the same solution domain. However, for low-dimensional and high-dimensional PDEs that describe the same physical regularity, there are differences in variables and solution domains. Therefore, we focus on how to align the solution domains of low-dimensional and high-dimensional equations and achieve transfer learning.

Proposed Algorithm

Overall Framework

The overall framework of the proposed method (HD-TLGP) is illustrated in Figure 1. The evolution process follows the basic framework of GPSR. Its core part is the use of prior knowledge (Step 1-2&5), pruning operators (Step 4) and evaluation functions (Step 3).

HD-TLGP first uses the structural transfer mechanism to transfer the structure of 1D PDE analytical solution to the form of high-dimensional PDE solution. If 1D PDE has no known solution, it can be obtained by GP. Based on the above algorithm, a knowledge base can be obtained to guide the exploration of solution of high dimensional PDE (Step 1). Then, HD-TLGP uses the knowledge base to improve the structure of partial individuals in the initial population (Step 2). During the evolution process, HD-TLGP uses the transfer rate $trpb$ to control when to improve the structure of the individual using the knowledge base (Step 5).

To improve the performance of HD-TLGP, we use gradient descent to optimize the constants in the generated expression and use automatic differentiation to evaluate the searched expression (Step 3). We also random perform the two pruning operators mentioned above (Step 4) to reduce the complexity of expression. When the stop condition is satisfied, we will use the best individual ind^* of the final population as the solution to the PDE.

Structural Transfer Mechanism

Genetic programming effectively solves PDEs, but struggles with high-dimensional ones due to the vast search space. To overcome this, HD-TLGP incorporates transfer learning.

The physical laws are expressed in terms of mathematical equations, and these equations have the same form in all dimensions. For example, consider the wave equation, which describes the behaviour of waves such as light and sound. The general form of the wave equation in n dimensions is given Eq. 1:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u, \tag{1}$$

where u is the displacement of the wave at position x and time t, ∇^2 is the Laplacian operator, and c is the speed of the wave. The wave equation has the same form in all dimensions, whether it is one-dimensional, two-dimensional, or three-dimensional. So we infer that the solutions of PDEs in different dimensions may be correlated.

Based on the above discussion, we propose a structural transfer mechanism. It transfers the prior knowledge by transferring the structure of the 1D PDE analytic solution to the form of a higher-dimensional PDE solution. In order to better understand the structural transfer mechanism proposed in this paper, we first give the mathematical definition of the structural transfer mechanism in solving PDE. Definition introduces the notion of domain. Definition introduces the notion of task.

(Domain). A symbolic expression space is a set of all possible symbolic expressions. A domain consists of a symbolic expression space. It is denoted as $\mathcal{D} = \{\mathcal{X}\}$, where $\mathcal{X}$ is a symbolic expression space.

(Task). Given a domain $\mathcal{D}$ define a task $\mathcal{T} = Y$ which consists of a specified PDE Y and an evaluation function f.

Let D_S, D_T be the source domain space and target domain space. T_S, T_T be the source learning task (solving 1D PDE) and targeted learning task(solving higher-dimensional PDE), respectively. Then structural transfer mechanism aims to improve the search efficiency of the solution that can satisfy the target task T_T using its prior knowledge, i.e, from $\mathcal{D}_S, \mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$.

