

Received 13 October 2023, accepted 29 October 2023, date of publication 8 November 2023, date of current version 28 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3331330

THEORY

A Solution Method for Differential Equations Based on Taylor PINN

YAJUAN ZHANG¹, MIN WANG², FANGWEI ZHANG³, (Senior Member, IEEE), AND ZHENRUI CHEN², (Member, IEEE)

¹College of Information Engineering, Hainan Vocational University of Science and Technology, Haikou, Hainan 571126, China

²Department of Basic Education, Shandong Jiaotong University, Weihai, Shandong 264209, China

³School of Maritime, Shandong Jiaotong University, Weihai, Shandong 264209, China

Corresponding author: Min Wang (wangmintougao2022@163.com)

This work was supported in part by the Hainan Provincial Natural Science Foundation of China under Project 621RC611; in part by the Project: Provincial High-Level Professional Group "Computer Network Technology," Hainan Province; and in part by the Project: The Fifth Round of Provincial Characteristic Key Discipline "Computer Science and Technology."

ABSTRACT Based on deep neural network, elliptic partial differential equations in complex regions are solved. Accurate and effective strategies and numerical methods for elliptic partial differential equations are proposed by implementing deep feedforward artificial neural network, appropriate loss function solving strategy are constructed. The solution of an elliptic partial differential equation is obtained by iteratively learning the parameters of a neural network. Constructing a composite multi-layer radial basis function neural network can improve the real function approximation performance and operational accuracy of the constructed multi-layer radial basis function neural network. Use this high-precision composite multi-layer radial basis function neural network to solve partial differential equations. By providing specific examples of solving partial differential equations, the effectiveness of this method is tested. An improved partial differential equation solving method based on deep neural networks (Taylor PINN) has been proposed. This method utilizes the universal approximation theorem of deep neural networks and the function fitting ability of Taylor's formula to achieve a meshless numerical solution process. The numerical experimental results on Helmholtz, Klein Gordon, and Navier Stokes equations show that Taylor PINN can well fit the mapping relationship between the coordinates of spatiotemporal points in the computational domain and the value of the desired function, which can provide accurate numerical prediction results. Compared with commonly used physical information based neural network methods, Taylor PINN improves prediction accuracy by 3-20 times for different numerical problems.


INDEX TERMS Partial differential equations, numerical analysis, neural network, Taylor's formula, loss function.

I. INTRODUCTION

With the improvement of computing power, numerical analysis is playing an increasingly important role in many scientific and engineering fields [1]. The underlying mathematical models of most numerical analysis problems can be attributed to the numerical solution of partial differential equations.

Traditional numerical methods mainly include finite difference method, finite element method and finite volume method [2]. These methods discretize the computational

domain into independent grid elements to obtain numerical approximations of unknown functions. However, in order to ensure the accuracy and effectiveness of numerical analysis, traditional methods are usually very time-consuming and heavily rely on manual experience. On the one hand, iteratively solving large linear/nonlinear equations requires expensive computational overhead; on the other hand, in order to avoid computational failure, frequent human-machine interaction is usually required during the mesh generation stage to distinguish and optimize the quality of the computational mesh, in order to meet the requirements of solver and simulation accuracy [3], [4]. As the problems

The associate editor coordinating the review of this manuscript and approving it for publication was Valentina E. Balas .

analyzed become more and more complex, heavy interaction and overhead limit the application efficiency of traditional numerical methods in real-time simulation, aerodynamic parameter design, optimization of space exploration and other issues.

The past few decades have witnessed a revolution in deep learning, with the development and evolution of artificial neural networks (ANNs) being one of the key elements. Machine learning methods based on ANNs can solve some very complex application problems in fields such as image processing, speech recognition, and medical diagnosis. ANNs have also been widely used in the field of mathematics because they can effectively approximate arbitrary functions [5]. Although ANNs have achieved impressive results in several important application fields, there are still many questions about why and how they work effectively [6]. Research shows that with the increase of network depth, the neural network has a stronger ability to approach functions [7]. Therefore, the application of deep neural network (DNN) to solving partial differential equations has received extensive attention from many researchers [8]. Berner et al. [9] studied a new algorithm for solving high-dimensional parabolic partial differential equations and backward stochastic differential equation. Gao et al. [10] proposed a deep Ritz method based on deep learning to solve partial differential equation problems through variational methods. Kidger et al. [11] proposed a method for solving non-homogeneous partial differential equations based on deep neural networks, parameterizing the solved physical quantities and searching for solutions to partial differential equations through neural network execution time evolution, thereby proving the rationality of using neural networks to characterize this quantity. Huang et al. [12] proposed an invasive approach for high-dimensional stochastic partial differential equations Completely unsupervised and networkless deep learning based solutions. Kien et al. [13] proposed a physical network to optimize variational problems based on training processes. Berg [14] and Zhang [15] et al. used Gaussian processes to fit linear operators on the basis of machine learning methods and further extended this method to the regression of non-linear operators. Chen et al. [16] used PINN based methods to solve partial differential equations in complex domains and achieved some good results. Although PINN have many advantages in solving partial differential equation problems, they still have some problems. Firstly, without a theoretical foundation to understand the scale of neural network structures and the required amount of data, it is impossible to guarantee that algorithms will not converge to local minima, and their training time is slower than traditional numerical methods. Chakraborty et al. [17] developed the Physical Information Extreme Learning Machine (PIELM), which can be applied to time-dependent linear partial differential equation problems. They also demonstrated that PIELM outperforms PINNs in a series of problems. Piscopo et al. [18] proposed a method for solving partial differential equations using artificial neural networks and adaptive configuration strategies.

Darbon et al. [19] proposed a neural network based partial differential equation solving model suitable for irregular boundaries. This model uses multi-layer perceptron and radial basis function neural networks to fit complex boundaries, and has achieved high prediction accuracy on some relatively simple two-dimensional and three-dimensional partial differential equations. Rodriguez-Torrado et al. [20] proposed a partial differential equation solving method based on Long Short Term Memory (LSTM) networks for solving high-dimensional partial differential equations. However, the above methods are only applicable to specific types of partial differential equations and lack certain universality. Dwivedi et al. [21] proposed Physical Information Neural Networks (PINN) for approximating the functions to be solved in partial differential equations.

Traditional physical information neural networks have low prediction accuracy in solving many physical problems based on partial differential equations. A high-precision solution method for partial differential equations based on deep learning networks is proposed. By constructing a solution that connects the output of the neural network with the required function, a meshless numerical solution process is achieved. The numerical experimental results of Klein Gordon equation [22], Helmholtz equation [23], and Navier Stokes equation [24] show that Taylor PINN can accurately fit the mapping relationship between spatiotemporal point coordinates and expected functions in the computational domain, and provide accurate numerical prediction results. Compared with commonly used neural network methods based on physical information, Taylor PINN has improved the prediction accuracy of different numerical problems by approximately 12 times.

