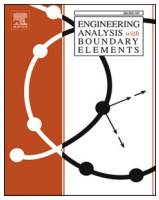
Engineering Analysis with Boundary Elements 166 (2024) 105802

Contents lists available at ScienceDirect

Engineering Analysis with Boundary Elements

journal homepage: www.elsevier.com/locate/enganabound



A gradient-enhanced physics-informed neural networks method for the wave equation

Guizhong Xie [a,*], Beibei Fu [a], Hao Li [a], Wenliao Du [a], Yudong Zhong [a], Liangwen Wang [a], Hongrui Geng [b], Ji Zhang [c], Liang Si [d]

[a] Henan Provincial Key Laboratory of Intelligent Manufacturing of Mechanical, Henan International Joint Laboratory of Complex Mechanical Equipment Intelligent Monitoring and Control, Zhengzhou University of Light Industry, Zhengzhou 450002, Henan, China
[b] College of Mathematics and Information Science, Zhengzhou University of Light Industry, Zhengzhou 450002, Henan, China
[c] Longmen Laboratory, Henan University of Science and Technology, Luoyang, China, Luoyang 471009, Henan, China
[d] School Of Materials Science And Engineering, Zhongyuan Critical Metals Laboratory, Zhengzhou University, Zhengzhou 450001, Henan, China

ARTICLE INFO

Keywords:
Gradient-enhanced physics-informed neural networks
Wave equations
Boundary hard constraint
Partial derivative loss function

ABSTRACT

Physics-informed neural networks (PINNs) have been proven to be a useful tool for solving general partial differential equations (PDEs), which is meshless and dimensionally free compared with traditional numerical solvers. Based on PINNs, gradient-enhanced physics-informed neural networks (gPINNs) add the partial derivative loss term of the independent variable and the physical constraint term, which improves the accuracy of network training. In this paper, the gPINNs method is proposed to solve the wave problem. The equation boundary value of the wave problem is added in the network construction, thus the network is forced to satisfy the equation boundary value during each training process. Two examples of wave equations are given in our paper, which are solved numerically by PINNs and gPINNs, respectively. It is found that using fewer data sets, gPINNs can learn data features more fully than PINNs and better results of gPINNs can be obtained.

1. Introduction

Partial differential equations (PDEs) are an important part of mathematics and play an important role in many physics, chemistry, and finance disciplines [1]. Hyperbolic PDEs are one of the most important ones and are used in many theoretical analyses and practical applications. Wave equation [2] is a typical representative equation of hyperbolic PDEs, which has important applications in many scientific research, prediction, and medical fields. Wave equation has the characteristics of propagation and oscillation, which is challenging to solve. Moseley et al. [3] accurately simulated complex wave phenomena by combining deep learning and physics principles, which is of great significance for geophysical exploration and seismic monitoring. Song et al. [4] are committed to solving the VTI wave equation of acoustic waves in the frequency domain, improving the accuracy of acoustic wave propagation simulation, and providing a new approach to understanding the propagation behavior of acoustic waves in anisotropic media. Song and Wang [5] proposed the Fourier feature physics-informed neural network to simulate seismic multifrequency wave fields. It is worth noting that the solution of PDEs may be indeterminate, thus in the analysis of specific problems, it is necessary to understand the specific conditions, such as boundary conditions, to constrain the equation and get the specific solution of the equation. Analytical solutions for many equations in practical applications are not easy to obtain, thus solving PDEs numerically has become an important research in scientific computation. Classical numerical solution methods, such as finite element method [6,7], difference method [8,9], boundary element [10-14], etc., approach the solution of PDEs by constantly reducing the step size through grid division, which has high-cost requirements and technical breakthrough limitations [15]. So far, PINNs [16-20] have been proven successful in solving PDEs by many articles. By coding PDEs and networks of different structures together, the physical laws of PDEs are learned, and the solution of PDEs is obtained by using PINNs. PINNs is a new method that is different from the traditional method because it does not require meshing and is not constrained by high dimensions. However, when using general PINNs to solve PDEs, the numerical solution usually has certain errors [21,22].

For this problem, different optimization methods have been proposed in recent years. For example, in reference [23,24], the author proposed Gradient-optimized physics-informed neural networks

* Corresponding author.
E-mail address: xieguizhong@zzuli.edu.cn (G. Xie).

https://doi.org/10.1016/j.enganabound.2024.105802
Received 13 January 2024; Received in revised form 22 May 2024; Accepted 22 May 2024
Available online 3 June 2024

Table 1
The steps of solving wave equations with PINNs.

The wave equation is solved by PINNs
1. Learn the meaning of the wave equation
2. Create a neural network structure (input, x=(x, t), output, $y = u^*$(x, t))
3. Take the residual points T (T = $\{T_f \times T_b \times T_i\}$)
4. Design loss function $\mathscr{L}(w; b; T)$
a. Equation loss $\mathscr{L}_f(w; b; T_f)$
b. Boundary loss $\mathscr{L}_b(w; b; T_b)$
c. Initial loss $\mathscr{L}_i(w; b; T_i)$
5. Build a training model (optimizer, learning rate, iteration…)
6. Repeat
Training network
a. Forward training
b. Backpropagation (updated w,b, loss function reduced)
Until The loss function converges or reaches the maximum number of iterations
7. End the workout
8. Evaluate the model

(GOPINNs), which are more robust to gradient fluctuations. In reference [25], the author proposed Mix-training physics-informed neural networks, which have stronger approximation ability. When the space–time domain of the solution expands, or the solution has a local sharp region, the proposed model still has high prediction accuracy. In reference [26], the author systematically describes the influence of sampling and distribution of equation residual handicap on the solving performance of PINNs. In reference [27], an adaptive weight method (SA-PINNS) is proposed for the stubborn regions that are difficult to learn in PDEs' training. This method independently checks the region where the loss value is difficult to decrease in training, and constantly adjusts the weights of this region, so that the network focuses on learning the data features of this region and makes the loss amount tend to balance. Yu and Lu et al. have proposed gradient-enhanced physics-informed neural networks (gPINNs) [28], which added the partial derivative of equation residuals into the calculation of loss function as a new loss term, which increased constraints of the equation, hoping that the network could learn the data features of the equation more fully, which could theoretically make the network training more accurate. The deep energy method is proposed in reference [29]. In mechanics problems, the loss of the equation is constructed using a variational scheme from the energy conservation law of PDEs, and the boundary value problem of PDEs is solved. A hard constraint method is proposed in reference [30]. By modifying the Dirichlet boundary of the network structure forced fitting equation, the same result is obtained with general soft constraints. The hard constraint does not need to be data-driven, and the calculation results are optimized for the complex boundary-forced network fitting. Zhang introduced continuous symmetric physics-informed neural networks [31]. Zhu proposed to construct a neural network satisfying the spatio-temporal parity symmetry of PDEs [32]. In reference [33], a Bayesian physics-informed neural networks (B-PINNs) is introduced to solve the uncertainty PDEs caused by noise. Pang et al. proposed fractional-order neural networks [34] to solve the integral equation. Gu *et al.* proposed the SelectNet framework [35] to solve the problem of high-dimensional PDEs. Lu et al. proposed the deepxde framework [36] to solve complex region-solving problems. In reference [37], a variational physics-informed neural network is introduced.

In this paper, gPINNs are employed to solve the wave equation. In the calculation process, we also combined the hard constraints [30], built the equation boundary value into the network, and forced to meet the boundary value conditions of the equation in each iteration training, to enhance the training performance of the network. By comparing the L2 relative errors predicted by PINNs and gPINNs with the same sampling points, the robustness of gPINNs is demonstrated and better results can be obtained even when there are fewer data points.