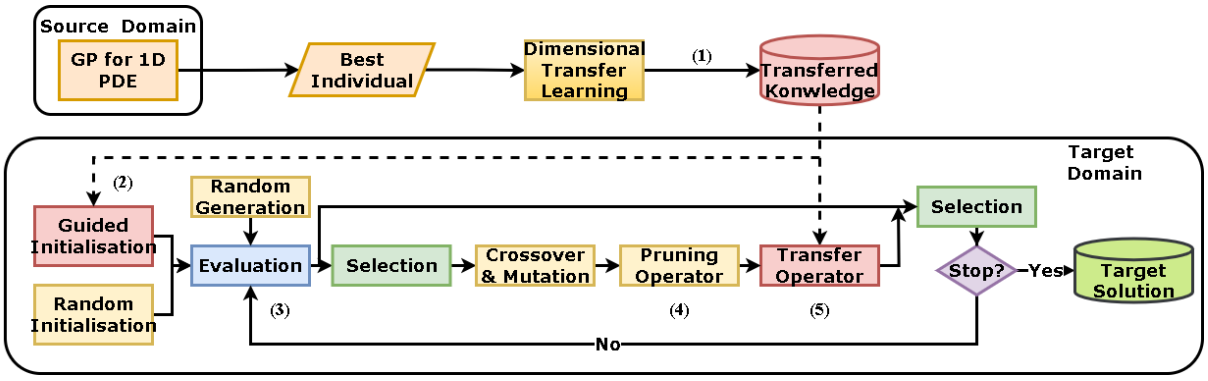

Figure 1: The pipeline of the proposed HD-TLGP: Step 1. The Structural Transfer Mechanism: transfer the structure of a 1D PDE analytical solution to the form of a high-dimensional PDE solution. Step 2. Guided Initialisation: improve the structure of partial individuals in the initial population by the knowledge base. Step 3. Evaluate via automatic differentiation: during the evolution process, use gradient descent to optimize the constants in the generated expression and use automatic differentiation to evaluate the searched expression. Step 4. Pruning Operators: Randomly perform the two pruning operators to reduce the complexity of the individual. Step 5. Transfer operator: improve the structure of the individual using the knowledge base.

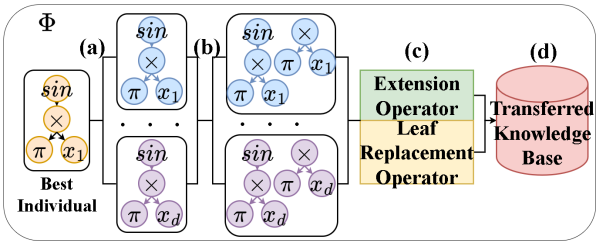

Figure 2: The process of the structural transfer operator Φ: (a) Perform the substitution operator. (b) Extract the subtree. (c) Perform the extension operator and leaf replacement operator. (d) The transferred knowledge base that can guide the search direction of solutions that satisfy the target task T_T.

Let f_s^* be the optimal solution on the source domain D_s and Φ be a structure transfer operator. The process of extending the optimal solution f_s^* to the target domain D_T can be represented as $\Phi\left(f_S^*\right) = F_T$, where F_T is the set of possible forms in which the optimal solution f_s^* extends to a higher dimensional. Figure 2 illustrates the process of using a structure transfer operator Φ to extend the optimal solution of a 1D PDE to the form of a high-dimensional PDE solution.

Assuming that the analytical solution of a 1D PDE is $u(x_1) = \sin(\pi \times x_1)$. The d dimensional PDE, which describes the same physical law as the 1D PDE, has the same form and parameters. It only has $d-1$ more dimensional-related variables $(x_2, , x_d)$ than the 1D PDE. We can replace the variable x_1 in the analytical solution of the 1D PDE with the variables $x_2, , x_d$ as Eq. 2.

$$[[u(x_1)/x_1 \rightarrow x_i]] = \sin(\pi \times x_i), \quad i = 2, , d, \qquad (2)$$

where $[[]]$ represent substitution operator. The set of formulas generated by the substitution operator is called M_1, which has a size of d.

The formulas in M_1 have the same structure but different

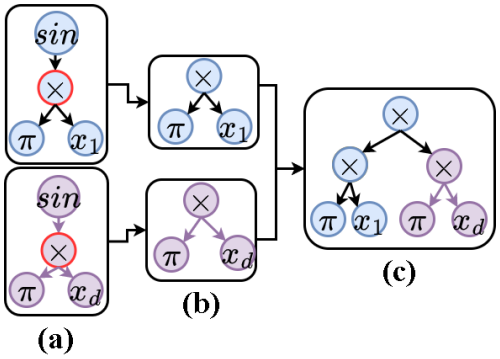

Figure 3: Extension Operator: Let $d = 2$, (a) represents the set M_1 generated by the substitution operator. Then select the internal node in the same position (circled in red). (b) represents the extracted subtree rooted by the internal node. Join these subtrees by multiplication as in (c).

variables. Each formula is represented by a tree structure. Then iterate through the internal nodes in each tree and extract the subtree rooted with the internal node as root. All subtrees in the same position of formulas in M_1 are joined by mathematical operator (e.g. +, ×). Figure 3 illustrates this process by using the ×.

The set M_1 generated by the substitution operator, the extracted subtree and the formula set M_2 generated by the extension operator form the set M_3. Randomly select two formulas in the set M_3, and replace the leaf node of one formula with the other formula to generate new formula. Perform $|M_3|$ times such Leaf Replacement Operator. The set M_3 and the individuals generated by the Leaf Replacement Operator form a new set, which we call Transferred Knowledge Base. Transferred Knowledge Base can guide the search direction of solutions that satisfy the target task T_T. We summarize the process of establishing a Transferred

Algorithm 1: Establishing a Transferred Knowledge Base

Input: f_s^* (The analytic solution of 1D PDE), d (dimension of high-dimensional PDE)
Output: $knowledge\_base$

Obtain M_1 generated by the substitution operator.
for $exp \in$M$_1$ **do**
  for internal node $i\_node \in exp$ **do**
    Extract the subtree rooted at i_node.
  end for
end for
Obtain M_3 generated by the extension operator.
$knowledge\_base \leftarrow \{\}$
for i $\in\{1, ..., |$M$_3|\ \}$ **do**
  new_exp $\leftarrow$ Perform Leaf Replacement Operator.
  knowledge_base $\leftarrow$ knowledge_base $\cup\{$new_exp$\}$.
end for
knowledge_base $\leftarrow$ knowledge_base $\cup M_3$.
return knowledge_base

Knowledge Base in Algorithm 1.

