Interpretable Solutions for Multi-Physics PDEs Using T-NNGP

Lulu Cao[1,2,3], **Zexin Lin**[1,2], **Kay Chen Tan**[3], **Min Jiang**[1,2*]

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

[2] 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

[3] Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University

lulucao120@gmail.com, zexinlin@stu.xmu.edu.cn, kctan@polyu.edu.hk, minjiang@xmu.edu.cn

Abstract

Multiphysics simulation aims to predict and understand interactions between multiple physical phenomena, aiding in comprehending natural processes and guiding engineering design. The system of Partial Differential Equations (PDEs) is crucial for representing these physical fields, and solving these PDEs is fundamental to such simulations. However, current methods primarily yield numerical outputs, limiting interpretability and generalizability. We introduce T-NNGP, a hybrid genetic programming algorithm that integrates traditional numerical methods with deep learning to derive approximate symbolic expressions for multiple unknown functions within a system of PDEs. T-NNGP initially obtains numerical solutions using traditional methods, then generates candidate symbolic expressions via deep reinforcement learning, and finally optimizes these expressions using genetic programming. Furthermore, a universal decoupling strategy guides the search direction and addresses coupling problems, thereby accelerating the search process. Experimental results on three types of PDEs demonstrate that our method can reliably obtain human-understandable symbolic expressions that fit both the PDEs and the numerical solutions from traditional methods. This work advances multiphysics simulation by enhancing our ability to derive approximate symbolic solutions for PDEs, thereby improving our understanding of complex physical phenomena.

Code — https://github.com/grassdeerdeer/T-NNGP

1 Introduction

Multiphysics studies the interactions between multiple physical fields within a unified framework, typically described by systems of partial differential equations (PDEs) (Gao, Zahr, and Wang 2022). For example, in fluid dynamics, flow velocity and pressure are closely interconnected, where an increase in fluid velocity is generally accompanied by a decrease in static pressure.

Multiphysics simulation is crucial for analyzing complex engineering and scientific problems by solving the PDEs that represent these interactions (Pryor 2009). For example, in the design of a car engine, it is essential to consider multiple physical fields, including thermodynamics, fluid mechanics, and structural mechanics. Thermodynamics governs how fuel burns to generate energy (Struchtrup 2014), fluid mechanics dictates how air and fuel mix (Morrison 2013), and structural mechanics ensures that engine components can withstand stress and vibrations (Adhywinaya and Budiman 2022). The interplay and balance of these fields are vital for optimal engine performance. Multiphysics simulations guide the design process by ensuring proper interaction between these fields and validating the smooth and efficient operation of the engine.

Traditional methods for solving PDEs can be broadly categorized into two main groups: analytical methods and numerical methods. Analytical methods, such as separation of variables (Polyanin and Zhurov 2021) and Green's function method (Amaratunga and Williams 1993), yield exact solutions but often struggle with complex PDE systems. Numerical methods, which discretize the domain and approximate solutions at discrete points, offer an alternative. The accuracy of these methods depends on the degree of discretization, with finer discretization leading to higher accuracy. However, complex PDEs typically require substantial discretization, causing numerical methods to converge slowly and demand significant computational resources.

Deep learning-based methods for solving PDE have advanced significantly in recent years (Takamoto et al. 2022; Gong et al. 2024). These methods use neural networks to approximate the solutions of PDE (Karniadakis et al. 2021; Wang and Zhong 2024), offering a significant advantage by eliminating the need for problem discretization, thus avoiding errors associated with traditional numerical methods. However, despite their potential, deep learning methods face challenges, including the risk of overfitting and the need for extensive data.

Traditional numerical methods and deep learning-based approaches for solving PDE face two primary challenges. Firstly, these methods provide numerical values only within the solution domain of the physical field, failing to reveal the inherent laws governing the physical field. This limitation restricts understanding and hinders the ability to modify physical fields effectively. Secondly, most existing methods focus on PDE describing a single physical field, whereas multiphysics simulations often involve the coupling of mul-

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

tiple PDEs. In such cases, a variable or parameter in one equation depends on the solution of another. For example, consider the coupled problem of fluid mechanics and heat conduction. Here, fluid velocity and pressure influence temperature distribution, and temperature changes influence fluid density and viscosity (Wang et al. 2021). Solving these coupled problems requires considering boundary conditions, initial conditions, and interactions between the equations.

To address the first challenge, symbolic regression has emerged as an effective tool for achieving interpretability. It employs genetic programming to evolve mathematical expressions that approximate PDE solutions, helping users understand the underlying physical laws (Cao et al. 2023). However, these methods have been limited to solving single physical fields. For coupled physics problems, existing strategies include full coupling, which solves all fields simultaneously but is resource-intensive, and separation, which simplifies multiphysics problems into single physics fields but neglects interactions and relies heavily on prior knowledge. An effective and universal decoupling strategy is crucial for overcoming these challenges.

In this work, we propose a hybrid genetic programming algorithm (T-NNGP) that advances the field of solving the system of PDEs. T-NNGP integrates traditional numerical methods with deep learning to derive approximate symbolic expressions for multiple unknown functions within a system of PDEs, addressing the limitations of existing numerical methods in terms of interpretability and generalizability.

• Hybrid optimization strategy: This strategy uniquely combines traditional numerical methods with deep reinforcement learning to derive symbolic expressions for systems of PDEs. This hybrid approach leverages the strengths of both methods, significantly improving the accuracy and interpretability of the solutions compared to existing methods.