The work is summarized as follows. In chapter 2, the wave equation is briefly introduced. In chapter 3, we introduce PINNs and gPINNs methods and how to use them to solve wave equations. In chapter 4, two examples of wave equations are given. PINNs and gPINNs methods are employed to solve them, and hard constraints are enforced in the process. In chapter 5, the research content of this paper is summarized.

2. Wave equation

The wave equation is an important equation in PDEs, which can describe various wave phenomena in nature, such as shear and longitudinal waves in earthquakes, ripples in the water surface, sound and light waves in the medical field, and so on. The wave equation is one type of PDEs with a second derivative of time t, and the general form is:

$$\frac{\partial^2 u}{\partial t^2} - a^2 \Delta u = f \tag{1}$$

where, $x \in \Omega \in R^n, t \in (0, \infty)$, u is an unknown function of x and t. $a^2$is the wave speed and Δu is the second derivative for x. f is the source term of the equation, a known function of the variable x. When f is 0, it is a homogeneous equation.

The wave equation is described by the string vibration problem. Assuming $x \in [0, l]$, $t \in (0, T]$ it on a bounded region Ω,$\Omega \in R^d$,$d = 1, 2$, and the initial displacement $A(x)$, the initial velocity $B(x)$. The initial conditions are as follows,

$$u(x,t)|_{t=0} = A(x) \tag{2}$$

$$\left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = B(x) \tag{3}$$

The boundary conditions are a function of the states at both ends of the string. When the two ends of the string are fixed in the horizontal state, the state of no displacement in the u direction, which is called the first type of boundary conditions, also known as Dirichlet boundary conditions, which is,

$$u(0,t) = 0 \tag{4}$$

$$u(l,t) = 0 \tag{5}$$

If the two ends of the string are not fixed and are in a free state, they can move freely in the u direction. This case is the second type of boundary of the equation, also known as Neumann boundary conditions, which represents the change in velocity at both ends of the string as follows,

$$\left.\frac{\partial u(x,t)}{\partial x}\right|_{x=0} = M(x) \tag{6}$$

$$\left.\frac{\partial u(x,t)}{\partial x}\right|_{x=l} = N(x) \tag{7}$$

There is also a third class of boundary conditions, where both displacement and velocity change at both ends of the string, called Robin boundary conditions. According to Hooke's law $F = -ku$, the force at the endpoints is $T\frac{\partial u(x,t)}{\partial x}|_{x=l} = -ku(l,t)$.

3. Methods

In this chapter, the PINNs and gPINNs methods are given and how to use them to solve the wave equation is introduced.

3. 1Physics-informed neural networks (PINNs)

PINNs is a deep artificial neural networks, that learn the physical laws of the equation by encoding them into it for iterative training, thus inferring the approximate solution. The problem of string vibration is analyzed in chapter 2, where the source term $f = 0$. We use PINNs to obtain the solution, steps such as Table 1.

The input node of the network is set to 2, x=(x,t), and the output

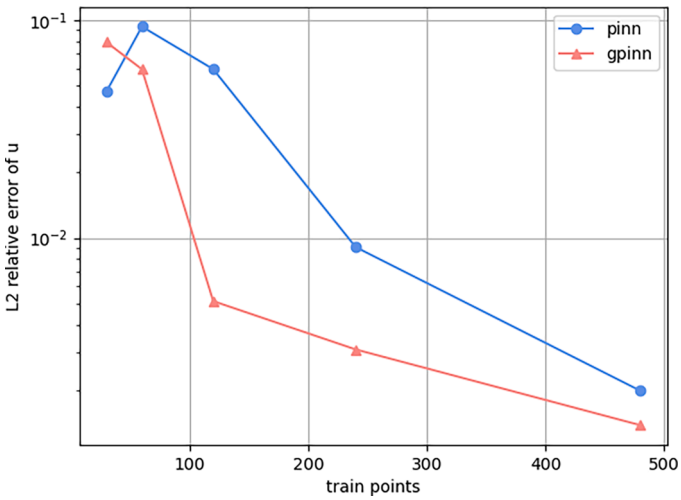


Fig. 1. L_2 relative errors by PINNs and gPINNs at different sampling points.

node is set to 1, $\boldsymbol{y} = u^*$(x,t|w, b), according to the one-dimensional string vibration equation. u*(x,t|w, b) is the result of network training. w,b are the parameters of the network nodes, which are updated with each training.

The residual points are taken within the range of equation variable values, and the method for taking can be referred to [23], where uniform sampling is used. The set of residuals can be expressed as $T = \{T_f \times T_b \times T_i\}$, where T_f is the point taken within the time-space, $T_b$is the point taken on the boundary, and $T_i$is the point taken in the initial state, respectively.

$$T_f = \{(x,t)|x \in [0,l], t \in (0,T]\}$$

$$T_b = \{(x,t)|x \in \Gamma, t \in (0,T]\}$$

$$T_i = \{(x,t)|x \in [0,l], t = 0\}$$

The loss function is designed using sampling points, and the design method applies the mean square error.

$$\mathscr{L}(w;b;T) = w_f \mathscr{L}_f(w;b;T_f) + w_b \mathscr{L}_b(w;b;T_b) + w_i \mathscr{L}_i(w;b;T_i)$$

where,$\mathscr{L}_f(w;b;T_f)$ is the space-time loss term, $\mathscr{L}_b(w;b;T_b)$ is the boundary loss term, $\mathscr{L}_i(w;b;T_i)$ is the initial loss term, and w_f,w_b, and w_i are the corresponding weights, respectively.

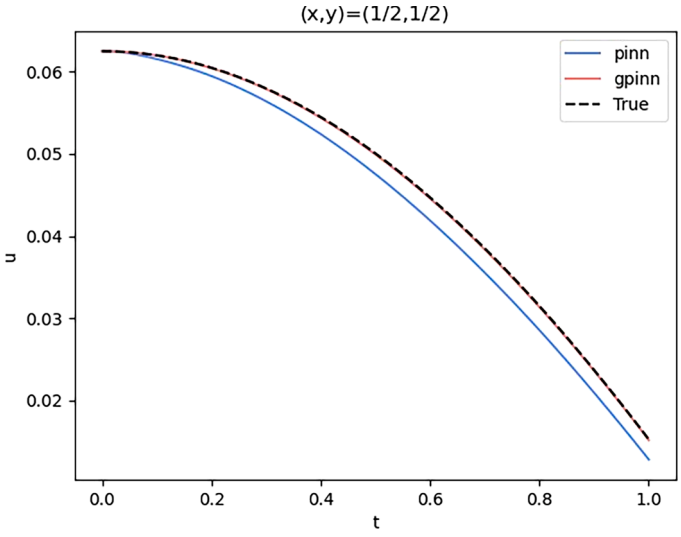


Fig. 2. (x,y)=(1/2,1/2), the change of u by PINNs and gPINNs.

$$\mathscr{L}_f(w;b;T_f) = \frac{1}{|T_f|}\sum_i^{T_f}\left|\frac{\partial^2 u^*}{\partial t_i^2} - a^2\Delta u^*\right|^2$$

$$\mathscr{L}_b(w;b;T_b) = \frac{1}{|T_b|}\sum_i^{T_b}\left(|u^*(x,t_i)|_{x=0}|^2 + |u^*(x,t_i)|_{x=l}|^2\right)$$

$$\mathscr{L}_i(w;b;T_i) = \frac{1}{|T_i|}\sum_i^{T_i}\left(\left(|u^*(x_i,t)|_{t=0} - A(x_i)|^2\right) + \left(\left|\frac{\partial u^*(x_i,t)}{\partial t}\right|_{t=0} - B(x_i)\right|^2\right)\right)$$

Build a training model, where network optimizer, learning rate, iteration number, and other parameters are defined. The loss function is compiled into the network and iteratively trained. If the network training loss function $\mathscr{L}(w;b;T)$ is less than a specific value, the training ends and the loop jumps out, indicating that the characteristics of the solution have been learned roughly. If $\mathscr{L}(w;b;T)$ is greater than a specific value, the network loops back and iteratively updates the parameter again until it converges or reaches a predetermined number of iterations.