Evaluate via Automatic Differentiation

To verify if the expression obtained is a solution to the PDE, we substitute it into the PDE and check if it satisfies the equation. This process involves taking derivatives of the function. There are three common methods for computing derivatives of functions: Automatic differentiation (AD) (Baydin et al. 2018), symbolic differentiation (Grabmeier, Kaltofen, and Weispfenning 2003), and numerical differentiation (Burden, Faires, and Burden 2015). AD can calculate any order of derivatives to machine accuracy, while symbolic differentiation can only calculate the derivative of algebraic expressions. Numerical differentiation requires the use of finite differences to estimate the derivative of a function at a point. Therefore we use AD to calculate the derivatives of a function, and then use these derivatives to calculate how well the function satisfies the PDE.

Here, we present the procedure of evaluating the formula u with AD using the example of the 2D Poisson equation as Eq. 3. The detail of the evaluation is shown in Figure 4.

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + 2\pi^2 \sin(\pi x_1)\sin(\pi x_2) = 0. \tag{3}$$

Suppose we have an observation dataset $D\,(X_i, y_i)$ where $i \in 1, 2, \ldots, q$ based on the differential equation to be solved. X_i is a d-dimensional vector, where $d = 2$. A loss $\mathcal{E}$ can be used as a fitness function to evaluate whether the searched function expression u can be used as a solution to the PDE. It can be defined as Eq. 4:

$$\mathcal{E} = \mathcal{E}_{data} + \mathcal{E}_{PDE}, \tag{4}$$

where $\mathcal{E}_{data}$ is the data fitting error. $\mathcal{E}_{PDE}$ is the PDE fitting error. The PDE to be solved consists of PDE and initial and boundary conditions. Therefore, PDE fitting error is the sum of the error of these two parts.

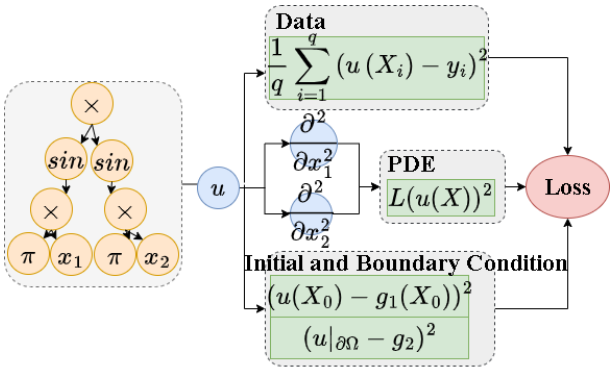

Figure 4: Fitness function: Evaluate the searched function expression u using the data fitting error and the PDE fitting error.

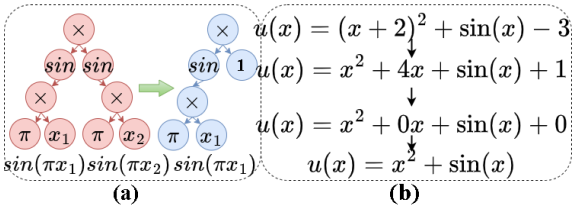

Figure 5: Pruning Operator: (a) Subtree pruning operator. (b) Simplification-pruning operator.

Pruning Operator

GP tends to produce more complex expressions with the increase of generation. These complex expressions can increase evaluation costs, reduce the interpretability and generalization ability of the expression. Pruning the expression is thus essential to improve its performance. We introduce two pruning operators: the subtree pruning operator, which works by selecting a random node in the expression tree that represents an individual and replacing the subtree rooted at that node with a constant value of 1 (Figure 5 (a)). And the simplification-pruning operator, which simplifies expressions and then randomly remove some structures in the formulas by setting some constants to zero (Figure 5 (b)).