• Universal Decoupling Strategy: We introduce a universal decoupling strategy during the genetic programming phase. This strategy effectively guides the search direction and addresses the coupling problems inherent in multiphysics simulations, which are often overlooked by other methods.

Unlike traditional methods that primarily provide numerical output, T-NNGP produces human-understandable symbolic expressions. These expressions not only fit the system of PDEs but also align with the numerical solutions provided by traditional methods. This dual compatibility enhances the interpretability and generalizability of the solutions across different physical scenarios, making them more useful for understanding complex physical phenomena. We conduct extensive experiments on three different types of PDEs to validate the effectiveness of T-NNGP. The results demonstrate that our method consistently outperforms existing techniques in terms of obtaining reliable, interpretable, and generalizable symbolic expressions. This empirical validation underscores the practical applicability and superior performance of T-NNGP in diverse multiphysics simulations.

2 Preliminary Studies

Partial Differential Equations

A partial differential equation (PDE) is an equation that involves partial derivatives of an unknown function with respect to multiple independent variables. It describes the relationship between independent variables, unknown function, and its partial derivatives.

Assuming $\mathcal{L}$ is a partial differential operator, a PDE can be formally expressed as $\mathcal{L}(f(\mathbf{X})) = 0$, where $f(\mathbf{X})$ is the unknown function to be solved, $\mathbf{X}$ represents a vector of independent variables. Solving a PDE refers to finding a mathematical expression for the unknown function $f(\mathbf{X})$ that satisfies the PDE.

A system of partial differential equations (PDEs) is a mathematical tool used to describe multi-physics fields, which consists of multiple PDEs. Let $\{\mathcal{L}_i\}_{i=1}^n$ be a set of differential operators, then a system of PDEs can be formalized as $\{\mathcal{L}_i(\mathbf{u}(\mathbf{X})) = 0\}_{i=1}^n$, where $\mathbf{u}(\mathbf{X})$ is a vector of unknown functions, and $\mathbf{X}$ is a vector of variables. $\mathbf{u}(\mathbf{X})$ usually contain two or more functions. For example, if $\mathbf{u}(\mathbf{X}) = [u_1(\mathbf{X}), u_2(\mathbf{X}), \ldots, u_m(\mathbf{X})]$, then we need to find the expressions for all $u_i(\mathbf{X})$ (where $i = 1, 2, \ldots, m$) that satisfy the PDEs. m is the number of unknown functions.

Related Work

Researchers have proposed many efficient methods to solve PDE in the past few decades, which can be divided into three categories: traditional numerical methods, deep learning-based methods and symbolic regression-based methods.

Traditional Numerical Methods mainly include the finite difference method (FDM), finite element method (FEM), finite volume method (FVM) and spectral method. FDM is the most commonly used numerical method for solving PDE. The basic idea behind FDM is to substitute the derivatives appearing in the PDE with finite difference approximations (Robertsson and Blanch 2020; Wahyudi, Lestari, and Gapsari 2021; Ullah et al. 2021). FEM discretizes the domain into a set of overlapping meshes or "finite elements". Within each element, the solution is approximated using a polynomial basis function (Polycarpou 2022; Szabó and Babuška 2021; David Müzel et al. 2020). FVM involves dividing the domain into a set of control volumes and approximating the solution within each volume using a polynomial basis function (Ali et al. 2022; Muhammad 2021; Haider and Ahmad 2022). The spectral method uses Fourier series or Chebyshev polynomials to approximate the solution of PDE (Bernardi and Maday 1997; Burns et al. 2020). Each of these numerical methods has advantages and disadvantages, and the choice of which method to use depends on the specific characteristics of the problem being solved.

When solving the system of PDEs, two commonly used strategies can be combined with the traditional numerical methods mentioned above. These strategies are called the "full coupling method" and "separation method". The full coupling method solves all unknown physical fields simultaneously (Phan and He 2022; Zhang et al. 2022). It is similar

to the method for solving a single PDE, but it requires significant memory and computation time. The separation method sequentially analyses each PDE, using the results from previously solved equations to solve the next one (Zimmerman 2006). However, it needs to determine the solving order of the PDEs artificially according to the specific problem.

Deep Learning-Based Methods can be broadly grouped into three categories: finite-dimensional operators, physics-informed neural networks (PINNs) and neural operators. Finite-dimensional operators leverage deep convolutional neural networks to parameterize the operator that generates the solution for a given PDE (Kag and Saligrama 2022; Nikolopoulos, Kalogeris, and Papadopoulos 2022). These methods are mesh-dependent. Different mesh resolutions can lead to different levels of errors in the computed solution. PINNs integrate the information from both the observation dataset and PDE. They achieve this by embedding the PDE directly into the loss function of a neural network, leveraging automatic differentiation techniques (Cai et al. 2021a; Yu et al. 2022; Mao, Jagtap, and Karniadakis 2020). Neural operators are neural networks, specifically designed and trained to learn the mapping from a PDE to its solution. Once trained, these neural operators can quickly generate approximate mesh-independent solutions to the PDE for various initial conditions (Lu et al. 2021; Li et al. 2020; Cao, Hong, and Jiang 2024). (Cai et al. 2021b) improved one of neural operators, DeepOnet (Lu et al. 2021), to solve the system of PDEs.