The essence of solving PDEs by PINNs is that the loss function $\mathscr{L}(w;b;T)$ is constantly optimized and the network parameters w and b are constantly updated. Thus, getting suitable parameters(w^*; b^*), and minimizing $\mathscr{L}(w;b;T)$, which can be expressed as $(w^*;b^*) = \arg\min_{(w;b)} \mathscr{L}(w;b;T)$.

3.2. Gradient-enhanced physics-informed neural networks (gPINNs)

gPINNs improve loss function terms based on PINNs. gPINNs codes the partial derivative of the independent variable of the equation into the calculation of the loss function as a constraint term. Adding more constraint terms of the equation, more PDEs physical rules are learned by the network, and the real solution of the equation is better approximated. In PINNs, the difference in the solution between the predicted network and reality is 0, and naturally, the difference in the partial derivative of the independent variable of the equation is also 0 in gPINNs.

The partial derivative of the independent variable of the equation can be defined as,

$$\frac{\partial^3 u}{\partial t^2 \partial x} - a^2\frac{\partial^3 u}{\partial x^3} = \frac{\partial f(x,t)}{\partial x} \tag{8}$$

$$\frac{\partial^3 u}{\partial t^3} - a^2\frac{\partial^3 u}{\partial x^2 \partial t} = \frac{\partial f(x,t)}{\partial t} \tag{9}$$

Eqs. (8) and (9) are the partial derivatives of the independent variables x and t, respectively. The above two expressions are designed as loss terms,

$$\mathscr{L}_{gx}(w;b;T_g) = w_{gx}\frac{1}{T_g}\sum_i^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^2 \partial x_i} - a^2\frac{\partial^3 u^*}{\partial x_i^3} - \frac{\partial f}{\partial x_i}\right|^2 \tag{10}$$

$$\mathscr{L}_{gt}(w;b;T_g) = w_{gt}\frac{1}{T_g}\sum_i^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^3} - a^2\frac{\partial^3 u^*}{\partial x_i^2 \partial t_i} - \frac{\partial f}{\partial t_i}\right|^2 \tag{11}$$

where, $\mathscr{L}_{gx}(w;b;T_g)$ is the partial derivative loss term of the independent variable x, $\mathscr{L}_{gt}(w;b;T_g)$ is the partial derivative loss term of the independent variable t, w_{gx},w_{gt} are the corresponding weights, and T_g is the residual handicap, respectively. gPINNs add Eqs. (10) and (11) to Step 4 in Table 1, which is the improvement compared to PINNs. This time, the loss function of gPINNs,

$$\begin{aligned}\mathscr{L}(w;b;T) = \ & w_f\mathscr{L}_f(w;b;T_f) + w_b\mathscr{L}_b(w;b;T_b) + w_i\mathscr{L}_i(w;b;T_i) \\ & + w_{gx}\frac{1}{T_g}\sum_i^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^2\partial x_i} - a^2\frac{\partial^3 u^*}{\partial x_i^3} - \frac{\partial f}{\partial x_i}\right|^2 \\ & + w_{gt}\frac{1}{T_g}\sum_i^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^3} - a^2\frac{\partial^3 u^*}{\partial x_i^2\partial t_i} - \frac{\partial f}{\partial t_i}\right|^2\end{aligned} \tag{12}$$

In the next chapter, PINNs and gPINNs methods are used to solve the wave equation numerically, and the applicability of these two methods is analyzed.

4. gPINNs method of the wave equation

In this chapter, PINNs and gPINNs are employed to solve the wave equation, and the boundary hard constraint method proposed in [30] is adopted in the solve process. A network with a hidden layer of 4 × 50 was used, which was first optimized 5000 times by the Adam [38] optimizer, followed by further optimization using l-BFGS [39].In both algorithms, the loss weight $w = 1$ is assumed.

Analytical solution of Eq. (13),

$$\begin{aligned}u(x,y,t) = & \sum_{k=0}^{\infty}\sum_{l=0}^{\infty}\frac{64}{\pi^6(2k+1)^3(2l+1)^3} \\ & \cdot\cos\left[\sqrt{(2k+1)^2+(2l+1)^2}t\right]\cdot\sin[(2k+1)\pi x]\cdot\cos[(2l+1)\pi y]\end{aligned} \tag{14}$$

PINNs and gPINNs were employed to solve this equation, respectively. PINNs' loss function is designed as,

$$\begin{aligned}\mathscr{L}(w;b;T) &= \mathscr{L}_f(w;b;T_f) + \mathscr{L}_i(w;b;T_i) \\ &= \frac{1}{|T_f|}\sum_{i=1}^{T_f}\left|\frac{\partial^2 u^*}{\partial t_i^2} - a^2 * \left(\frac{\partial^2 u^*}{\partial x_i^2} + \frac{\partial^2 u^*}{\partial y_i^2}\right)\right|^2 \\ &\quad + \frac{1}{|T_i|}\sum_i^{T_i}\left(|u^*(x_i,0) - x_i(x_i-1)y_i(y_i-1)|^2 + \left|\frac{\partial u^*(x_i,t)}{\partial t}|_{t=0}\right|^2\right)\end{aligned} \tag{15}$$

gPINNs' loss function is designed as,

$$\begin{aligned}L_g(w;b;T) &= L_f(w;b;T_f) + L_i(w;b;T_i) + Lg_x(w;b;T_g) + Lg_y(w;b;T_g) + L_{gt}(w;b;T_g) \\ &= \frac{1}{|T_f|}\sum_{i=1}^{T_f}\left|\frac{\partial^2 u^*}{\partial t_i^{\,2}} - a^2 * \left(\frac{\partial^2 u^*}{\partial x_i^{\,2}} + \frac{\partial^2 u^*}{\partial y_i^{\,2}}\right)\right|^2 \\ &\quad + \frac{1}{|T_i|}\sum_i^{T_i}\left(\left|u^*(x_i,0) - x_i(x_i-1)y_i(y_i-1)\right|^2 + \left|\frac{\partial u^*(x_i,t)}{\partial t}\right|_{t=0}\right|^2\right) \\ &\quad + \frac{1}{|T_g|}\sum_{i=1}^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^{\,2}\partial x_i} - a^2 * \left(\frac{\partial^3 u^*}{\partial x_i^{\,3}} + \frac{\partial^3 u^*}{\partial y_i^{\,2}\partial x_i}\right)\right|^2 \\ &\quad + \frac{1}{|T_g|}\sum_{i=1}^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^{\,2}\partial y_i} - a^2 * \left(\frac{\partial^3 u^*}{\partial x_i^{\,2}\partial y_i} + \frac{\partial^3 u^*}{\partial y_i^{\,3}}\right)\right|^2 \\ &\quad + \frac{1}{|T_g|}\sum_{i=1}^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^{\,3}} - a^2 * \left(\frac{\partial^3 u^*}{\partial x_i^{\,2}\partial t_i} + \frac{\partial^3 u^*}{\partial y_i^{\,2}\partial t_i}\right)\right|^2\end{aligned} \tag{16}$$