Experiments

In this section, experiments on three types of PDE are conducted to evaluate the performance of the proposed HD-TLGP. We evaluate the performance of the proposed HD-TLGP compared with traditional numerical method (FEM) (Multiphysics 1998), deep-learning based method (PINN) and GPSR based method (PR-GPSR) (Cao et al. 2023). Then we give the optimal solutions of HD-TLGP and PR-GPSR to verify that our proposed method can obtain more reliable and human-understandable real solutions or algebraic equivalent solutions of PDE. In addition, we also plot the convergence traces of the best fitness function values for each generation. The effectiveness of pruning operators is also verified.

Name	PDE	Initial Condition / Boundary Condition
heat1	$\frac{\partial u}{\partial t} - 0.4\left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}\right) = 0, \mathbf{x} \in [0,1]^2, t \in (0,1]$	$u(\mathbf{x}, 0) = \delta(\mathbf{x})$
heat2	$\frac{\partial u}{\partial t} - 0.4\left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2}\right) = 0, \mathbf{x} \in [0,1]^3, t \in (0,1]$	$u(\mathbf{x}, 0) = \delta(\mathbf{x})$
heat3	$\frac{\partial u}{\partial t} - \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}\right) = 0, \mathbf{x} \in [0,1]^2, t \in (0,1]$	$u(\mathbf{x}, 0) = \delta(\mathbf{x})$
heat4	$\frac{\partial u}{\partial t} - \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2}\right) = 0, \mathbf{x} \in [0,1]^3, t \in (0,1]$	$u(\mathbf{x}, 0) = \delta(\mathbf{x})$
poisson1	$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + 2\pi^2 \prod_{i=1}^{2} \sin(\pi x_i), \mathbf{x} \in [0,1]^2$	$u(\mathbf{x}) = 0, \mathbf{x} \in \partial[0,1]^2$
poisson2	$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2} + 3\pi^2 \prod_{i=1}^{3} \sin(\pi x_i), \mathbf{x} \in [0,1]^3$	$u(\mathbf{x}) = 0, \mathbf{x} \in \partial[0,1]^3$
poisson3	$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + 1 = 0, \mathbf{x} \in \{(\mathrm{x}_1, x_2) \vert x_1^2 + x_2^2 \leq 1\}$	$u(\mathbf{x}) = 0, \mathbf{x} \in \ \{(x_1, x_2) \vert x_1^2 + x_2^2 = 1\}$
poisson4	$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2} + 1 = 0, \mathbf{x} \in \{(x_1, x_2, x_3) \vert x_1^2 + x_2^2 + x_3^2 \leq 1\}$	$u(\mathbf{x}) = 0, \mathbf{x} \in \{(x_1, x_2, x_3) \vert x_1^2 + x_2^2 + x_3^2 = 1\}$
advection1	$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = 0, \mathbf{x} \in [0,1]^2, t \in (0,2]$	$u(\mathbf{x}, 0) = x_1 + x_2$
advection2	$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} + \frac{\partial u}{\partial x_3} = 0, \mathbf{x} \in [0,1]^3, t \in (0,2]$	$u(\mathbf{x}, 0) = x_1 + x_2 + x_3$
advection3	$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = 0, \mathbf{x} \in [0,1]^2, t \in (0,2]$	$u(\mathbf{x}, 0) = \sin(x_1) + \sin(x_2)$
advection4	$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} + \frac{\partial u}{\partial x_3} = 0, \mathbf{x} \in [0,1]^3, t \in (0,2]$	$u(\mathbf{x}, 0) = \sin(x_1) + \sin(x_2) + \sin(x_3)$

Table 1: Summary of PDE with their initial condition, boundary condition and analytical solution.

Experiment Settings

In this section, the details of the test problems are presented. We test the proposed method on the Heat equation, Poisson equation and Advection equation. Table 1 gives each PDE a simple name and describes the detailed information.

Performance Comparison

In this section, we compared the performance of the proposed HD-TLGP with the PDE solver, PINN (Karniadakis et al. 2021), and PR-GPSR (Cao et al. 2023). The PDE solver is provided by COMSOL Multiphysics (Multiphysics 1998), which uses the finite element method (FEM) to solve PDE. Physics-Informed Neural Networks (PINNs) are a popular state-of-the-art machine learning model used to solve PDE. They have recently gained a lot of research interest due to their ability to learn the underlying physics of a system and make accurate predictions. PR-GPSR is the latest GPSR for solving PDE. It has shown good performance in solving analytical solutions of simple PDE. Its settings stay the same as HD-TLGP.

Table 2 shows the mean squared error (MSE) and variance of error between the predicted and true values using the above algorithm. HD-TLGP achieved the highest number (7 out of 12) of the best results. PDE solver achieved 5 of the best results, and the PDE involved are relatively simple. The MSE of HD-TLGP is relatively stable. PDE Solver, PINN, and PR-GPSR only performed well on a few PDE with lower dimensions and relatively simple solutions. This further proves the effectiveness of HD-TLGP in solving high-dimensional PDE.