II. METHOD AND PRINCIPLE

A. NEURAL NETWORKS AND THEIR STRUCTURES

First, a single hidden layer neural network is defined. Given the d -dimensional row vector x as the input of the model, the k -dimensional output of the neural network is in the following form:

$$y = \sigma(xw_1 + b_1)w_2 + b_2 \quad (1)$$

In the formula, w_1 and w_2 are sizes $d \times Q$; b_1 and b_2 are bias; $\sigma(\cdot)$ is the activation function. In the deep neural network, each hidden layer linearly processes the input variables through the weight matrix and deviation [25], then the activation function is used for nonlinear transformation. The nonlinear output of the previous hidden layer continues to be the input of the next hidden layer and repeats the linear processing of the weight matrix and the nonlinear transformation of the activation function again. The new weight matrix and deviation in each hidden layer constitute a new hidden layer in the neural network [26]. The ability of neural networks to approximate complex nonlinear functions can be improved by adding more hidden layers or increasing the dimension of hidden layers. Neural networks with $k - 1$ hidden layers can be expressed as:

$$N_\theta(z) = T_k \circ \sigma \circ T_{k-1} \circ \dots \circ T_1 \circ \sigma \circ T_0(z) \quad (2)$$

where: $T(z_{k-1}) = w_k z_{k-1} + b_k$, w_k represents the weight matrix between $k-1$ to the $(k-1)$ -th layer, z_{k-1} and b_k represent the output and deviation vectors of the hidden layer $k-1$. $N_\theta(z)$ is the output of the neural network, $z_0 = z$ represents an input parameter $\sigma(\cdot)$ is a nonlinear activation function. At present, the most popular activation function includes Sigmoid, Tanh and ReLU. ReLU activation function is one of the most widely used functions, in the form of $f(z) = \max(0, z)$. However, the higher derivative of ReLU is 0, which limits the applicability of this paper to deal with differential equations composed of higher derivatives. Tanh or Sigmoid activation function can be used for second-order or higher-order partial differential equations. The output of the sigmoid function is relatively flat [27], and when the input changes little, the output changes little, which may not be flexible enough for linear models. The Tanh activation function is antisymmetric [28], and by allowing the output of each neuron to take values on the interval $[-1, 1]$. The system bias problem caused by sigmoid activation is overcome. In addition, compared with the training of asymmetric activated deep neural networks [29]. The training process of anti-symmetric activated deep neural network converges faster. In the regression problem of given multiple training data points, this paper can use Euclidean loss function to calibrate the weight matrix and bias, as shown below:

$$J(\theta; x, y) = \frac{1}{M} \sum_{i=1}^M \|y_i - \hat{y}_i\|^2 \quad (3)$$

In the formula, J represents the mean square error; $x = \{x_1, x_2, \dots, x_M\}$ is the input vector; $y = \{y_1, y_2, \dots, y_M\}$ is the output vector; $\{y_1, y_2, \dots, y_3\}$ are the predicted outputs of the neural network. The model parameters can be solved according to equation (4):

$$(\omega_1^*, \omega_2^*, \dots, b_1^*, b_2^*, \dots) = \underset{(\omega_1, \dots, b_1, \dots)}{\text{arg min}} J(\theta; x, y) \quad (4)$$

Use gradient descent method to optimize the model, specifically, in the i -th iteration, the model parameters $\theta = \{\omega_1, \omega_2, \dots, b_1, b_2, \dots\}$ are updated through equation (5):

$$\theta^{(i+1)} = \theta^{(i)} - \eta^{(i)} \nabla_{\theta} J^{(i)}(\theta^{(i)}; x, y) \quad (5)$$

Among:

$$w_k^{(i+1)} = w_k^{(i)} - \eta^{(i)} \frac{\partial}{\partial w_k^{(i)}} J(\theta; x, y) \quad (6)$$

$$b_k^{(i+1)} = b_k^{(i)} - \eta^{(i)} \frac{\partial}{\partial b_k^{(i)}} J(\theta; x, y) \quad (7)$$

$\eta(i)$ is the step size of the i -th iteration.

In the backpropagation process, the chain rule is used to first calculate the gradient of the last layer, and finally the gradient of the first layer. Some gradients of the current layer are reused in the previous gradient calculation. This reverse flow of information promotes the effective calculation of each layer's gradient in deep neural networks. Gradient descent iterative algorithm with given learning rate η ; Iterative convergence criterion neural network parameters $\theta = 0$. The output of the neural network is y . The training set $\{x_1, \dots, x_M\}$ for collecting M samples did not meet the stop criterion while,

with the corresponding target being y_i ; Calculate objective function:

$$J(\theta; x, y) = \frac{1}{M} \sum_{i=1}^M \|y_i - \hat{y}_i\|^2.$$

B. SOLVING TAYLOR PINN FOR PARTIAL DIFFERENTIAL EQUATIONS BASED ON DEEP NEURAL NETWORKS

In order to further improve the prediction accuracy of Physics-Informed Neural Network, this paper proposes an improved partial differential equation solving method based on deep neural networks, Taylor PINN [30]. This method also transforms the problem of solving differential equations into an optimization problem of functions. However, unlike traditional Physics-Informed Neural Network, Taylor PINN incorporates a formal construction process of the function to be solved in the prediction process, which no longer directly uses regression models to output discrete predicted values [31]. However, connects the neural network output with the function to be solved by constructing a solution. In this process, how to construct the form of the solution becomes the key after the problem is transformed [32]. Taylor's formula is a widely used method for function approximation, and Theorem 1 is the n th order Taylor expansion theorem of a univariate function:

Theorem 1: Assume the function $f(x)$ is differentiable to order $n+1$ in the neighborhood of $x = x_0$, then for any point x located in this neighborhood:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \dots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + R_n(x) \quad (8)$$

Among, $R_n(x)$ is $(x - x_0)^n$ higher order infinitesimal of the remainder.

According to the above theorem [33], this article uses a polynomial function space based on Taylor's formula to construct the solution form, that is, let each function to be solved in the partial differential equation be as shown in equation (9):

$$u = \lambda_0 + \lambda_1(\lambda_2 x + \lambda_3 t) + \dots + \lambda_{k1}(\lambda_{k2} x + \lambda_{k3} t)^k \quad (9)$$

Compared with other function spaces (such as Fourier series or various radial basis functions), the polynomial space composed of the linear weighting of the coordinates (x, t) and their products has flexible scalability, and has relatively small computational cost. In terms of model architecture, Taylor PINN not only uses a fully connected layer, but also introduces an input enhancement layer as an enhancement method to expand the dimension of input point coordinates. This enhancement layer implements a discrete linear mapping based on spatiotemporal point coordinates $\varphi(x, t)$, as shown in equation (10):

$$\varphi : (x, t) \rightarrow \{(x, t), (x^2, t^2), \sin(x, t), \cos(x, t)\} \quad (10)$$

In addition, in the construction of loss function. Taylor PINN introduces a static weight parameter α balance the contribution of different loss terms to iterative convergence [34], so that the neural network model can better meet each loss