4.1. Example 1

This section considers two-dimensional wave equations with Dirichlet boundary conditions. Suppose the variable $x, y \in [0, 1]$ on Ω,$t \in (0, 1]$, and the wave speed is $a = 1/\pi$. In the initial state, the displacement $A(x) = x(x-1)y(y-1)$is the velocity $B(x) = 0$. The values on the boundary are 0 everywhere, and the expression can be obtained as follows,

$$\begin{aligned}&\frac{\partial^2 u}{\partial t^2} = a^2 * \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ &u(x,y,0) = x(x-1)y(y-1) \\ &\left.\frac{\partial u(x,y,t)}{\partial t}\right|_{t=0} = 0 \\ &u(0,y,t) = 0, \quad u(1,y,t) = 0 \\ &u(x,0,t) = 0, \quad u(x,1,t) = 0\end{aligned} \tag{13}$$

The L_2 relative errors predicted by PINNs and g PINNs are obtained at different sampling points, as shown in Fig. 1. The pink line is the L_2 relative errors of gPINNs, and the blue line is the L_2 relative errors of PINNs. As can be seen from Fig. 1, the L_2 relative errors predicted by gPINNs are always less than PINNs in the same sampling points state, regardless of the number of sampling points.

When (x,y)=(1/2,1/2), the residual points are taken with 200 points in the time-space and 100 points in the initial state. PINNs and gPINNs are employed to calculate the changing value of u over time t, and the results are shown in Fig. 2. The pink line represents the result predicted by gPINNs, and the blue line represents the PINNs. The black dashed lines represent the real solution of the equation. The predicted value of gPINNs fits perfectly with the U-value image of the real solution of the equation, while PINNs do not.

In addition, taking the time t=[0.25, 0.5, 0.75], PINNs and gPINNs are applied to calculate the solution value, and compared with the true solution of the equation, respectively. The difference is plotted in Fig. 3.

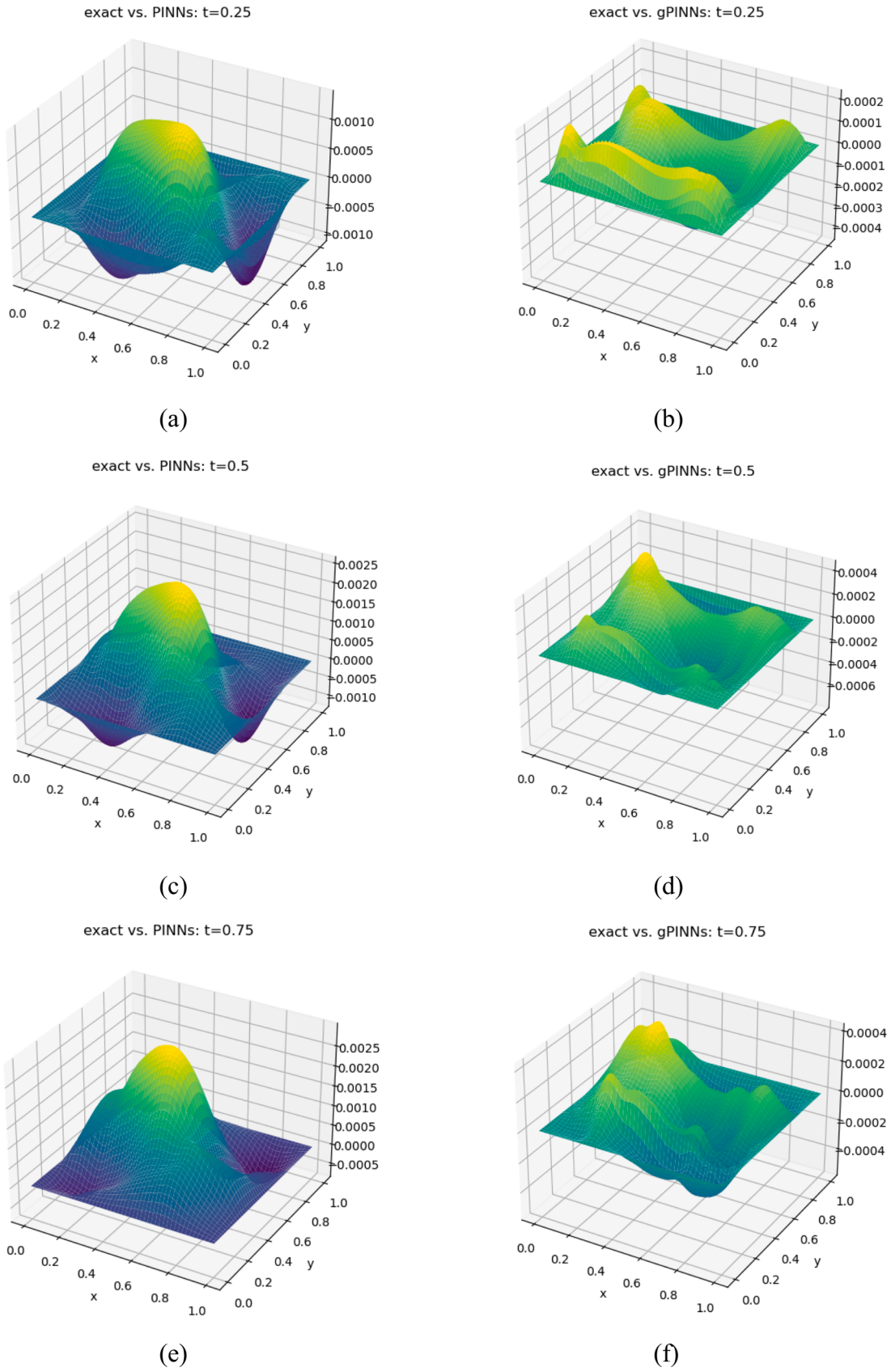


Fig. 3. t=[0.25, 0.5, 0.75], the difference by applying PINNs and gPINNs. (a),(c) and (e) are differences obtained by applying PINNs. (b),(d) and (f) are differences obtained by applying gPINNs.

Fig. 3(a), Fig. 3(c), and Fig. 3(e) show the PINNs difference between the real solution and the predicted solution, and Fig. 3(b), Fig. 3(d), and Fig. 3. (f) show the gPINNs difference. As can be seen from Fig. 3, the difference of gPINNs is smaller compared to PINNs at any time, and its accuracy is also an order of magnitude better than PINNs.

We compare training losses for PINNs and gPINNs, as shown in Fig. 4. In addition, we obtain the value of each loss function, as shown in Fig. 5. Fig. 4 shows the training losses of PINNs and gPINNs, with the horizontal axis representing the number of training and the vertical axis representing the loss value. Fig. 5(a) shows the common loss for PINNs and gPINNs, and Fig. 5(b) shows the change for each loss. Fig. 5(a) shows that there is no significant difference between the loss values of the same loss term for PINNs and gPINNs. Fig. 5(b) shows that the values of the three loss terms introduced by gPINNs are relatively large.

From the training loss image, the loss function of gPINNs is relatively large. This is because PINNs mainly consider the in-domain and initial conditions, while gPINNs add three loss function terms on this basis. Specifically, compared with PINNs, gPINNs have three more loss functions related to the partial derivative of the independent variable. This means that gPINNs consider the derivative information of the solution more fully during the calculation of the loss function. Since these derivative terms are higher order and more complex functions, their loss value will be relatively large during training. The loss calculation method of PINNs and gPINNs is to add all the losses, so the addition of complex derivative terms will lead to an increase in the overall loss function value. Even if the other loss terms have the same value, the value of the overall loss function will be larger due to the presence of additional derivative terms.