Symbolic Regression-Based Methods (Udrescu and Tegmark 2020; Petersen et al. 2021; Mundhenk et al. 2021) can obtain approximate solutions to PDE by combining observation datasets with physical constraints based on PDE. Unlike traditional numerical methods and deep learning-based methods, this method provides solutions in the form of mathematical expressions. Researchers can intuitively grasp how the physical field described by the PDE evolves over time and space. In (Oh et al. 2023), GPSR is introduced to search for an expression that satisfies a given PDE and its initial conditions. By designing the search space and fitness function reasonably, GPSR can efficiently obtain analytical solutions for low-dimensional PDE. (Cao et al. 2023) introduce a new operator called the simplification-pruning operator to find expression solutions with high accuracy and conciseness. (Cao et al. 2024) proposed a method called HD-TLGP to solve high-dimensional PDE based on a structural transfer mechanism. However, it is important to note that symbolic regression-based methods are currently limited to solving single PDEs and cannot be directly applied to address the system of PDEs that involve coupled unknown functions.

3 Proposed Algorithm

In this section, we describe the details of the proposed T-NNGP. First, we present the framework of T-NNGP. Then, we show how T-NNGP combines the traditional numerical method with deep reinforcement learning to improve the quality of the population. Next, a universal decoupling strategy is described. Finally, we introduce how to update the population with crossover and mutation operators based on the decoupling strategy to guide the search direction.

Overall Framework

The framework of the proposed method (T-NNGP) is illustrated in Figure 1. T-NNGP first uses traditional numerical methods to obtain the numerical solutions of the system of PDEs. Based on the physical datasets composed of numerical solutions, deep reinforcement learning generates an array of symbolic expression candidates for each unknown function. These candidate expressions form the neural-guided knowledge base (Step 1). Then T-NNGP sample the symbolic expression from the neural-guided knowledge base in order and form individuals based on the decoupling strategy. These individuals together construct the neural-guided population (Step 2). Next, T-NNGP substitutes them into the system of PDEs to evaluate the performance of the searched individuals. The evaluation results are then used to update the decoupling strategy (Step 3). Finally, crossover and mutation operations are applied based on the decoupling strategy to generate new individuals (Step 4). These new individuals are optimized and evaluated, and the top-performing individuals from this generation are selected to enter the next generation. The whole process of T-NNGP is shown in Algorithm 1.

Algorithm 1: T-NNGP

Input: $pdes$ (A system of PDEs to be solved), ${c_u}^0$ (the degree of constraint on each unknown function), f (Fitness function)
Output: $\mathcal{T}^*$ (Optimal symbolic expression solutions)
$\{\mathbf{X}_j, \mathbf{u}_j\}_{j=1}^k \leftarrow$ FEM($pdes$)
$\mathcal{T}_{\mathrm{RNN}} \leftarrow$ RNN($\{\mathbf{X}_j, \mathbf{u}_j\}_{j=1}^k$)
$\mathcal{T}_{\mathrm{GP}}^0 \leftarrow$ null, $t \leftarrow 0$
repeat
  $prob_u^t \leftarrow$ ProbabilityMapping($c_u{}^t$)
  $\mathcal{T}_{\mathrm{GP}}^t \leftarrow \mathcal{T}_{\mathrm{GP}}^t \cup$ Sample from $\mathcal{T}_{\mathrm{RNN}}$ based on $prob_u^t$
  $offspring \leftarrow$ Apply crossover and mutation based decoupling order to $\mathcal{T}_{\mathrm{GP}}^t$
  Optimize the constant of new individuals generated by mutation.
  Evaluate offspring by f
  Update ${c_u}^{t+1}$ as Eq. 5
  $\mathcal{T}_{\mathrm{GP}}^{t+1} \leftarrow$ Select top individuals from $\mathcal{T}_{\mathrm{GP}}^t \cup$ offspring
until stop condition is satisfied

Neural-Guided Individual

In a system of PDEs, there are multiple unknown functions. Relying solely on genetic programming to search for numerous expressions can be highly challenging. In this study, we employ deep reinforcement learning to generate candidate expressions based on physical datasets produced by traditional numerical methods. These expressions are then used to construct individuals that can represent solutions to systems of PDEs.

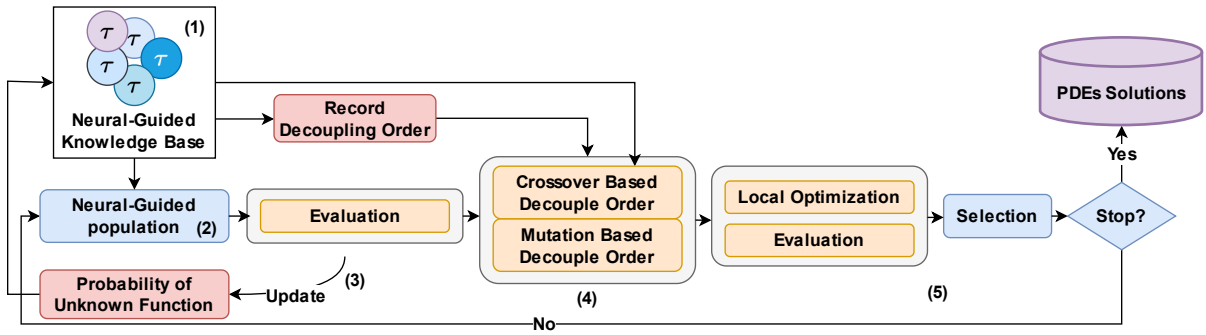

Figure 1: The pipeline of the proposed T-NNGP: Step 1. Based on the physical datasets composed of numerical solutions, deep reinforcement learning generates the neural-guided knowledge base. Step 2. Construct the neural-guided population by sampling the symbolic expression from the neural-guided knowledge base in order and form individuals based on the decoupling strategy. Step 3. Evaluate the performance of the searched individuals. Meanwhile, update the decoupling strategy. Step 4. Apply crossover and mutation based on decoupling to generate new individuals. Step 5. Selection: Select the top individuals to enter the next generation.