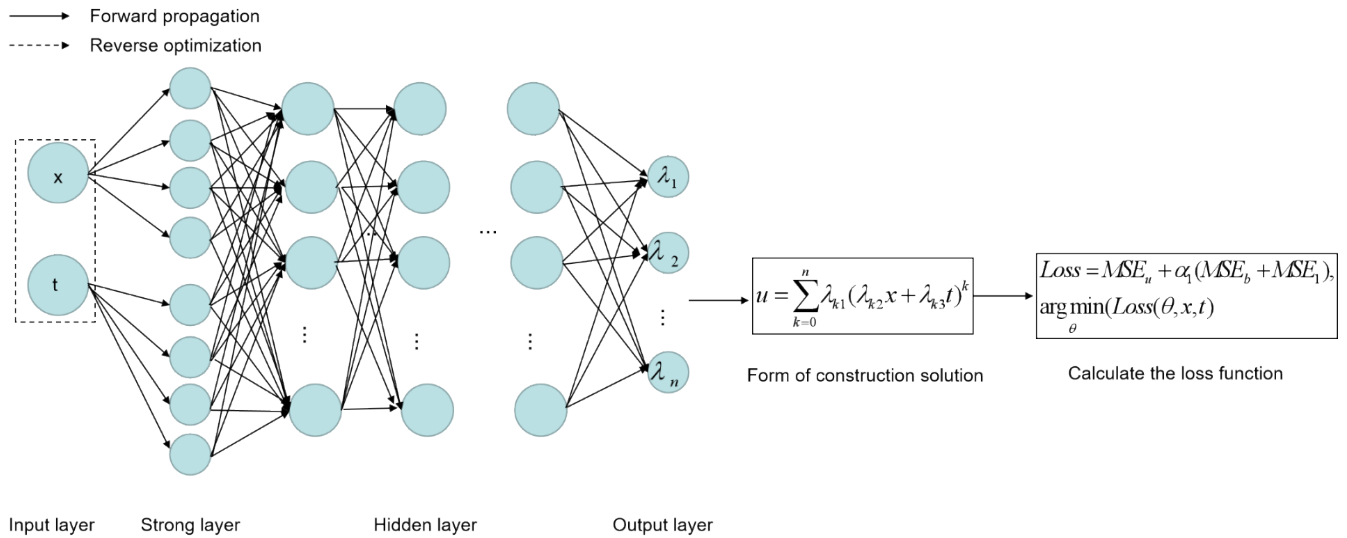


FIGURE 1. Overall structure of Taylor PINN.

term during the training and optimization stage, and thus converge to the optimal value faster. Finally, Taylor PINN’s loss function is shown in equation (11):

$$Loss = MSE_u + \alpha_1(MSE_b + MSE_1) \quad (11)$$

Among them, the equation residual term MSE_u is used to constrain the neural network to satisfy Control equation. Assuming $\{x_u^i, t_u^i\}_{i=1}^{N_u} \subset R^d$ is the time within the computational domain Empty points, N_u is the number of selected internal points. Equation residual term MSE_u calculation is shown in equation (12):

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |\frac{\partial^k u}{\partial t^k}(x, t) - D[u(x, t)]|^2 \quad (12)$$

The condition terms MSE_b and MSE_1 in the Loss function are used to approximate the optimization results of the bundle neural network satisfy the boundary and Initial condition. $\{x_b^i, t_b^i\}_{i=1}^{N_b} \subset R^d$ is the spatiotemporal point on each boundary, and the partial differential equation. The construction method of conditional items is shown in equations (13):

$$MSE_b = \frac{1}{N_b} \sum_{i=1}^{N_b} B[u(x, t)] - h(x, t)|^2 \quad (13)$$

$$MSE_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} [\frac{\partial^l u}{\partial t^l}(x, 0) - g(x)]^2 \quad (14)$$

By minimizing the Loss function Loss, Taylor PINN can utilize the nerve the function. It can fit ability of the Taylor PINN satisfies both equation terms and conditions simultaneously. The approximate solution u to be solved θ , Namely:

$$u \approx u(\theta) \quad (15)$$

The approximate solution is represented by a Feedforward neural network Any point coordinate in the domain, the trained network model can pass through Efficient prediction of corresponding function values through several matrix multiplication, thereby achieving End-to-end mapping of point

coordinates to physical fields. PINN based training The practice and prediction process eliminates dependence on computational grids, greatly improving Improved the efficiency of numerical analysis.

Figure 1 shows the overall structure of Taylor PINN, which includes an input layer, an input enhancement layer, a hidden layer for high-dimensional function space feature learning, and the final output layer. Among them, the input layer of Taylor PINN takes the spatiotemporal coordinates of the computational domain as input and propagates to a hidden layer composed of fully connected layers after dimensionality expansion. Forward propagation to obtain output λ . Calculate the value of the function to be solved as a coefficient in the solution form based on Taylor’s formula. In the training phase, the function value obtained is used to calculate the value of the loss function and the gradient of back propagation. In the prediction phase, this output value will be directly used to calculate the predicted solution of the function to be solved.

The training process of Taylor PINN is shown in figure 1. Taylor PINN training process input: Calculate the spatiotemporal point coordinates within the domain. Output: The trained Taylor PINN model.

Step1: Select spatiotemporal point coordinates from both the computational domain and the boundary to form a training sample set.

Step2: Build the Taylor PINN neural network and build the loss function Loss according to the partial differential equation to be solved [35].

Step3: Initialize network parameters.

Step4: Select spatiotemporal point coordinates from the sample set as training samples and inputs them into Taylor PINN.

Step5: Use the optimization algorithm to minimize the loss function Loss, and update the optimization parameters in the neural network.

Step6: Repeat steps 4 and 5 until the predetermined number of iteration rounds are completed.

C. HELMHOLTZ EQUATION

In the second test case, this paper uses Taylor PINN to solve the two-dimensional Helmholtz equation. The equation is widely used in the fields of acoustics, electromagnetism and elasticity [36]. The two-dimensional Helmholtz equation is shown in equation (16):

$$\Delta u(x, y) + k^2 u(x, y) = q(x, y), (x, y) \in \Omega \quad (16)$$

The boundary conditions of the equation are shown in equation (17):

$$u(x, y) = h(x, y), (x, y) \in \partial\Omega \quad (17)$$

Among them, Δ is a Laplace operator, and the calculation domain Ω is $[0, 1] \times [0, 1]$. When $k = 1$ and the source term $q(x, y)$ is shown in equation (18):

$$\begin{aligned} q(x, y) = & -\pi^2 \sin(\pi x) \sin(4\pi y) \\ & - (4\pi)^2 \sin(\pi x) \sin(4\pi y) \\ & + k^2 \sin(\pi x) \sin(4\pi y) \end{aligned} \quad (18)$$

An analytical solution to the equation can be constructed, as shown in equation (15):

$$u_{ref} = \sin(\pi x) \sin(4\pi y) \quad (19)$$

In Taylor PINN, the loss function corresponding to Helmholtz equation is shown in equation (20):

$$\begin{aligned} Loss = & \frac{1}{N_u} \sum_{i=1}^{N_u} |\Delta u(x, y) + u(x, y) - q(x, y)|^2 \\ & + \frac{\alpha_1}{N_b} \sum_{i=1}^{N_b} |u(x, y) - h(x, y)|^2 \end{aligned} \quad (20)$$

Table 1 shows the L^2 errors of PINN and Taylor PINN when the number of single layer neurons varies with a fixed network layer of 3. From Table 2, it can be seen that as the number of neurons increases, both network models can achieve improved prediction accuracy [37].

TABLE 1. Effect of the number of neurons in a single-layer network on the predictive performance of Taylor PINN.

Number of layer neurons	L^2 -error	
	PINN	Taylor PINN
10	4.67E+00	7.38E-02
20	8.23E-01	6.60E-02
30	2.48E-01	3.56E-02
40	3.05E-01	2.45E-02
50	2.27E-01	1.86E-02
60	2.25E-01	2.46E-02
70	1.33E-01	1.86E-02
80	5.52E-01	2.46E-02
90	1.10E-01	1.66E-02
100	3.66E-01	8.23E-02

III. EXPERIMENTS AND RESULTS

A. NUMERICAL EXAMPLES

In the process of neural network training, this paper uses multilayer perceptron, which contains two hidden layers, each hidden layer contains 30 hidden units, and a linear output unit. The activation function of each hidden unit is $\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Determine the computational boundary and select the boundary N training points. At the same time, arbitrarily select M training points within the calculation area, and use two parts of the training points ($M + N$) as the training set to learn the neural network model in the TensorFlow framework. Through training, obtain the optimal parameters of the neural network and the approximate solution of the model. Then, select enough points in the calculation area as the test set, and test the approximate solution after training to verify the accuracy of the method.