However, although the loss of gPINNs is relatively large, it does not mean that the model performance will degrade. Rather, it reflects that gPINNs take derivative information into account more comprehensively when solving differential equation problems, thus helping to improve the accuracy and generalization of the model. Fig. 1, Fig. 2, and Fig. 3 show that the training performance of gPINNs is better. Although the loss function is larger, gPINNs produce more accurate and reliable results.

We compare the difference in training computational cost between PINNs and gPINNs. The training computational cost of PINNs and gPINNs is mainly reflected in the construction and optimization of the loss function.

For the design of the loss function, PINNs mainly consider the intra-domain and initial conditions. On this basis, gPINNs add three loss function terms, as shown in Eq. (15) and Eq. (16). The additional loss term of gPINNs is shown in Eq. (17). These loss terms are associated with the derivatives of independent variables such as spatial coordinates and time, taking into account the rate of change of the solution in the space of the independent variable. Therefore, the loss function of gPINNs contains more terms and needs to calculate more derivatives. gPINNs added losses as follows.

$$\begin{aligned}
&\frac{\partial^3 u^*}{\partial t^2 \partial x} - a^2 * \left(\frac{\partial^3 u^*}{\partial x^3} + \frac{\partial^3 u^*}{\partial y^2 \partial x}\right) \\
&\frac{\partial^3 u^*}{\partial t^2 \partial y} - a^2 * \left(\frac{\partial^3 u^*}{\partial x^2 \partial y} + \frac{\partial^3 u^*}{\partial y^3}\right) \\
&\frac{\partial^3 u^*}{\partial t^3} - a^2 * \left(\frac{\partial^3 u^*}{\partial x^2 \partial t} + \frac{\partial^3 u^*}{\partial y^2 \partial t}\right)
\end{aligned} \tag{17}$$

The three loss terms added by gPINNs require more derivative calculations. In order to obtain the partial derivative loss term

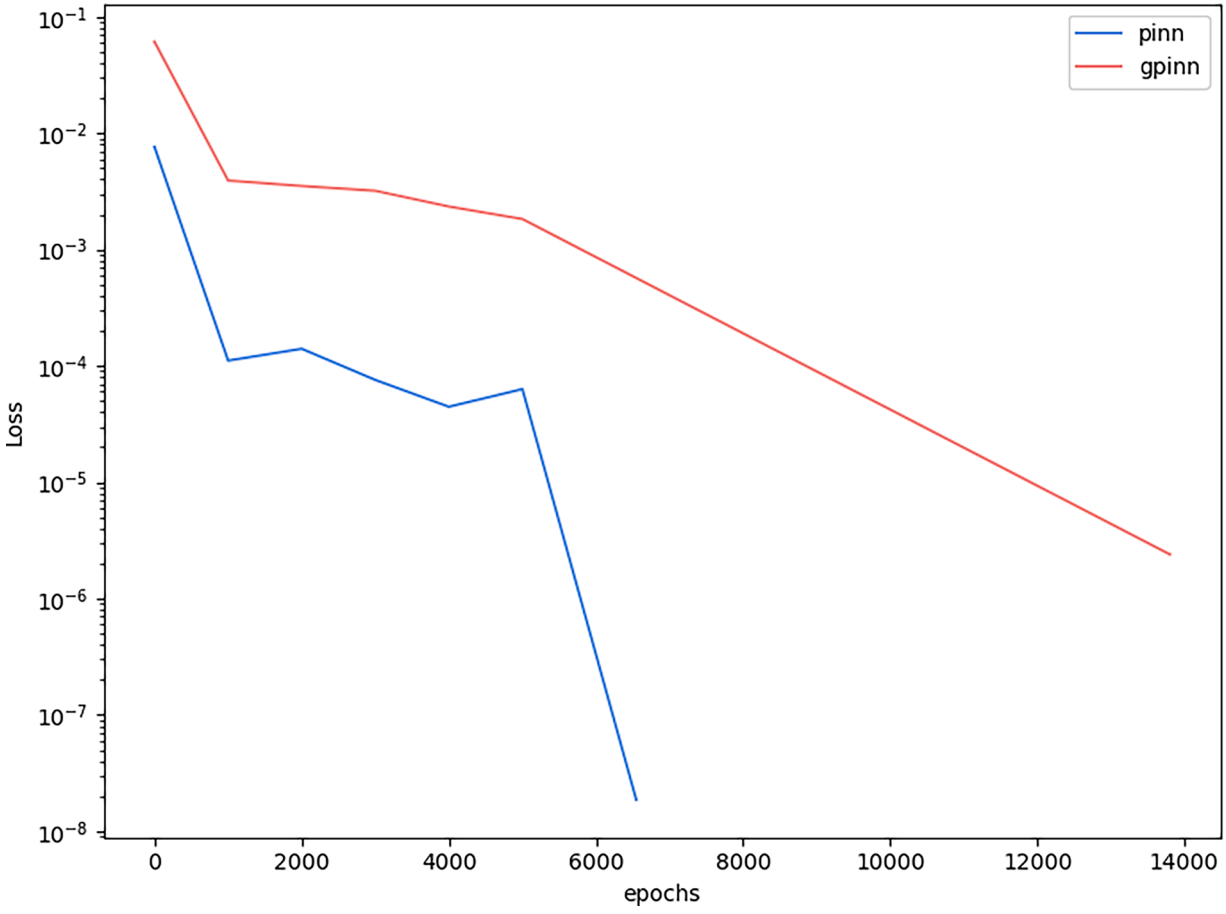


Fig. 4. Training loss of PINNs and gPINNs at different sampling points.

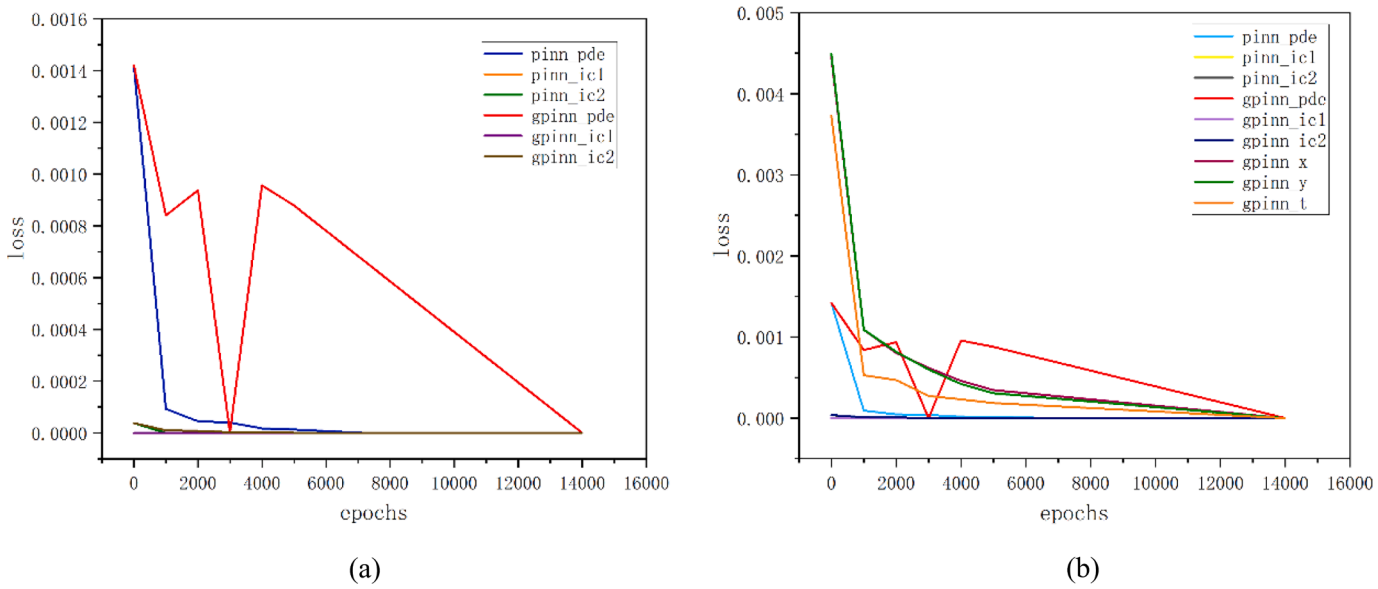


Fig. 5. Each loss of PINNs and gPINNs.