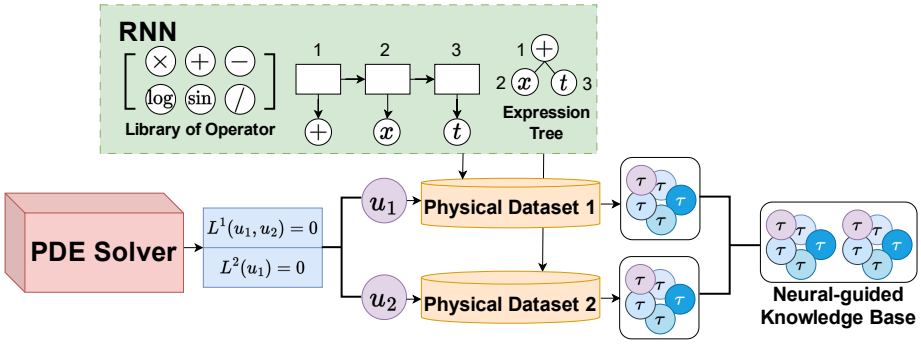

Figure 2: The process of generating a neural-guided knowledge base: The numerical solution of the partial solution domain is obtained by using the traditional numerical method (PDE Solver). These numerical solutions are used to compose multiple datasets describing unknown functions in the system of PDEs. The neural network (e.g. RNN) generates many candidate solutions that can fit the datasets. The knowledge base composed of these candidate solutions is called the neural-guided knowledge base.

Deep reinforcement learning can leverage the powerful representation capabilities of deep learning to discover mathematical expressions that best fit a given dataset $(\mathbf{X}, \mathbf{y})$ (Petersen et al. 2019). Appendix B provides detailed instructions on how to train it to discover expressions. This method works by using a recursive neural network (RNN) to autoregressively sample a symbol sequence, which is a preorder traversal of the symbolic expression tree. The previous symbol and dataset are input into the RNN, which predicts the probability distribution of the next symbol. By repeating these steps, a large number of expressions can be generated. After generating the expression, we use gradient descent to fine-tune the numerical coefficients based on the dataset. The sampling process is depicted in Figure 2.

We use the reward function $R(\tau, (\mathbf{X}, \mathbf{y})) = 1/(1 + \mathrm{NRMSE})$ to evaluate the quality of a generated mathematical expression f, which can be represented by a sequence τ. Where $\mathrm{NRMSE} = \frac{1}{\sigma_y}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f\left(X_i\right)\right)^2}$. Since the reward function is not differentiable for the parameters θ of RNN, traditional gradient-based optimization methods cannot be applied. Instead, reinforcement learning is employed, where the RNN acts as a policy that outputs actions (e.g., sampling a symbol) based on the current state (e.g., parent and sibling inputs).

The goal is to maximize the expectation of reward under the policy $p(\tau \mid \theta)$ generated by RNN, which is defined as Eq. 1.

$$J(\theta) := \mathbb{E}_{\tau \sim p(\tau|\theta)}[R(\tau, (\mathbf{X}, \mathbf{y}))] \tag{1}$$

By taking the derivative of the Eq. 1 to the policy parameters θ, Eq. 2 is obtained, which can then be used to maximize the expectation of reward.

$$\begin{aligned} \nabla_\theta J_(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim p(\tau|\theta)}[R(\tau, (\mathbf{X}, \mathbf{y}))] \\ &= \mathbb{E}_{\tau \sim p(\tau|\theta)}\left[R(\tau, (\mathbf{X}, \mathbf{y}))\nabla_\theta \log p(\tau \mid \theta)\right] \\ &\approx \frac{1}{N}\sum_{i=1}^{N} R\left(\tau^{(i)}, (\mathbf{X}, \mathbf{y})\right)\nabla_\theta \log p\left(\tau^{(i)} \mid \theta\right) \end{aligned} \tag{2}$$

Each unknown function corresponds to a physical dataset. Based on each physical dataset, the deep reinforcement learning method can generate many candidate expressions

corresponding to each unknown function. These candidate expressions collectively form a neural-guided knowledge base.

Decoupling Strategy

Due to the errors and incompleteness of numerical results, each unknown function corresponds to many candidate expressions. Although these expressions have different structures, they can all fit the given dataset well. Suppose there are m unknown functions, and each unknown function has q candidate expressions. To find the expressions that best satisfy the system of PDEs, we need to evaluate q^m times. Therefore, we propose a decoupling strategy, which gives the order of solving for each unknown function and the number of samples in the neural-guided knowledge base.

Suppose we have a system of PDEs as shown in Eq. 3.

$$\begin{aligned} L_1\left(u_1, u_2\right) &= 0 \\ L_2\left(u_1\right) &= 0 \end{aligned} \tag{3}$$

There are two unknown functions u_1 and u_2. According to Eq. 3, the degree of constraint (denoted as c_u) for u_1 and u_2 can be calculated, as shown in Eq. 4:

$$c_u = N_{eq} + N_{var}, \tag{4}$$

where N_{eq} represents that the unknown function has appeared in N_{eq} equations, and N_{var} represents that the unknown function has appeared total N_{var} times in the equations. For u_1, the value of N_{eq} takes 2, the value of N_{var} takes 2. For (u_1, u_2), the value of (c_{u_1}, c_{u_2}) takes (4,2).