Firstly, consider the Poisson equation with mixed boundary conditions:

$$\begin{cases} \nabla^2 u(x, y) = e^{-x}(x^{-2} + y^3 + 6y) \\ (x, y) \in \Omega \\ u(0, y) = y^3, u(1, y) = e^{-1}(1 + y^3) \\ u(x, 0) = xe^{-x}, \frac{\partial u}{\partial y}|_{y=1} = 3e^{-x} \end{cases} \quad (21)$$

Among them, $x, y \in [0, 1]$. Take the calculation area Ω as a square, randomly select $M = 500$ training points within Ω , uniformly select $N = 200$ training points on partial Ω , and use M and N together as the training set, as shown in Figure 2. The analytical solution of the equation is:

$$u(x, y) = e^{-x}(x + y^3) \quad (22)$$

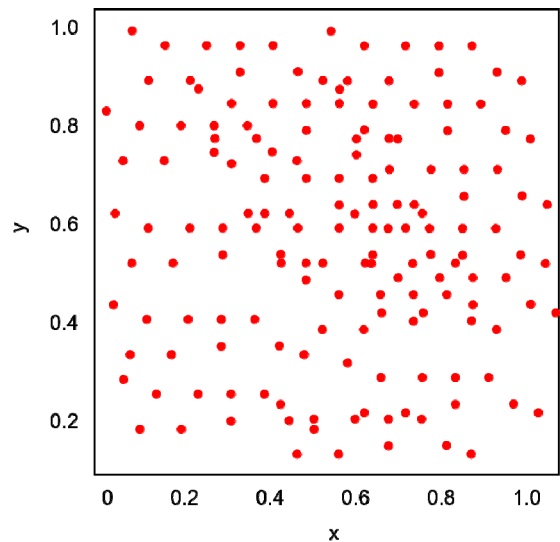


FIGURE 2. Training set.

The loss function is constructed as follows:

$$\begin{aligned} J(\theta) = & \frac{1}{N_{int}} \sum_{i=1}^{N_{int}} (\nabla^2 N(x, y, \theta) \\ & - e^{-x}(x^{-2} + y^3 + 6y))^2 \end{aligned}$$

$$\begin{aligned}
 & + \frac{\lambda_1}{N_{dir}} \sum_{p=1}^{N_{dir}} (N(x, y, \theta) - g(x, y))^2 \\
 & + \frac{\lambda_2}{N_{neu}} \sum_{q=1}^{N_{neu}} \left(\frac{\partial N(x, y, \theta)}{\partial y} - 3e^{-x} \right)^2 \quad (23)
 \end{aligned}$$

Among them, λ_1 is 1000, λ_2 is 30, select learning rate η . After 30000 iterations of training with a value of 0.001, the training solution of the equation was obtained. Taking $M = 20000$ and $N = 2000$ as the test set, the analytical and DNN approximate solutions of the test set are shown in Figures 3 and 4. As shown in the figure, the blue, purple, orange, and yellow colors represent the approximate solution values.

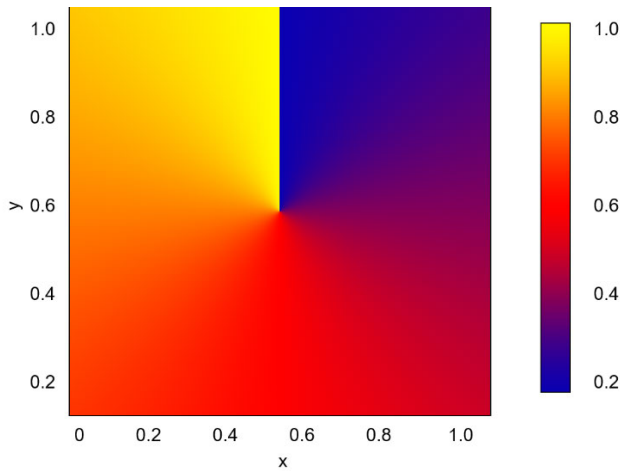


FIGURE 3. Analytical solution.

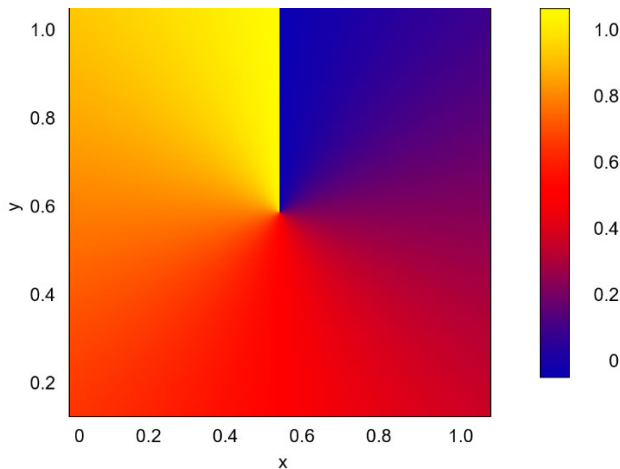


FIGURE 4. DNN approximate solution.

Figure 5 shows the errors on the test, and it can be seen that the error between the neural network solution and the analytical solution is -1% - 5%, indicating an ideal effect. As shown in the figure, the blue purple orange yellow color in the figure represents the test error value of the approximate solution.

For the Poisson equation, five gradient optimization algorithms, Adam, SGD, Adadelata, RMSProp, Adagrad, are used

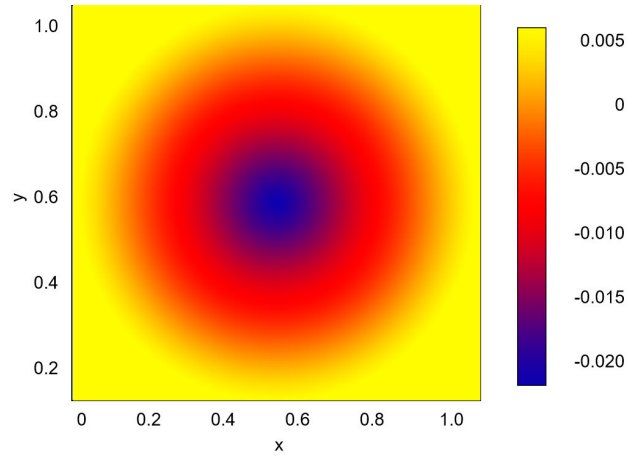


FIGURE 5. Test set error.

to process the loss function. The convergence curve of the loss function is shown in Figure 6. The horizontal axis represents the number of iterations of the gradient descent algorithm, and the vertical axis represents the loss value of the loss function. With the increase of the number of iterations, the loss value gradually decreases. Among them, the optimization effect of the Adam optimization algorithm on the model is better than other methods, The main reason is that this algorithm combines the advantages of the Adarad and RMSProp algorithms. Compared to traditional gradient descent algorithms, the Adam optimization algorithm has adaptive ability and faster computational convergence. The SGD optimization algorithm results in similar gradient descent curve effects to the Adam algorithm, but when the dataset is large, the SGD algorithm has better convergence speed and efficiency than Adam. So suitable optimization algorithms should be selected in different situations. This article's calculations show that using the Adam optimization algorithm can achieve ideal results.