$\frac{\partial^3 u^*}{\partial t^2 \partial x} - a^2 * \left(\frac{\partial^3 u^*}{\partial x^3} + \frac{\partial^3 u^*}{\partial y^2 \partial x}\right)$ of the independent variable x, three automatic differential calculations are needed to obtain $\frac{\partial^3 u^*}{\partial t^2 \partial x}$, $\frac{\partial^3 u^*}{\partial x^3}$, and $\frac{\partial^3 u^*}{\partial y^2 \partial x}$. The same is true for the independent variables y and t.

In the optimization process, PINNs only need to calculate the residual difference between the model prediction and the real physical equation, and then update the network parameters through the back-propagation algorithm. Such calculations are relatively inexpensive because only the model's predicted value is compared to the true value. The computational cost of gPINNs is increased compared to PINNs. Three automatic differential calculations are required for each

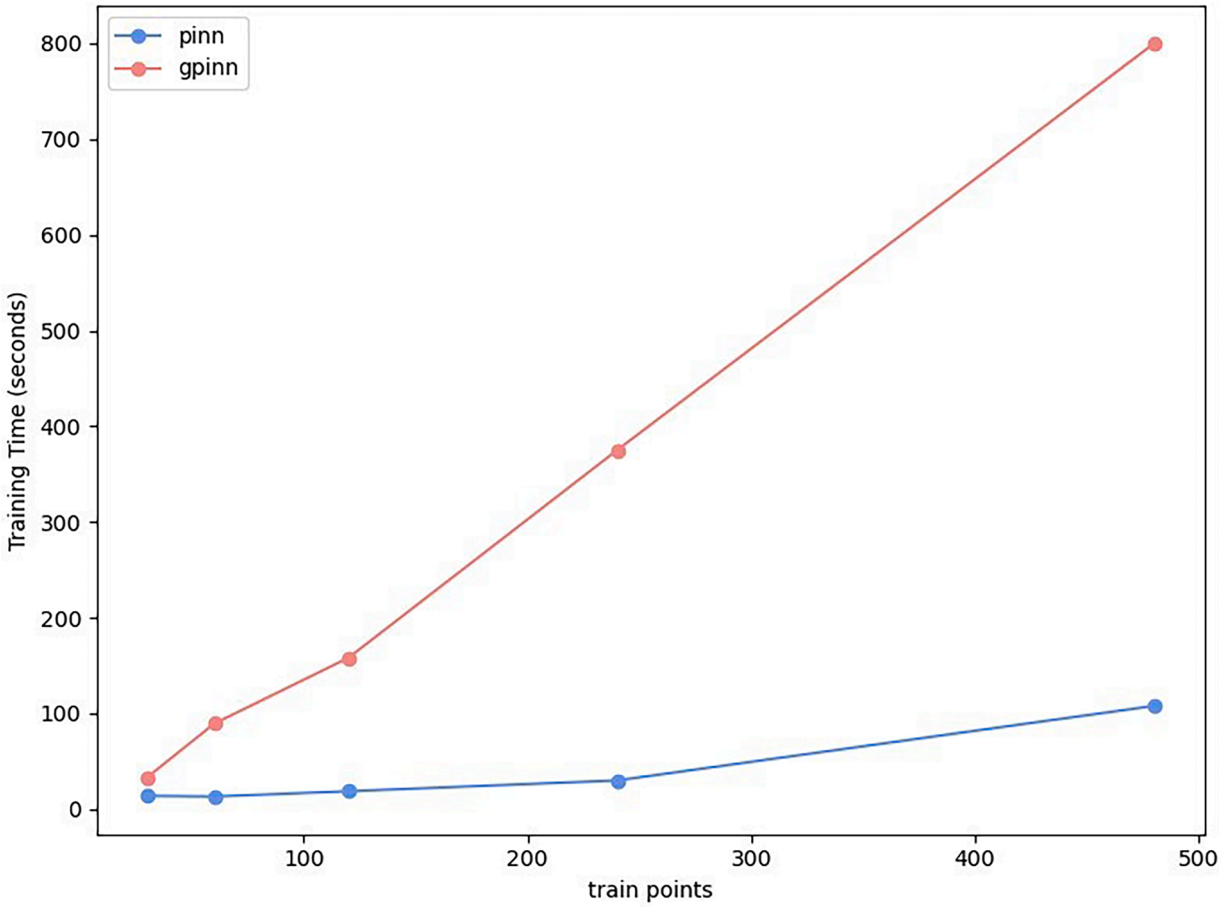


Fig. 6. Training time of PINNs and gPINNs at different sampling points.

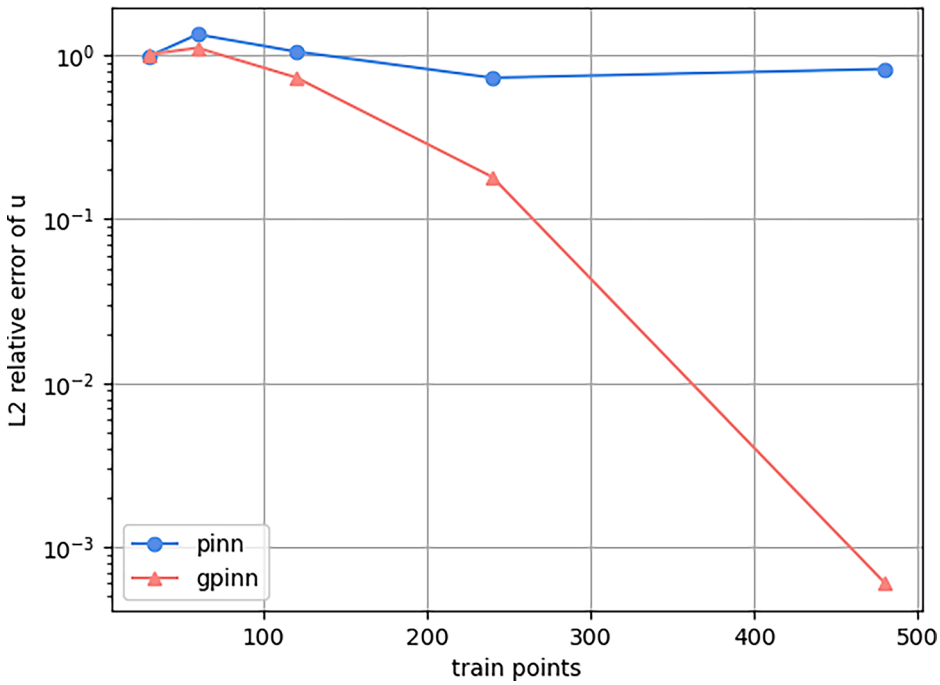


Fig. 7. L2 relative errors by PINNs and gPINNs at different sampling points.

additional loss and nine additional times for three losses. In each iteration of training, these loss terms need to be additionally computed, which will increase the training time and computational cost.