It is worth noting that the MSE and variance of PDE Solver on advection4 are very large. This may be due to the complex initial conditions of advection4, which leads to unstable numerical solutions and very large outliers. The data-driven method can solve the problem of PDE outliers by FEM. Other comparison algorithms add observation data in the optimization process, so the outlier problem only appears in PDE Solver. This further illustrates the limitations of traditional numeric approaches.

Symbolic Solution Comparison

HD-TLGP attempts to solve PDE by searching for formulas that satisfy PDE in symbolic expressions. The symbolic model is the language of natural science. Different from the deep-learning model, the symbolic model is compact, interpretable and has good generalization ability. Therefore, symbolic regression has a natural human-understandable nature.

Table 3 show the simplified optimal formula of HD-TLGP and SP-GPSR. Limited by the length of Table 3, some PR-GPSR solutions are not listed here. It can be seen that our method obtained the exact analytical solution within a limited number of iterations, while the other methods only obtained approximate solutions that can fit the observed data, but not the exact solution. This demonstrates thatHD-TLGP can accurately and efficiently give human-understandable solutions to PDE.

Convergence Trace Visualization

In this section, we plot the convergence traces of the best objective values obtained by the HD-TLGP and PR-GPSR over 100 generations. The convergence traces on the advection 1 instance are shown in Figure 6. From the convergence traces, we can find that compared with PR-GPSR, the objective values of our algorithm can quickly converge to a small value. This is because our method transfers the solutions that can guide the population to explore in a good direction.

Verification of Pruning Operators

In this section, we track the average length of the population before and after using the pruning operator. Figure 7 shows the average length of individuals in each generation over 100 generations. As the number of generations increases, HD-TLGP without a pruning operator tends to produce more complex expressions. The individuals of HD-TLGP with a pruning operator are relatively stable in length.

Name	HD-TLGP	PDE solver	PINN	PR-GPSR
heat 1	**1.65e-06±8.22e-07**	7.34e-02±3.65e-02	6.19e-02±2.63e-02	5.31e-00±8.48e-01
heat 2	**3.98e-09±2.84e-09**	2.55e-02±1.82e-02	2.22e-02±1.50e-02	2.39e-02±2.13e-01
heat 3	**2.33e-08±8.40e-09**	2.47e-02±8.9e-03	6.21e-02±2.62e-02	1.70e-02±1.63e-02
heat 4	**2.53e-08±1.48e-08**	5.6e-03±3.2e-03	4.59e-03±2.45e-03	4.90e-03±4.90e-03
poisson 1	4.70e-08±4.67e-08	**9.86e-12±8.12e-12**	6.12e-02±4.86e-03	5.13e-07±3.47e-07
poisson 2	1.33e-07±1.29e-07	**5.06e-09±4.80e-09**	8.10e-04±6.47e-04	1.15e-01±6.73e-02
poisson 3	1.04e-08±2.84e-09	**9.27e-13±6.52e-13**	4.19e-08±2.57e-14	3.44e-02±2.51e-02
poisson 4	1.53e-07±4.42e-08	**2.19e-11±1.09e-11**	3.24e-06±2.84e-06	1.99e-02±1.30e-02
advection 1	3.5e-07±1.00e-07	**2.33e-11±6.99e-12**	7.62e-03±3.57e-04	4.90e-03±2.00e-03
advection 2	**1.85e-32±1.85e-32**	1.15e-06±1.15e-06	6.56e-01±3.91e-01	7.55e-01±7.18e-01
advection 3	**4.75e-05±4.48e-05**	2.63e-02±2.09e-02	9.54e-03±1.26e-04	6.84e-01±6.82e-01
advection 4	**00e-00±00e-00**	2.60e+40±2.60e+40	5.49e-02±2.67e-02	9.93e-01±9.61e-01