The advection equation is of great significance in the study of atmospheric motion. The motion equation, heat flux equation, and water vapor equation all contain advection terms. However, numerical solution methods are difficult to obtain analytical solutions for such equations, so studying neural networks to solve advection equations is of great practical significance.

$$\begin{cases}
 Lu = a \frac{\partial^2 u}{\partial x^2} + b \frac{\partial^2 u}{\partial y^2} = f(x, y) \\
 (x, y) \in \Omega \\
 u(1, y) = \sin(3\pi) \cos(3\pi) \\
 \frac{\partial u}{\partial y} \Big|_{y=1} = -3\pi \sin(3\pi x) \sin(3\pi) \\
 \frac{\partial u}{\partial y} \Big|_{y=0} = 0
 \end{cases} \quad (24)$$

where: $x, y \in [0, 1]$, and the mixed boundary conditions are met, the analytical solution of the equation is:

$$u(x, y) = \sin(3\pi x) \cos(3\pi y) \quad (25)$$

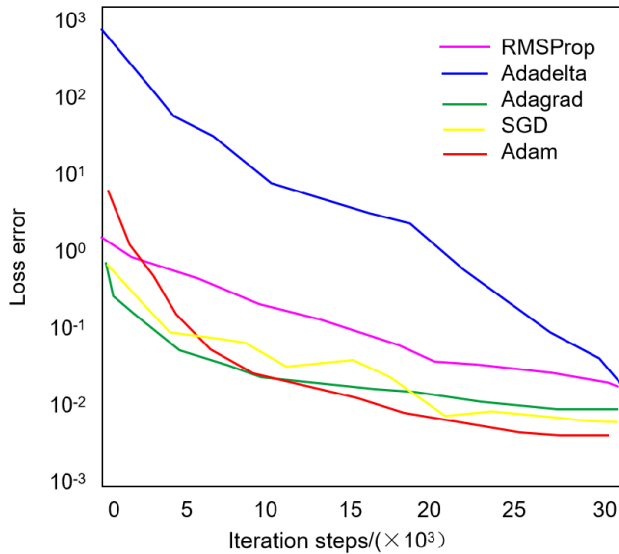


FIGURE 6. Convergence of different optimization methods.

Among them, a and b are the advection coefficients. In this problem, take $a = 2$ and $b = 3$, take the calculation area Ω as a square, generate $M = 500$ arbitrary training points within Ω , and generate $N = 200$ training points on partial Ω , as shown in Figure 7.

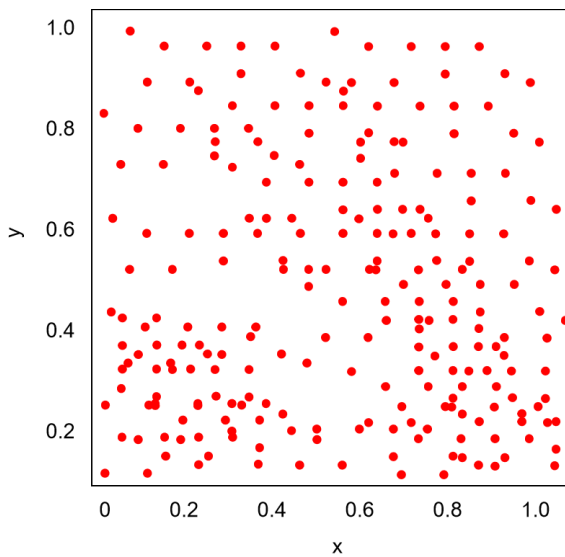


FIGURE 7. Training set.

Figures 8 and 9 show the function results of the test set on the analytical solution and DNN solution, respectively.

B. KLEIN-GORDON EQUATION

This section uses three numerical cases, Klein Gordon equation, Helmholtz equation and Navier Stokes equation [41]. Taylor PINN’s ability to solve partial differential equations, and compare its prediction performance with

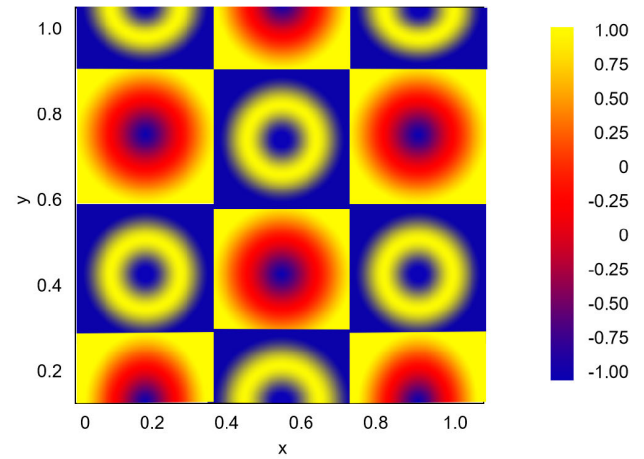


FIGURE 8. Analytical solution.

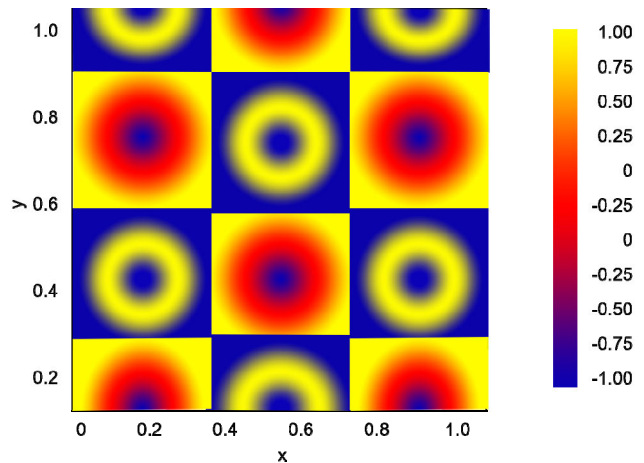


FIGURE 9. DNN approximate solution.

traditional PINN under the conditions of the same network structure, hyperparameter and number of training iterations. The initial learning rates used in the experiment were all 0.001, with a decay rate of 0.9 per 1000 rounds. The optimizer used Adam with a batch size of 128. The testing platforms for the experiment are IntelCorei7-6700 and NVIDIA Tesla P100, and the experimental environment is TensorFlow1.15.0. The differential terms in partial differential equations are unified using the automatic differential function in TensorFlow. The prediction accuracy of the neural network model is measured using L^2 error, and its calculation method is shown in equation (26):

$$L^2 - error = \frac{|u_{ref} - u_{pred}|^2}{|u_{ref}|^2} \tag{26}$$

Among them, u_{ref} is the reference solution of the equation to be solved, usually represented by analytical or high-precision numerical solutions. $U_{n,pred}$ is a predictive solution provided by neural networks.

In the first test case, Klein Gordon equation is used to evaluate the fitting ability of the proposed method. Klein Gordon

equation is one of the basic partial differential equations in quantum physics, solid-state physics and other fields. The form of the one-dimensional Klein Gordon equation is shown in equation (26):

$$u_{tt} + \alpha u_{xx} + \beta u^\gamma = q(x, t),$$

$$(x, t) \in \Omega \times [0, T], \Omega = [0, 1] \quad (27)$$

The boundary conditions are shown in equation (28):

$$u(x, t) = h(x, t), (x, t) \in \partial\Omega \times [0, T] \quad (28)$$

The initial conditions are as shown in equations (29) and (30):

$$u(x, 0) = g_1(x), x \in \Omega \quad (29)$$

$$u_t(x, 0) = g_2(x), x \in \Omega \quad (30)$$

When $g_1(x) = g_2(x) = 0, \alpha = \beta = 1, \gamma = 3$. We can construct an analytical solution of the equation as a reference solution for neural network prediction, as shown in equation (31):

$$u_{ref}(x, t) = x \cos(5\pi t) + (xt)^3 \quad (31)$$

In this paper, $h(x, t)$ and $q(x, t)$ are obtained by using the analytic solution, which is used to calculate the residual and condition terms in the loss function later. For partial differential equations (equations (29) to (31)).