Fig. 6 shows the comparison of PINNs and gPINNs training times at different sampling points. The horizontal coordinate represents the number of sampling points, and the vertical coordinate represents the training time of the two methods under different sampling points. The image shows that gPINNs spend more training time than PINNs. Because of this, the accuracy of gPINNs prediction is higher than that of PINNs. For example, at 300 sampling points, the predicted result of gPINNs is closer to the actual solution than PINNs, as shown in Figs. 2 and Fig.3.

4.2. Example 2

Another wave equation is considered as follows,

$$\begin{aligned}
&\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2},\\
&u(x,y,0) = \sin(\pi x)\sin(\pi y),\\
&\left.\frac{\partial u(x,y,t)}{\partial t}\right|_{t=0} = 0\\
&u(0,y,t) = 0, \quad u(1,y,t) = 0,\\
&u(x,0,t) = 0, \quad u(x,1,t) = 0
\end{aligned} \tag{18}$$

Assuming the values of x, y, and t are all [0,1], and the wave speed a is set to 1.0. The initial displacement is set tosin(πx)sin(πy), and the initial velocity is set to 0. The values on the boundary are 0 everywhere. PINNs and gPINNs are employed to solve the above wave equation, respectively.

Analytical solution of Eq. (18),

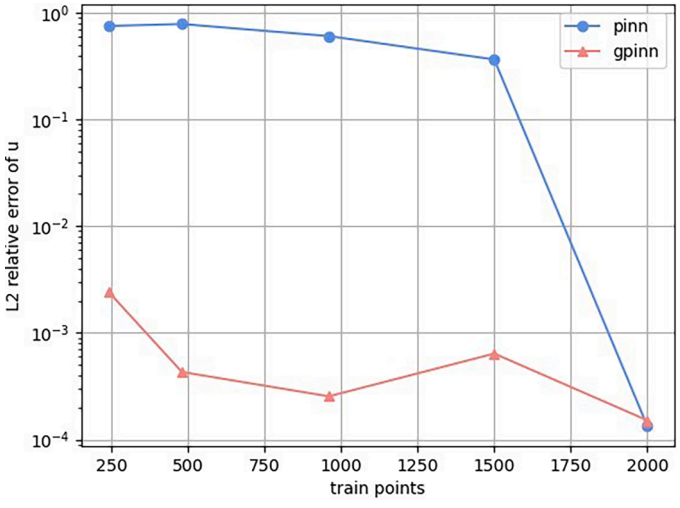


Fig. 8. L_2 relative errors by PINNs and gPINNs as the sampling points are increased.

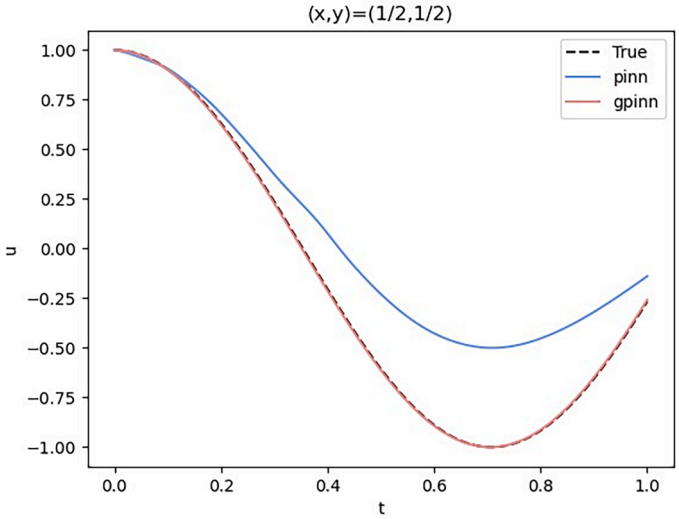


Fig. 9. (x,y)=(1/2,1/2), the change of u by PINNs and gPINNs.

$$u(x,y,t) = \cos\left(\sqrt{2}\pi t\right)\cdot\sin(\pi x)\cdot\sin(\pi y) \tag{19}$$

And the loss function of PINNs is defined as,

$$\begin{aligned}\mathscr{L}(w;b;T) &= \mathscr{L}_f(w;b;T_f) + \mathscr{L}_i(w;b;T_i)\\ &= \frac{1}{|T_f|}\sum_{i=i}^{T_f}\left|\frac{\partial^2 u^*}{\partial t_i^2} - \left(\frac{\partial^2 u^*}{\partial x_i^2} + \frac{\partial^2 u^*}{\partial y_i^2}\right)\right|^2\\ &+ \frac{1}{|T_i|}\sum_{i}^{T_i}\left(|u^*(x_i,0) - \sin(\pi x_i)\sin(\pi y_i)|^2 + \left|\frac{\partial u^*(x_i,t)}{\partial t}\right|_{t=0}\right|^2\end{aligned} \tag{20}$$

while the loss function of gPINNs is defined as,

$$\begin{aligned}\mathrm{L}_g(w;b;T) &= \mathrm{L}_f(w;b;T_f) + \mathrm{L}_i(w;b;T_i) + \mathrm{L}_{gx}(w;b;T_g) + \mathrm{L}_{gy}(w;b;T_g) + \mathrm{L}_{gt}(w;b;T_g)\\ &= \frac{1}{|T_f|}\sum_{i=i}^{T_f}\left|\frac{\partial^2 u^*}{\partial t_i^{\,2}} - \left(\frac{\partial^2 u^*}{\partial x_i^{\,2}} + \frac{\partial^2 u^*}{\partial y_i^{\,2}}\right)\right|^2\\ &+ \frac{1}{|T_i|}\sum_{i}^{T_i}\left(\left|u^*(x_i,0) - \sin(\pi x_i)\sin(\pi y_i)\right|^2 + \left|\frac{\partial u^*(x_i,t)}{\partial t}\right|_{t=0}\right|^2\\ &+ \frac{1}{|T_g|}\sum_{i=i}^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^{\,2}\partial x_i} - \left(\frac{\partial^3 u^*}{\partial x_i^{\,3}} + \frac{\partial^3 u^*}{\partial y_i^{\,2}\partial x_i}\right)\right|^2\\ &+ \frac{1}{|T_g|}\sum_{i=i}^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^{\,2}\partial y_i} - \left(\frac{\partial^3 u^*}{\partial x_i^{\,2}\partial y_i} + \frac{\partial^3 u^*}{\partial y_i^{\,3}}\right)\right|^2\\ &+ \frac{1}{|T_g|}\sum_{i=i}^{T_g}\left|\frac{\partial^3 u^*}{\partial t_i^{\,3}} - \left(\frac{\partial^3 u^*}{\partial x_i^{\,2}\partial t_i} + \frac{\partial^3 u^*}{\partial y_i^{\,2}\partial t_i}\right)\right|^2\end{aligned} \tag{21}$$

Similarly, Fig. 7 shows the L_2 relative errors for PINNs and gPINNs at different sampling points. The horizontal coordinate is different sampling points, and the vertical coordinate corresponds to L_2 relative errors. The pink line is gPINNs, and the blue line is PINNs. As can be seen from Fig. 7, there is a small difference in L_2 relative errors between the two methods when the number of sampling points is small. The L_2 relative errors calculated by gPINNs began to decrease rapidly with the increase of sampling points. When the sampling points are increased from 120 to 240 points, the L_2 relative errors calculated by gPINNs rapidly decrease by two orders of magnitude, while the PINNs drop only slightly.