Table 2: Summary of the models' performance for MSE and variance of error

Name	HD-TLGP	PR-GPSR
heat 1	$0.198\exp(-0.625x_1^2/t)\exp(-0.625x_2^2/t)/t$	$1.2019\exp(1.1839 * x_1)$
heat 2	$0.0887\exp(-0.625x_1^2/t)\exp(-0.625 * x_2^2/t)\exp(-0.625x_3^2/t)/t^{1.5}$	$0.0015\exp(t) * \exp(x_3)\exp(\exp(x_3))$
heat 3	$0.0795\exp(-x_1^2/(4t))\exp(-x_2^2/(4t))/t$	$0.1968 * x_1$
heat 4	$0.0224\exp(-x_1^2/(4t))\exp(-x_2^2/(4t))\exp(-x_3^2/(4t))/t^{1.5}$	$\exp(x_2\exp(x_3))\exp(-2.5502\exp(x_3))$
poisson 1	$\sin(3.141 * x_1) * \sin(3.142 * x_2)$	$\sin(3.14 * x_1) * \sin(3.14 * x_2)$
poisson 2	$\sin(3.142 * x_1) * \sin(3.140 * x_2) * \sin(3.142 * x_3)$	-
poisson 3	$-0.2502 * x_1^2 - 0.2500 * x_2^2 + 0.25014$	$-0.4637 * x_1^2 + 0.2365 * x_1 + 0.1426$
poisson 4	$-0.167 * x_1^2 - 0.167 * x_2^2 - 0.166 * x_3^2 + 0.167$	-
advection 1	$-1.999 * t + x_1 + x_2$	$-1.9640 * t + x_1 + 0.8572 * x_2$
advection 2	$-3.00 * t + x_1 + 1.00 * x_2 + 1.00 * x_3$	$-t^2 - t * x_1 + t * x_3 + x_1 - 0.682 * x_2$
advection 3	$-\sin(0.9838 * t - x_1) - \sin(0.9979 * t - x_2)$	$-\sin(0.9999 * t - 0.9999 * x_2)$
advection 4	$-\sin(t - x_1) - \sin(t - x_2) - \sin(t - x_3)$	-

Table 3: Optimal Solution of HD-TLGP and SP-GPSR

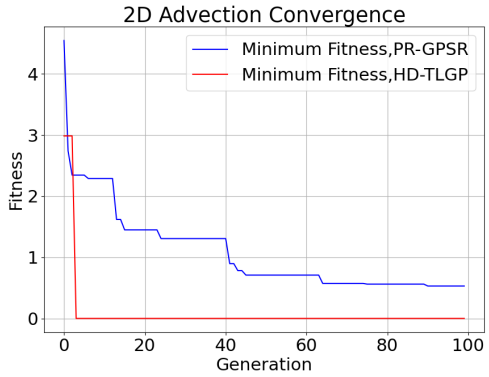

Figure 6: Convergence traces of best objective values obtained by HD-TLGP and PR-GPSR on the advection 1.

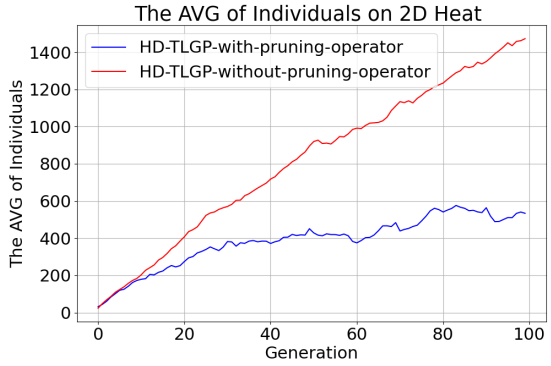

Figure 7: The average length of individuals in a population on the heat 1.

The complex expressions can increase evaluation costs, reduce the interpretability of the expression, and reduce its generalization ability. Therefore, the existence of pruning operators is very necessary.

Conclusion

In this paper, we propose a new method called HD-TLGP to solve PDE. The method proposes a structure transfer mechanism based on the analytic solution of 1D PDE to guide the exploration of solutions to high-dimensional PDE. The performance of the algorithm is further improved by using automatic differentiation and pruning operators. Experiments on three types of PDE show that the method can obtain reliable and human-understandable real or algebraic equivalent solutions to high-dimensional PDE, and its convergence speed is better than compared method.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under Grant No. 62276222.

References

Abraham-Shrauner, B. 2018. Analytic Solutions of Nonlinear Partial Differential Equations by the Power Index Method. *Symmetry*, 10(3): 76.

Acevedo, J. I.; and Lelièvre, T. 2018. A non linear approximation method for solving high dimensional partial differential equations: Application in finance. *Math. Comput. Simul.*, 143: 14–34.

Ardeh, M. A.; Mei, Y.; and Zhang, M. 2021. Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 26(4): 765–779.

Ardeh, M. A.; Mei, Y.; Zhang, M.; and Yao, X. 2022. Knowledge transfer genetic programming with auxiliary population for solving uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, 27(2): 311–325.

Augusto, D. A.; and Barbosa, H. J. 2000. Symbolic regression via genetic programming. In *Proceedings. Vol. 1. Sixth Brazilian symposium on neural networks*, 173–178. IEEE.