In this case, this article constructs a Taylor PINN network model with 3 hidden layers, with 100 neurons per layer. In the training phase, this paper seeks the optimal network parameters by minimizing the loss function to fit the closed prediction solution of Klein Gordon equation. We set the total number of training rounds to 4000, with static weights α_1 is 10. The test set consists of 10000 spatiotemporal points uniformly distributed within the computational domain. The prediction accuracy (L^2 error) of Taylor PINN under different orders of Taylor formula is shown in Figure 10.

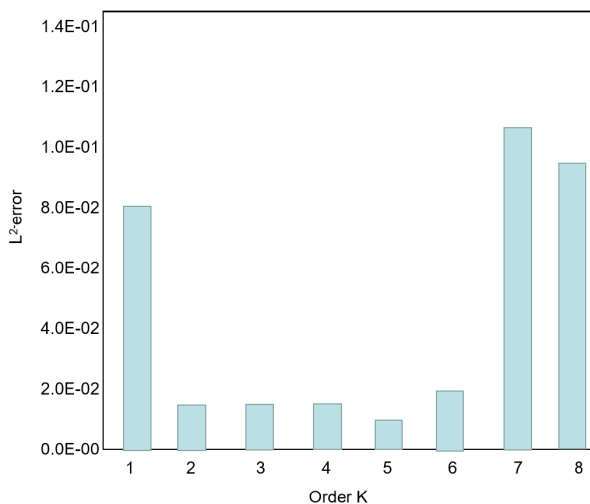


FIGURE 10. Taylor PINN under different orders.

In this case, this article tested the impact of different network sizes on the predictive performance of Taylor PINN

by changing the number of network layers and single-layer neurons. In order to better analyze the performance results at different scales, this article compares Taylor PINN with PINN at the same scale. Figure 11 shows the prediction results of Taylor PINN on the Helmholtz equation case when the order is 3 and the number of fixed single-layer neurons is 50.

From Figure 11, it can be seen that when the number of network layers is 3. Taylor PINN achieves the best prediction performance, with an L^2 error of 1.86E-02. When the number of network layers continues to increase, the prediction accuracy will actually deteriorate. However, Taylor PINN has achieved higher prediction performance than PINN, and the prediction accuracy can be improved up to 12 times.

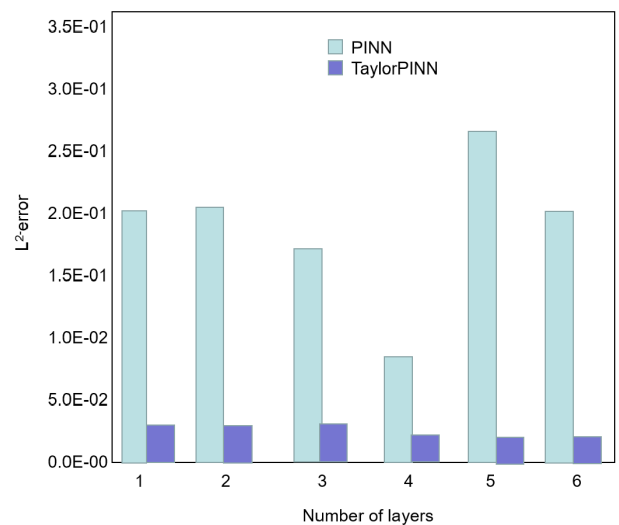


FIGURE 11. Prediction performance of Taylor PINN with different network layers on the Helmholtz equation.

Table 2 shows the L^2 errors of PINN and Taylor PINN when the number of single layer neurons varies with a fixed network layer of 3. From Table 2, it can be seen that as the number of neurons increases, both network models can achieve improved prediction accuracy.

TABLE 2. Effect of the number of neurons in a single-layer network on the predictive performance of Taylor PINN.

Number of layer neurons	L^2 -error	
	PINN	TaylorPINN
10	4.67E+00	7.38E+00
20	8.23E+00	6.06E+00
30	2.48E+00	3.56E+00
40	3.05E+00	2.45E+00
50	2.27E+00	1.86E+00
60	2.25E+00	2.46E+00
70	1.33E+00	1.94E+00
80	5.52E+00	1.52E+00
90	1.10E+00	1.66E+00
100	3.66E+00	1.07E+00

C. ABLATION EXPERIMENT AND COMPARISON OF RESULTS

In this case, this article tested the impact of different network sizes on the predictive performance of Taylor PINN by changing the number of network layers and single-layer neurons. In order to better analyze the performance results at different scales, this article compares Taylor PINN with PINN at the same scale. Figure 12 shows the prediction results of Taylor PINN on the Helmholtz equation case when the order is 3 and the number of fixed monolayer neurons is 50. From Figure 12, it can be seen that when the number of network layers is 3, Taylor PINN achieves the best prediction performance, with an L2 error of 1.86E-02. When the number of network layers continues to increase, the prediction accuracy will actually deteriorate. However, under different network layers, Taylor PINN achieved higher prediction performance than PINN, and the prediction accuracy can be improved up to 12 times. The purple color in the figure represents the loss function error of PNN, while the green color represents the Taylor PINN loss function error.

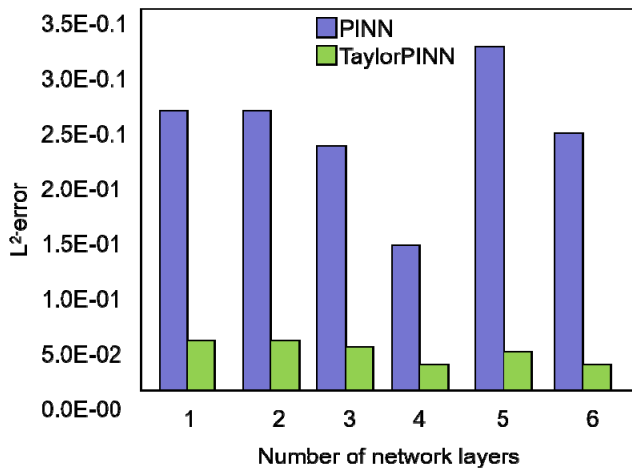


FIGURE 12. Prediction performance of Taylor PINN with different network layers on the Helmholtz equation.

To test the accuracy and speed of the method r in solving partial differential equations, a certain partial differential equation is given, and the method used in this paper is applied to solve the given partial differential equation. The solution results and process time are compared with other methods to verify the effectiveness of the method used in this paper. The partial differential equation given is:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = (\lambda^2 + \mu^2) \exp(\lambda x_1 + \mu x_2) \quad (32)$$

In the equation, the boundary conditions are $x_1 \in [0, 1], x_2 \in [0, 1]$:

$$\begin{cases} u = \exp(\lambda x_1 + \mu x_2), & x_1 = 0, x_2 = 1 \\ u_1 = \lambda \exp(\lambda x_1 + \mu x_2), & x_1 = 0, x_2 = 1 \end{cases} \quad (33)$$

In the formula: Take $\lambda = 2, \mu = 3$. The exact solution of equation is:

$$u_e = (x_1, x_2) = \exp(\lambda x_1 + \mu x_2) \quad (34)$$

Under 50 training samples, the standard deviation and root mean square error of the solution obtained by this method at different levels were compared with the exact solution. The comparison results are shown in Table 3. From Table 3, it can be seen that under a fixed number of training samples, as the number of layers increases, the standard deviation and root mean square error of the method presented in this paper show a gradually decreasing trend. Especially when the method used in this paper is a 2-layer structure, the standard deviation and root mean square error are reduced by one order of magnitude compared to the 1-layer structure, with the most significant decrease; Starting from the 5-layer structure, the decrease in standard deviation and root mean square error begins to decrease. Therefore, the 4-layer structure proposed in this paper is used for solution and compared with other methods.