The c_u are converted to the probability $prob_u$ by negating them and normalizing the results (see details in Algorithm 2 of Appendix A). Consequently, higher c_u values yield lower probabilities $prob_u$. The larger $prob_u$ for an unknown function u is, the more likely it is to be solved in the first order. This is because when dealing with problems involving multiple unknowns, it is often beneficial to initially solve for the unknowns that are subject to fewer constraints. By doing so, we can gather the necessary information for subsequent steps, ultimately leading to a more efficient solution for the entire problem.

The individuals in the neural-guided population are generated based on probability $prob_u$. Figure 3 shows process of generating two individuals from the neural-guided knowledge base. Assume that there are 5 candidate expressions for u_1 and u_2 respectively in the neural-guided knowledge base. Suppose the order of solving (u_2, u_1) is generated based on probability $prob_u$. The order of solving the unknown function in the later means that it is subject to more constraints than the unknown function solved in the previous. Thus the unknown function solved in the later is more difficult to solve. Therefore, we set the number of expressions sampled later to be twice the number of the previous sample. u_2 samples only one expression, u_1 samples two expression. Combining u_1 and u_2 can generate an individual. Repeatedly generate a sampling order based on $prob_u$ until the number of individuals reaches the specified population size.

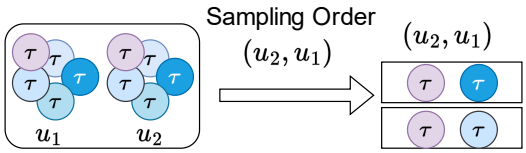

Figure 3: Decoupling Strategy

The $prob_u$ is calculated from c_u. The c_u is dynamically updates according to Eq. 5:

$$c_{u_j} = \alpha \times c_{u_j} + \sum_{i=1}^{num} \frac{1}{order_{ij}} \times \mathcal{E}_{PDE_i}, \tag{5}$$

where c_{u_j} represents the c_u value of the unknown function u_j, j=1, . . . , m. The $order_{ij}$ represents the sampling order of u_j in individual i. The num represents the population size. The $\mathcal{E}_{PDE_i}$ represents a fitness function to evaluate whether the searched function expressions $\mathbf{u}$ ($\mathbf{u}$ = (u_1, u_2)) can be used as a solution to the PDEs. It can be defined as Eq. 6:

$$\mathcal{E}_{PDE} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i(\mathbf{u}(\mathbf{X}))^2. \tag{6}$$

Eq. 5 is designed with the following considerations. As illustrated in Figure 3, during the process of generating individuals (solutions) for the system of PDEs, expressions sampled earlier tend to occur more frequently in the population compared to those sampled later. Thus expressions sampled earlier play a more significant role in determining the quality of the overall solution. If the current individual (solution) fails to satisfy the system of PDEs, the value of c_u for the unknown function sampled earlier is increased. This adjustment has the effect of reducing the priority sampling probability of the unknown function sampled earlier.

Update Mechanism

This section proposes crossover and mutation operators based on decoupling order to generate new individuals. An individual in the evolution of the T-NNGP contains multiple expressions representing unknown functions. When crossover or mutation is applied to all expressions within an individual simultaneously, it can become difficult to precisely preserve the performing well expressions while enhancing those performing bad expressions. By leveraging the decoupling order, T-NNGP selectively modifies only a few expressions within each individual, allowing for a more focused and precise evolution of the solutions.

The crossover operator works by selecting the first sampled expression in an individual. It then replaces this selected expression with a corresponding expression from the neural-guided knowledge base. The mutation operator works by randomly selecting a subtree within the last sampled expression in an individual. After that, it randomly generates a new subtree and replaces the selected subtree with the newly generated one.

4 Experiments

In this section, experiments on three types of systems of PDEs are conducted to evaluate the performance of the proposed T-NNGP. Firstly, we present the symbolic expression solution by T-NNGP. Next, we compared T-NNGP with PhySO (Tenachi, Ibata, and Diakogiannis 2023) (deep reinforcement learning for generating neural-guided knowledge base) by the system of PDEs residual errors resulting from substituting the obtained expressions into the system of PDEs. Then, we evaluate the performance of the proposed T-NNGP compared with DeepM&Mnet (Cai et al. 2021b) and PhySO (Tenachi, Ibata, and Diakogiannis 2023) in fitting physical datasets. In addition, we plot the average fitness of the population with and without the decoupling strategy to verify the effectiveness of the decoupling strategy.

Experiment Settings

We tested the proposed method on three types of systems of PDEs. The details of the test problems are presented below (see more details in Appendix C). Table 3 in Appendix D lists the main parameters setting of T-NNGP.

Diffusion-convection. Eq. 7 is a system of PDEs that describe the processes of diffusion and convection in fluid dynamics.

$$\begin{aligned} -\nabla \cdot \nabla u + \nabla v \cdot \nabla u = 0 \\ -\nabla \cdot \nabla v = 0 \end{aligned} \tag{7}$$

Joule heating. Eq. 8 is a system of PDEs that describes the propagation of current and heat in a conductor.

$$\begin{aligned} -\nabla \cdot (\sigma \nabla V) = 0 \\ \rho C \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = \sigma |\nabla V|^2 \end{aligned} \tag{8}$$

Parabolic. Eq. 9 is a system of parabolic PDEs that describe the change of unknown functions over time and space.

$$\begin{aligned} \frac{\partial u_1}{\partial t} = 0.024 \frac{\partial^2 u_1}{\partial x^2} - F(u_1 - u_2) \\ \frac{\partial u_2}{\partial t} = 0.170 \frac{\partial^2 u_2}{\partial x^2} + F(u_1 - u_2) \end{aligned} \tag{9}$$

In the systems of PDEs mentioned above, there are variations in the notations used to denote the unknown functions. For clearer and more consistent description, Table 1 redefines the notations for these unknown functions.