Baydin, A. G.; Pearlmutter, B. A.; Radul, A. A.; and Siskind, J. M. 2018. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18: 1–43.

Bernardi, C.; and Maday, Y. 1997. Spectral methods. *Handbook of numerical analysis*, 5: 209–485.

Bomarito, G.; Townsend, T.; Stewart, K.; Esham, K.; Emery, J.; and Hochhalter, J. 2021. Development of interpretable, data-driven plasticity models with symbolic regression. *Computers & Structures*, 252: 106557.

Burden, R. L.; Faires, J. D.; and Burden, A. M. 2015. *Numerical analysis*. Cengage learning.

Burgess, G. R. 1999. *Finding approximate analytic solutions to differential equations using genetic programming*. Citeseer.

Cao, H.; Kang, L.; Chen, Y.; and Yu, J. 2000. Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 1: 309–337.

Cao, L.; Zheng, Z.; Ding, C.; Cai, J.; and Jiang, M. 2023. Genetic Programming Symbolic Regression with Simplification-Pruning Operator for Solving Differential Equations. In *International Conference on Neural Information Processing*, 287–298. Springer.

Çayan, S.; Özhan, B. B.; and Sezer, M. 2022. An adaptive approach for solving fourth-order partial differential equations: algorithm and applications to engineering models. *Comput. Appl. Math.*, 41(8).

Chang, C.; Chiang, T.; Hsu, T.; Chuang, T.; Fang, W. Z.; and Yu, T. 2023. Taylor Polynomial Enhancer using Genetic Programming for Symbolic Regression. In Silva, S.; and Paquete, L., eds., *Companion Proceedings of the Conference on Genetic and Evolutionary Computation, GECCO 2023, Companion Volume, Lisbon, Portugal, July 15-19, 2023*, 543–546. ACM.

Chen, Q.; Xue, B.; and Zhang, M. 2019a. Differential evolution for instance based transfer learning in genetic programming for symbolic regression. In *Proceedings of the genetic and evolutionary computation conference companion*, 161–162.

Chen, Q.; Xue, B.; and Zhang, M. 2019b. Instance based transfer learning for genetic programming for symbolic regression. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, 3006–3013. IEEE.

Choi, J.; Kim, N.; and Hong, Y. 2023. Unsupervised Legendre-Galerkin Neural Network for Solving Partial Differential Equations. *IEEE Access*, 11: 23433–23446.

Dong, J.; Zhong, J.; Chen, W.; and Zhang, J. 2023. An Efficient Federated Genetic Programming Framework for Symbolic Regression. *IEEE Trans. Emerg. Top. Comput. Intell.*, 7(3): 858–871.

Du, S.; and Cai, Z. 2021. Adaptive Finite Element Method for Dirichlet Boundary Control of Elliptic Partial Differential Equations. *J. Sci. Comput.*, 89(2): 36.

Felippa, C. A. 2004. Introduction to finite element methods. *University of Colorado*, 885.

Gao, J.; Zhao, M.; Du, N.; Guo, X.; Wang, H.; and Zhang, J. 2019. A finite element method for space-time directional fractional diffusion partial differential equations in the plane and its error analysis. *J. Comput. Appl. Math.*, 362: 354–365.

Giampaolo, F.; Rosa, M. D.; Qi, P.; Izzo, S.; and Cuomo, S. 2022. Physics-informed neural networks approach for 1D and 2D Gray-Scott systems. *Adv. Model. Simul. Eng. Sci.*, 9(1): 5.

Gockenbach, M. S. 2011. *Partial Differential Equations - Analytical and Numerical Methods (2. ed.)*. SIAM. ISBN 978-0-89871-935-2.

Grabmeier, J.; Kaltofen, E.; and Weispfenning, V. 2003. *Computer algebra handbook: foundations, applications, systems*. Springer.

Gugat, M. 2006. Book Review: K.W. Morton, D.F. Mayers: Cambridge University Press, 2005. ISBN: 0-521-60793-0. 278 pp, paperback, US 43.00 - K.W. Morton and D.F. Mayers: Numerical solution of partial differential equations. *Math. Methods Oper. Res.*, 63(2): 385.

Haider, C.; de França, F. O.; Burlacu, B.; and Kronberger, G. 2023. Shape-constrained multi-objective genetic programming for symbolic regression. *Appl. Soft Comput.*, 132: 109855.

Haslam, E.; Xue, B.; and Zhang, M. 2016. Further investigation on genetic programming with transfer learning for symbolic regression. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 3598–3605. IEEE.

Iba, H. 2008. Inference of differential equation models by genetic programming. *Information Sciences*, 178(23): 4453–4468.