TABLE 3. Comparison of solution accuracy for different layers of this method.

Number of layers	standard deviation	Root mean square error
1	8.75×10^{-4}	6.39×10^{-4}
2	8.12×10^{-4}	5.65×10^{-4}
3	6.33×10^{-4}	4.45×10^{-4}
4	5.03×10^{-4}	3.26×10^{-4}
5	4.85×10^{-4}	2.83×10^{-4}
6	4.62×10^{-4}	2.52×10^{-4}
7	4.38×10^{-4}	2.11×10^{-4}

The number of training samples is selected as 10, 20, 30, 40, 50, and the reduced basis finite element method for solving parametric partial differential equation and the method for solving wave equation based on the AH orthogonal basis function are used as the comparison method of the method in this paper. In this paper is used to solve the problem, and the standard deviation and root mean square error between the solution results of each method and the accurate solution are compared, as shown in Figure 13.

From Figure 13, it can be concluded that under the same number of training samples, the standard deviation and root mean square error of the solution results of the three methods are ranked in descending order. This indicates that the solution accuracy is directly proportional to the number of training samples, and the solution accuracy of this method is improved compared to the other two comparative methods.

Implement statistics on the duration of each method's solving process, and compare the solving speed of each method based on the statistical results, in order to further test the solving effect of this method. The statistical results are shown in Figure 14. Analyzing Figure 14, it can be concluded that as the number of training samples increases, the duration of obtaining high-precision solutions for each method also increases. However, under the same number of

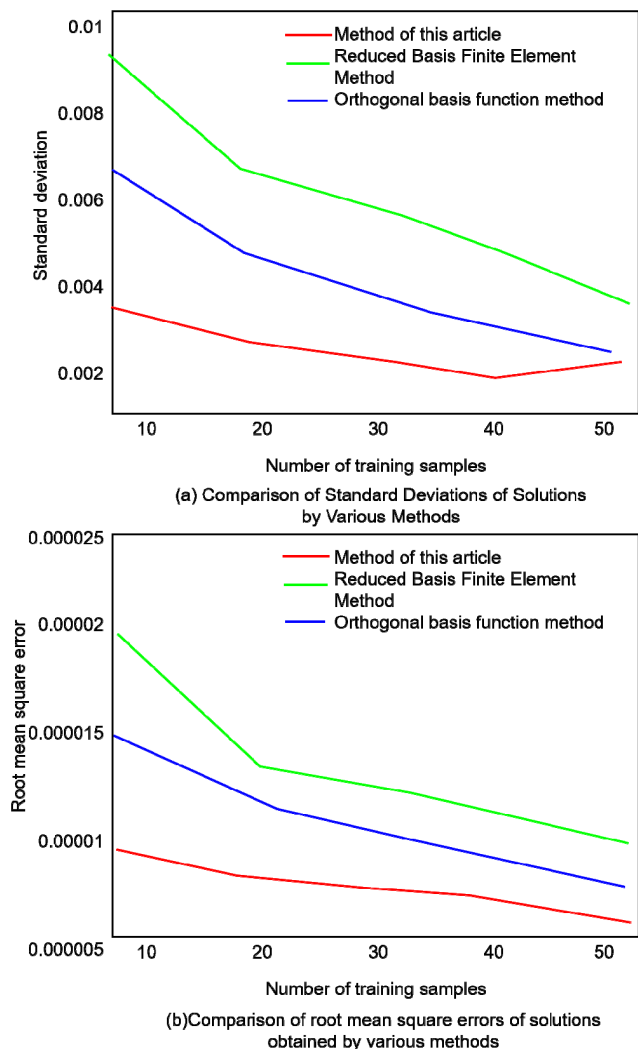


FIGURE 13. Comparison of solution results of various methods.

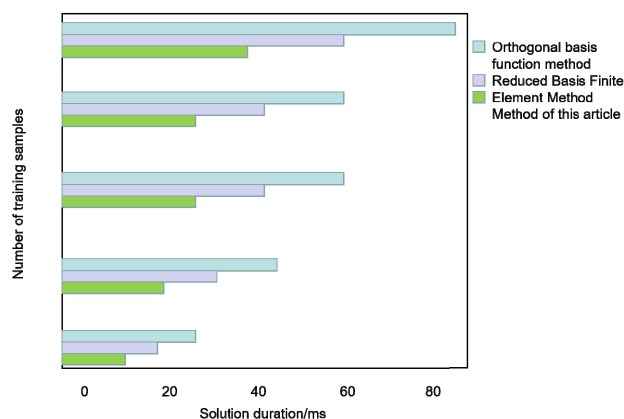


FIGURE 14. Statistical results of solution time for each method.

training samples, the high-precision solution duration of our method is lower than that of the other two methods, indicating that our method has a higher solution speed in high-precision solving partial differential equations. Based on the results of

all the above examples, it can be seen that compared with the two comparative methods, the proposed method has higher solution accuracy and speed, and performs better in solving partial differential equations. It can achieve high-precision and efficient solutions for partial differential equations.

IV. CONCLUSION

We propose a Taylor PINN method to solve elliptic partial differential equations in complex domains. This method uses a completely unsupervised gradient descent algorithm for iterative training, without the need to call other numerical solvers for partial differential equations during the solution process. Numerical examples show that under the neural network computing framework provided in this paper, approximate solutions of any elliptical partial differential equation can be obtained through training in any region, and have high accuracy. In future work, consider further research on network models, such as introducing attention mechanisms into network models to improve the accuracy of the model; The proposed methods will also be improved and applied to more complex physical problem scenarios to further enhance their practicality.

REFERENCES

- [1] R.-F. Zhang and S. Bilige, "Bilinear neural network method to obtain the exact analytical solutions of nonlinear partial differential equations and its application to p-gBKP equation," *Nonlinear Dyn.*, vol. 95, no. 4, pp. 3041–3048, Mar. 2019.
- [2] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," in *Proc. AAAI Spring Symp.*, 2021, pp. 2002–2041.
- [3] R. Zhang, S. Bilige, and T. Chaolu, "Fractal solitons, arbitrary function solutions, exact periodic wave and breathers for a nonlinear partial differential equation by using bilinear neural network method," *J. Syst. Sci. Complex.*, vol. 34, no. 1, pp. 122–139, Feb. 2021.
- [4] Z. Sabir, M. R. Ali, M. A. Z. Raja, R. Sadat, and D. Baleanu, "Dynamics of three-point boundary value problems with gudermannian neural networks," *Evol. Intell.*, vol. 16, no. 2, pp. 697–709, Apr. 2023.
- [5] J. Jia and A. R. Benson, "Neural jump stochastic differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–12.
- [6] X. Youxiong, "Accurate solution of kinematics of Hongwei 3-RPR parallel mechanism," *Mach. Tool. Hydraul.*, vol. 50, no. 23, pp. 6–11, 2022.
- [7] G. Xiaobin, "Solving partial differential equations in complex regions based on deep neural networks," *J. Lanzhou Univ. Technol.*, vol. 48, no. 6, pp. 149–157, 2022.
- [8] L. Jingwei, "Artificial intelligence and deep learning methods for solving differential equations: Status quo and prospects," *J. Intell. Sci. Technol.*, vol. 4, no. 4, pp. 461–476, 2022.
- [9] J. Berner, P. Grohs, and A. Jentzen, "Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black-scholes partial differential equations," *SIAM J. Math. Data Sci.*, vol. 2, no. 3, pp. 631–657, Jan. 2020.
- [10] G. Chang, "A time-varying complex matrix equation based on finite time neural networks," *J. Zhejiang Univ. Sci. Technol.*, vol. 34, no. 5, pp. 409–418, 2022.
- [11] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 6696–6707.
- [12] H. Zhongmin, "A deflection moment coupling neural network method for in-plane variable stiffness thin plate bending problems," *J. Mech.*, vol. 53, no. 9, pp. 2541–2553, 2021.
- [13] K. I. D. Nguyen and X. Zhuang, "A structural optimization algorithm based on deep neural networks," *J. Zhejiang Univ. Sci. A*, vol. 22, no. 8, pp. 609–624, 2021.