Name	u_1	u_2
Diffusion-convection	v	u
Joule heating	V	T
Parabolic	u_1	u_2

Table 1: Redefined of unknown function symbol in systems of PDEs

In this paper, the physical dataset is generated by solving the systems of PDEs utilising the finite element method (FEM) (Permann et al. 2020; Li and Chen 2019). We select PhySO (Tenachi, Ibata, and Diakogiannis 2023) as a deep reinforcement learning algorithm to generate the neural-guided knowledge base within the framework of T-NNGP.

Name	T-NNGP	PhySO	IR (%)
Diffusion-convection	**4.02e-03**	1.60e-02	74.875
Joule heating 1	**2.36e00**	1.11e15	99.999
Joule heating 2	**1.10e-13**	2.85e14	100.0
Joule heating 3	**1.48e-14**	4.04e08	100.0
Joule heating 4	**1.78e-04**	3.59e06	99.999
Joule heating 5	**2.03e-08**	1.19e06	99.999
Parabolic 1	**1.93e00**	3.35e02	99.423
Parabolic 2	**2.82e-01**	4.51e02	99.937
Parabolic 3	**6.70e-02**	5.38e-01	87.546

Table 2: The MSE of the expressions fitting error to the system of PDEs

Partial Differential Equations Fitting Comparison

In this section, we compare the performance of the proposed T-NNGP and PhySO (Tenachi, Ibata, and Diakogiannis 2023) for solving the system of PDEs. For PhySO, we select the expressions generated by PhySO with the best ability to fit the dataset as the solutions of the PDEs. It is worth noting that we only compare PhySO and T-NNGP because only the output of PhySO and T-NNGP are expressions that can be taken into a system of PDEs to verify the accuracy of the solution.

Table 2 displays the errors of solutions obtained by T-NNGP and PhySO after these solutions are substituted into the system of PDEs. The specific calculation process is shown in Eq. 6. The ability of T-NNGP to fit PDEs is superior to PhySO in all instances. The fourth column represents the improvement rate (IR) of T-NNGP over PhySO, which is calculated as specified in Eq. 10.

$$IR = \frac{MSE_{PhySO} - MSE_{T-NNGP}}{MSE_{PhySO}} \times 100\% \tag{10}$$

In five instances, we can even consider the expressions generated by T-NNGP to be a perfect match for the given system of PDEs. The expressions produced by PhySO match the system of PDEs in only two instances. In other instances, solutions generated by PhySO caused huge errors. It can be inferred from Table 2 that the solution obtained by T-NNGP is much more likely to be the solution of a system of PDEs than the solution obtained by PhySO.

Tables 3 present the expression solutions of 3 types systems of PDEs provided by T-NNGP.

Numerical Data Fitting Comparison

In this section, we compared the performance of the proposed T-NNGP with the DeepM&Mnet (Cai et al. 2021b), and PhySO (Tenachi, Ibata, and Diakogiannis 2023) to fit the physical dataset. DeepM&Mnet is the only deep learning method that attempts to solve the system of PDEs. Table 4 presents the mean absolute error (MAE) and the variance of the error between the predicted values using the above algorithms and the values in the physical dataset.

For the symbolic regression method (PhySO and T-NNGP), 100 data points are randomly selected to generate expression. For DeepM&Mnet, 10% of the dataset is taken

Name	u_1	u_2
Diffusion-convection	$0.421 - 0.220z$	$0.000451xz - 0.407z + 1.068$
Joule heating 1	$1.00 - 1000x$	$0.0521sin(2xz - 2.00z) + 460$
Joule heating 2	$-1000x + z + 1.00$	$0.969ysin(z) + 382$
Joule heating 3	$-1000x - 0.255z + 1.00$	$yzsin(t) + 380$
Joule heating 4	$1.00 - 1000x$	$0.0184sin(0.569t + 136) + 384$
Joule heating 5	$1.00 - 1000x$	$yz^2 - z^3 + 368.659$
Parabolic 1	$sin(sin(x))$	$0.997sin(x)$
Parabolic 2	$0.984sin(x + 0.0438)$	$0.995sin(x)$
Parabolic 3	$0.954 - 0.0326x$	$0.728sin(0.954tx + 0.582) + 0.232$