Iqbal, M.; Zhang, M.; and Xue, B. 2016. Improving classification on images by extracting and transferring knowledge in genetic programming. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 3582–3589. IEEE.

Kag, A.; and Saligrama, V. 2022. Condensing CNNs with Partial Differential Equations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, 600–609. IEEE.

Karniadakis, G. E.; Kevrekidis, I. G.; Lu, L.; Perdikaris, P.; Wang, S.; and Yang, L. 2021. Physics-informed machine learning. *Nature Reviews Physics*, 3(6): 422–440.

Koyamada, K.; Long, Y.; Kawamura, T.; and Konishi, K. 2021. Data-driven derivation of partial differential equations using neural network model. *Int. J. Model. Simul. Sci. Comput.*, 12(2): 2140001:1–2140001:19.

Lai, J.; Liu, F.; Anh, V. V.; and Liu, Q. 2021. A space-time finite element method for solving linear Riesz space fractional partial differential equations. *Numer. Algorithms*, 88(1): 499–520.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020a. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Stuart, A.; Bhattacharya, K.; and Anandkumar, A. 2020b. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33: 6755–6766.

Liu, Y.; Cheng, Z.-L.; Xu, J.; Yang, J.; and Wang, Q.-W. 2016. Improvement and validation of genetic programming symbolic regression technique of silva and applications in deriving heat transfer correlations. *Heat transfer engineering*, 37(10): 862–874.

Lu, L.; Jin, P.; and Karniadakis, G. E. 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*.

Moukalled, F.; Mangani, L.; Darwish, M.; Moukalled, F.; Mangani, L.; and Darwish, M. 2016. *The finite volume method*. Springer.

Mu, J.; and Kim, B. 2023. A Dynamic-Precision Bit-Serial Computing Hardware Accelerator for Solving Partial Differential Equations Using Finite Difference Method. *IEEE J. Solid State Circuits*, 58(2): 543–553.

Multiphysics, C. 1998. Introduction to COMSOL multiphysics®. *COMSOL Multiphysics, Burlington, MA, accessed Feb*, 9(2018): 32.

Nikolopoulos, S.; Kalogeris, I.; and Papadopoulos, V. 2022. Non-intrusive surrogate modeling for parametrized time-dependent partial differential equations using convolutional autoencoders. *Eng. Appl. Artif. Intell.*, 109: 104652.

Oh, H.; Amici, R.; Bomarito, G.; Zhe, S.; Kirby, R.; and Hochhalter, J. 2023a. Genetic Programming Based Symbolic Regression for Analytical Solutions to Differential Equations. *arXiv preprint arXiv:2302.03175*.

Oh, H.; Amici, R.; Bomarito, G. F.; Zhe, S.; Kirby, R. M.; and Hochhalter, J. D. 2023b. Genetic Programming Based Symbolic Regression for Analytical Solutions to Differential Equations. *CoRR*, abs/2302.03175.

Peng, Y.; Hu, D.; and Xu, Z. J. 2023. A non-gradient method for solving elliptic partial differential equations with deep neural networks. *J. Comput. Phys.*, 472: 111690.

Prathap, T.; and Rao, R. N. 2023. Uniformly convergent finite difference methods for singularly perturbed parabolic partial differential equations with mixed shifts. *J. Appl. Math. Comput.*, 69(2): 1679–1704.

Roubíček, T. 2013. *Nonlinear partial differential equations with applications*, volume 153. Springer Science & Business Media.

Ta, H.; Wong, S. W.; McClanahan, N.; Kimn, J.; and Fu, K. 2023. Exploration on Physics-Informed Neural Networks on Partial Differential Equations (Student Abstract). In Williams, B.; Chen, Y.; and Neville, J., eds., *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 16344–16345. AAAI Press.

Tang, S.; Feng, X.; Wu, W.; and Xu, H. 2023. Physics-informed neural networks combined with polynomial interpolation to solve nonlinear partial differential equations. *Comput. Math. Appl.*, 132: 48–62.

Thomas, J. W. 2013. *Numerical partial differential equations: finite difference methods*, volume 22. Springer Science & Business Media.

Vadyala, S. R.; and Betgeri, S. N. 2023. General implementation of quantum physics-informed neural networks. *Array*, 18: 100287.

Versino, D.; Tonda, A.; and Bronkhorst, C. A. 2017. Data driven modeling of plastic deformation. *Computer Methods in Applied Mechanics and Engineering*, 318: 981–1004.

Zheng, Z.; and Wang, Y. 2023. A second-order accurate Crank-Nicolson finite difference method on uniform meshes for nonlinear partial integro-differential equations with weakly singular kernels. *Math. Comput. Simul.*, 205: 390–413.