- [14] J. Berg and K. Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," *Neurocomputing*, vol. 317, pp. 28–41, Nov. 2018.
- [15] Z. Qiang, "Research on the solution of shield tunneling pose based on neural network-Newton hybrid algorithm," *Mech. Transm.*, vol. 45, no. 7, pp. 24–29, 2021.
- [16] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–13.
- [17] S. Chakraborty, "Transfer learning based multi-fidelity physics informed deep neural network," *J. Comput. Phys.*, vol. 426, Feb. 2021, Art. no. 109942.
- [18] M. L. Piscopo, M. Spannowsky, and P. Waite, "Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions," *Phys. Rev. D, Part. Fields*, vol. 100, no. 1, Jul. 2019, Art. no. 016002.
- [19] J. Darbon, G. P. Langlois, and T. Meng, "Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures," *Res. Math. Sci.*, vol. 7, no. 3, pp. 1–50, Sep. 2020.
- [20] R. Rodriguez-Torrado, P. Ruiz, L. Cueto-Felgueroso, M. C. Green, T. Friesen, S. Matringe, and J. Togelius, "Physics-informed attention-based neural network for hyperbolic partial differential equations: Application to the Buckley–Leverett problem," *Sci. Rep.*, vol. 12, no. 1, p. 7557, May 2022.
- [21] V. Dwivedi, N. Parashar, and B. Srinivasan, "Distributed physics informed neural network for data-efficient solution to partial differential equations," 2019, *arXiv:1907.08967*.
- [22] L. Weihua and Z. Chong, "Low-light laser image enhancement at night based on 3D virtual technology," *Laser J.*, vol. 42, no. 9, pp. 119–123, 2021.
- [23] D. R. Parisi, M. C. Mariani, and M. A. Laborde, "Solving differential equations with unsupervised neural networks," *Chem. Eng. Process., Process Intensification*, vol. 42, nos. 8–9, pp. 715–721, Aug. 2003.
- [24] Y. Longquan and J. Wei, "Gradient descent neural network method for solving nonlinear equations with singular Jacobian matrices," *J. Hubei Univ. Eng.*, vol. 40, no. 6, pp. 69–73, 2020.
- [25] Y. Longquan, "Neural network method for solving absolute value equations and linear complementarity," *J. Shaanxi Univ. Technol.*, vol. 36, no. 5, pp. 72–81, 2020.
- [26] A. Hasan, J. M. Pereira, R. Ravier, S. Farsiu, and V. Tarokh, "Learning partial differential equations from data using neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3962–3966.
- [27] Y. Chen and J. W. L. Wan, "Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions," *Quant. Finance*, vol. 21, no. 1, pp. 45–67, Jan. 2021.
- [28] Y. Chen, H. Yu, X. Meng, X. Xie, M. Hou, and J. Chevallier, "Numerical solving of the generalized black-scholes differential equation using Laguerre neural network," *Digit. Signal Process.*, vol. 112, May 2021, Art. no. 103003.
- [29] W. Jing, "Precision diagnosis of tiny breast tumors based on laser image processing technology," *Microcomput. Appl.*, vol. 37, no. 9, pp. 126–129, 2021.
- [30] W. Qichang, "Solution of pursuit and escape strategies for infinite time domain spacecraft based on deep neural networks," *Aerosp. Control*, vol. 37, no. 6, pp. 13–18&58, 2019.
- [31] M. Poli, S. Massaroli, A. Yamashita, H. Asama, and J. Park, "TorchDyn: A neural differential equations library," 2020, *arXiv:2009.09346*.
- [32] H. Zhang, X. Gao, J. Unterman, and T. Arodz, "Approximation capabilities of neural ODEs and invertible residual networks," 2019, *arXiv:1907.12998*.
- [33] X. Fan, "Solving inverse kinematics of planar 2R manipulator based on BP neural network with unique characteristics," *J. Hunan Univ. Technol.*, vol. 33, no. 5, pp. 51–56, 2019.
- [34] T. Bo, "Calculation of IGBT module switch losses based on BP neural network," *High Voltage Electr. Appl.*, vol. 55, no. 7, pp. 27–32, 2019.
- [35] X. Fan, "Multi module neural network solution for inverse kinematics of space 3R manipulator," *China Mech. Eng.*, vol. 30, no. 10, pp. 1233–1238, 2019.
- [36] S. Chakraverty and S. Mall, "Single layer Chebyshev neural network model with regression-based weights for solving nonlinear ordinary differential equations," *Evol. Intell.*, vol. 13, no. 4, pp. 687–694, Dec. 2020.
- [37] R. Matthey and S. Ghosh, "A physics informed neural network for time-dependent nonlinear and higher order partial differential equations," 2021, *arXiv:2106.07606*.
- [38] L. Jie, "Research on artificial intelligence design method of appearance color matching based on product function," *J. Changchun Normal Univ.*, vol. 40, no. 10, pp. 38–44, 2021.
- [39] J. Y. Araz, J. Carlos Criado, and M. Spannowsky, "Elvet—A neural network-based differential equation and variational problem solver," 2021, *arXiv:2103.14575*.
- [40] C. J. Zúñiga-Aguilar, A. Coronel-Escamilla, J. F. Gómez-Aguilar, V. M. Alvarado-Martínez, and H. M. Romero-Ugalde, "New numerical approximation for solving fractional delay differential equations of variable order using artificial neural networks," *Eur. Phys. J. Plus*, vol. 133, no. 2, pp. 1–16, Feb. 2018.
- [41] Y. He, Z. Meng, H. Xu, and Y. Zou, "A dynamic model of evaluating differential automatic method for solving plane problems based on BP neural network algorithm," *Phys. A, Stat. Mech. Appl.*, vol. 556, Oct. 2020, Art. no. 124845.



YAJUAN ZHANG was born in Huanggang, Hubei, China, in 1988. She received the master's degree from Hainan University, China. She is currently with the Department of Information Engineering, Hainan Vocational University of Science and Technology. Her research interests include electronic communication, the Internet of Things engineering, and big data.



MIN WANG was born in Rongcheng, Shandong, China, in 1982. She received the master's degree from the Dalian University of Technology. She is currently with Shandong Jiaotong University. Her research interests include deep learning and image processing.



FANGWEI ZHANG (Senior Member, IEEE) was born in Heze, Shandong, China, in 1980. He is currently a Professor, a Ph.D. Supervisor, the Discipline Leader, a Doctor, and a Postdoctoral Researcher in transportation planning and management.



ZHENRUI CHEN (Member, IEEE) was born in Heze, Shandong, China, in 1989. He received the master's degree from the Beijing University of Technology. His research interests include deep learning and image processing.

...