Table 3: Optimal solution of T-NNGP

Name	T-NNGP		DeepM&Mnet		PhySO	
	u_1	u_2	u_1	u_2	u_1	u_2
instance 1	8.07e-02±7.89e-03	1.44e-02±6.26e-04	1.84e-02± 8.29e-04	1.52e-02± 6.27e-04	**1.84e-02±8.16e-04**	**1.40e-02±6.41e-04**
instance 2.1	2.65e-05±4.63e-09	3.16e01±2.70e03	2.68e-01± 9.94e-02	**2.94e00± 9.63e00**	**2.05e-05±4.62e-09**	4.86e00±4.05e01
instance 2.2	2.71e-05±4.70e-09	1.49e01±6.75e02	2.61e-01± 9.23e-02	**2.12e-01± 2.31e-02**	**1.87e-05±4.56e-09**	2.38e00±9.21e00
instance 2.3	1.97e-05±4.57e-09	1.59e01±6.41e02	2.70e-01± 9.51e-02	**2.52e00± 2.41e00**	**1.81e-05±4.55e-09**	6.24e00±6.45e01
instance 2.4	3.77e-05±5.24e-09	1.19e01±4.40e02	2.58e-01± 8.79e-02	**3.26e-01± 1.28e-02**	**2.00e-05±4.58e-09**	6.19e00±4.72e01
instance 2.5	**1.60e-05±4.55e-09**	2.41e01±4.40e02	2.58e-01± 8.80e-02	**3.42e00± 4.90e00**	**1.60e-05±4.55e-09**	7.23e00±9.77e01
instance 3.1	2.27e-01±6.74e-02	1.66e-01±5.63e-02	**1.15e-01± 2.25e-02**	**6.37e-02± 1.13e-02**	1.16e-01 ± 2.31e-02	8.30e-02±1.41e-02
instance 3.2	2.14e-01±5.54e-02	1.53e-01±5.86e-02	**8.87e-02± 1.57e-02**	**6.33e-02± 8.65e-03**	9.13e-02±1.53e-02	8.75e-02±1.35e-02
instance 3.3	6.48e-02±5.15e-03	5.37e-01±2.53e-02	9.56e-02± 1.99e-02	3.68e-02± 3.39e-03	**2.19e-02±9.05e-04**	**1.40e-02±4.82e-04**

Table 4: Summary the MAE and variance of the model fitting error to the numerical dataset

as training data. Compared to the entire dataset, very little data is used for training, so we use all the data in the dataset for evaluation.

T-NNGP achieved 1 of the best results. T-NNGP achieved the same order of magnitude accuracy as the optimal value in 8 results. There may be two reasons why the performance of T-NNGP in MAE is not very good. One reason is that T-NNGP needs to consider the constraints introduced by the PDEs and the physical dataset. Another reason is that the numerical values in the physical dataset are approximate solutions to the system of PDEs within the solution domain, rather than exact true values. Therefore, MAE cannot fully measure the accuracy of methods in solving the system of PDEs. Based on the comprehensive experimental results from Section **Partial Differential Equations Fitting Comparison** and this section, the symbolic expressions generated by T-NNGP are the most accurate in most instances.

Ablation Studies

To verify the effectiveness of the proposed decoupling strategy and update mechanism, we track the fitness of the population with and without using the proposed strategy (mechanism). Figure 4 shows the convergence traces over 100 generations on the Diffusion-convection. The blue line represents T-NNGP without the decoupling strategy. The yellow line represents T-NNGP without the update mechanism. The red line represents the proposed T-NNGP. It shows that T-NNGP using a decoupling strategy and update mechanism has lower mean fitness than without. A more complete presentation of these experiments can be found in Appendix E.

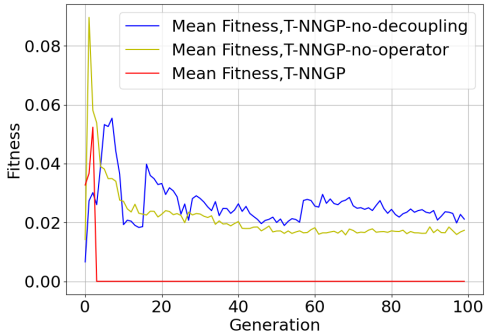

Figure 4: Convergence traces of mean fitness values obtained by T-NNGP on the Diffusion-convection.

5 Conclusion

Multiphysics simulations are essential for solving complex engineering and scientific problems. In this paper, we introduce T-NNGP, a hybrid neural-guided genetic programming approach aimed at solving systems of PDEs across multiple physical fields. To our knowledge, this is the first attempt at applying symbolic regression in multiphysics simulations. Our experimental results on three distinct types of PDEs indicate that T-NNGP can produce symbolic solutions that are both accurate and interpretable. In the future, we plan to use real-world observational datasets instead of traditional numerical physics datasets. We believe this methodology could serve as a foundation for future research and refinement in the multiphysics simulations area.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62276222 and the Public Technology Service Platform Project of Xiamen City, Grant No.3502Z20231043. This work was also supported by the Research Grants Council of the Hong Kong SAR (Grant No. PolyU11211521, PolyU15218622, PolyU15215623, and C5052-23G), and the National Natural Science Foundation of China (Grant No. U21A20512).

References

Adhywinaya, I. N.; and Budiman, B. A. 2022. Structural Design and Analysis of Electric Car Engine Mount. *International Journal of Sustainable Transportation Technology*, 5(1): 32–37.

Ali, A. H.; Jaber, A. S.; Yaseen, M. T.; Rasheed, M.; Bazighifan, O.; and Nofal, T. A. 2022. A comparison of finite difference and finite volume methods with numerical simulations: Burgers equation model. *Complexity*, 2022.

Amaratunga, K.; and Williams, J. R. 1993. Wavelet based Green's function approach to 2D PDEs. *Engineering Computations*, 10(4): 349–367.

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

Burns, K. J.; Vasil, G. M.; Oishi, J. S.; Lecoanet, D.; and Brown, B. P. 2020. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2(2): 023068.

Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; and Karniadakis, G. E. 2021a. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12): 1727–1738.

Cai, S.; Wang, Z.; Lu, L.; Zaki, T. A.; and Karniadakis, G. E. 2021b. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436: 110296.

Cao, L.; Hong, H.; and Jiang, M. 2024. Fast Solving Partial Differential Equations via Imitative Fourier Neural Operator. In *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Cao, L.; Liu, Y.; Wang, Z.; Xu, D.; Ye, K.; Tan, K. C.; and Jiang, M. 2024. An Interpretable Approach to the Solutions of High-Dimensional Partial Differential Equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20640–20648.

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.

David Müzel, S.; Bonhin, E. P.; Guimarães, N. M.; and Guidi, E. S. 2020. Application of the finite element method in the analysis of composite materials: A review. *Polymers*, 12(4): 818.

Gao, H.; Zahr, M. J.; and Wang, J.-X. 2022. Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390: 114502.

Gong, Y.; Hou, Y.; Wang, Z.; Lin, Z.; and Jiang, M. 2024. Adversarial Learning for Neural PDE Solvers with Sparse Data. *arXiv preprint arXiv:2409.02431*.

Haider, J. A.; and Ahmad, S. 2022. Dynamics of the Rabinowitsch fluid in a reduced form of elliptic duct using finite volume method. *International Journal of Modern Physics B*, 36(30): 2250217.

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.

Li, J.; and Chen, Y.-T. 2019. *Computational partial differential equations using MATLAB®*. Crc Press.

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

Lu, L.; Jin, P.; Pang, G.; Zhang, Z.; and Karniadakis, G. E. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3): 218–229.

Mao, Z.; Jagtap, A. D.; and Karniadakis, G. E. 2020. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360: 112789.

Morrison, F. A. 2013. *An introduction to fluid mechanics*. Cambridge University Press.

Muhammad, N. 2021. Finite volume method for simulation of flowing fluid via OpenFOAM. *The European Physical Journal Plus*, 136(10): 1–22.

Mundhenk, T.; Landajuela, M.; Glatt, R.; Santiago, C. P.; Petersen, B. K.; et al. 2021. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. *Advances in Neural Information Processing Systems*, 34: 24912–24923.

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. 2023. Genetic Programming Based Symbolic Regression for Analytical Solutions to Differential Equations. *arXiv preprint arXiv:2302.03175*.

Permann, C. J.; Gaston, D. R.; Andrš, D.; Carlsen, R. W.; Kong, F.; Lindsay, A. D.; Miller, J. M.; Peterson, J. W.; Slaughter, A. E.; Stogner, R. H.; et al. 2020. MOOSE: Enabling massively parallel multiphysics simulation. *SoftwareX*, 11: 100430.

Petersen, B. K.; Landajuela, M.; Mundhenk, T. N.; Santiago, C. P.; Kim, S.; and Kim, J. T. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Petersen, B. K.; Landajuela, M.; Mundhenk, T. N.; Santiago, C. P.; Kim, S. K.; and Kim, J. T. 2019. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *arXiv preprint arXiv:1912.04871*.

Phan, H.; and He, L. 2022. Efficient modeling of mistuned blade aeroelasticity using fully-coupled two-scale method. *Journal of Fluids and Structures*, 115: 103777.

Polyanin, A. D.; and Zhurov, A. I. 2021. *Separation of variables and exact solutions to nonlinear PDEs*. CRC Press.

Polycarpou, A. C. 2022. *Introduction to the finite element method in electromagnetics*. Springer Nature.

Pryor, R. W. 2009. *Multiphysics modeling using COMSOL®: a first principles approach*. Jones & Bartlett Publishers.

Robertsson, J. O.; and Blanch, J. O. 2020. Numerical methods, finite difference. *Encyclopedia of solid earth geophysics*, 1–9.

Struchtrup, H. 2014. *Thermodynamics and energy conversion*. Springer.

Szabó, B.; and Babuška, I. 2021. Finite Element Analysis: Method, Verification and Validation.

Takamoto, M.; Praditia, T.; Leiteritz, R.; MacKinlay, D.; Alesiani, F.; Pflüger, D.; and Niepert, M. 2022. PDEBench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611.

Tenachi, W.; Ibata, R.; and Diakogiannis, F. I. 2023. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *arXiv e-prints*, arXiv:2303.03192.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16): eaay2631.

Ullah, H.; Hayat, T.; Ahmad, S.; and Alhodaly, M. S. 2021. Entropy generation and heat transfer analysis in power-law fluid flow: Finite difference method. *International Communications in Heat and Mass Transfer*, 122: 105111.

Wahyudi, S.; Lestari, P.; and Gapsari, F. 2021. Application of Finite Difference Methods (FDM) on mathematical model of bioheat transfer of one-dimensional in human skin exposed environment condition. *J Mech Eng Res Develop*, 44(5): 1–9.

Wang, K.; Zhang, Z.-D.; Li, M.-J.; and Min, C.-H. 2021. A coupled optical-thermal-fluid-mechanical analysis of parabolic trough solar receivers using supercritical CO2 as heat transfer fluid. *Applied Thermal Engineering*, 183: 116154.

Wang, Y.; and Zhong, L. 2024. NAS-PINN: neural architecture search-guided physics-informed neural network for solving PDEs. *Journal of Computational Physics*, 496: 112603.

Yu, J.; Lu, L.; Meng, X.; and Karniadakis, G. E. 2022. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393: 114823.

Zhang, D.; Zhang, L.; Huiying, T.; and Yulong, Z. 2022. Fully coupled fluid-solid productivity numerical simulation of multistage fractured horizontal well in tight oil reservoirs. *Petroleum Exploration and Development*, 49(2): 382–393.

Zimmerman, W. B. 2006. *Multiphysics modeling with finite element methods*, volume 18. World Scientific Publishing Company.