To further verify the accuracy of the above conclusions, more sampling points were selected. Observing Fig. 8 can be found that the L_2 relative errors calculated by PINNs begin to decrease gradually as sampling points are taken. Combined with Fig. 7 and Fig. 8, the L_2 relative errors calculated by PINNs at 2000 sampling points (Fig. 8) are close to that calculated by gPINNs at 300 sampling points (Fig. 7). gPINNs can achieve a similar accuracy with a small number of sampling points as PINNs with a large number of sampling points.

Similarly, when (x,y)=(1/2, 1/2), PINNs and gPINNs are employed to compute the wave equation, accompanied by taking 200 sample points in the time-space domain and 100 sample points for the initial state. The results predicted by gPINNs almost fit the true solution, while PINNs do not, as can be seen from Fig. 9. Now, the sampling points of PINNs have increased to 2000, while gPINNs remain unchanged. At this time, the wave equation continues to be calculated with two methods, as shown in Fig. 10. PINNs can also fit the real solution well. The prediction solution obtained by gPINNs at 300 sampling points is similar to that obtained by PINNs at 2000 sampling points, and it can be concluded that gPINNs have a stronger ability to learn data features.

In the same way, taking the time t=[0.25, 0.5, 0.75], the difference between the real solution and the predicted solution is obtained by applying PINNs and gPINNs, as shown in Fig. 11. Fig. 11(a), Fig. 11(d) and Fig. 11(g) show the PINNs difference between the true and predicted solutions obtained at 300 sample points, and Fig. 11(b), Fig. 11(e) and Fig. 11(h) show the PINNs difference at 2000 sample points, respectively. Fig. 11(c), Fig. 11(f), and Fig. 11(i) show the gPINNs difference between the true and predicted solutions obtained at 300 sample points.

Unsurprisingly, at 300 sampling points, a large difference is calculated by applying PINNs, which is not predictive of the equation, and the error expanded by two orders of magnitude compared to the other two results. The prediction error is greatly reduced when the sampling points are

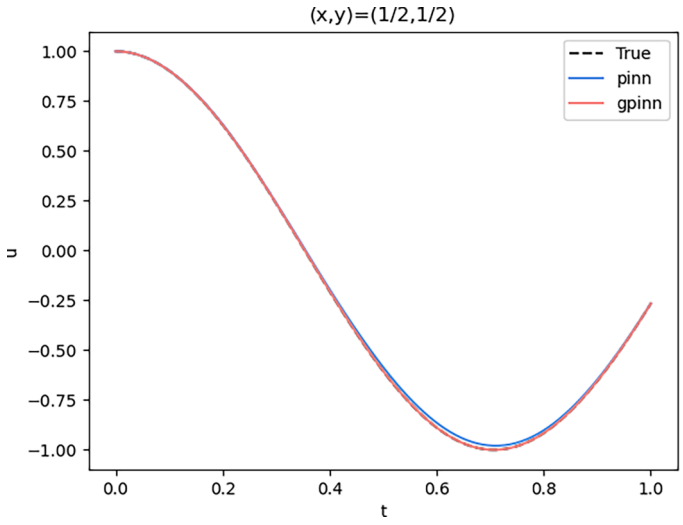


Fig. 10. (x,y)=(1/2,1/2), the change of u by PINNs and gPINNs, and only the sample points of PINNs are increased to 2000.

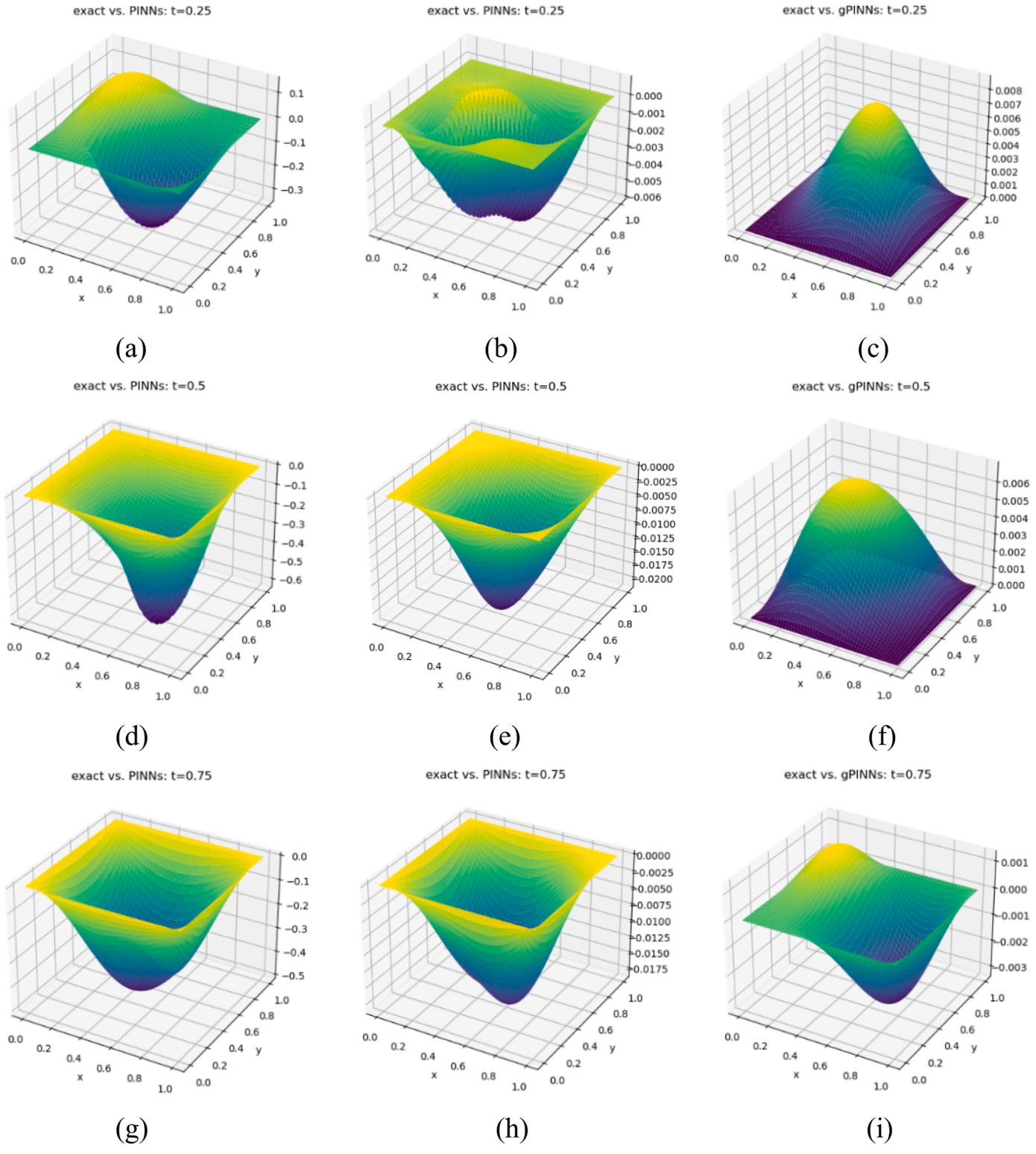


Fig. 11. t=[0.25, 0.5, 0.75], the difference by applying PINNs and gPINNs. (a),(d) and (g) are the difference obtained by applying PINNs at 300 sampling points. (b), (e) and (h) are the difference obtained by applying gPINNs at 2000 sampling points. (c), (f) and (i) are the difference obtained by applying gPINNs at 300 sampling points.

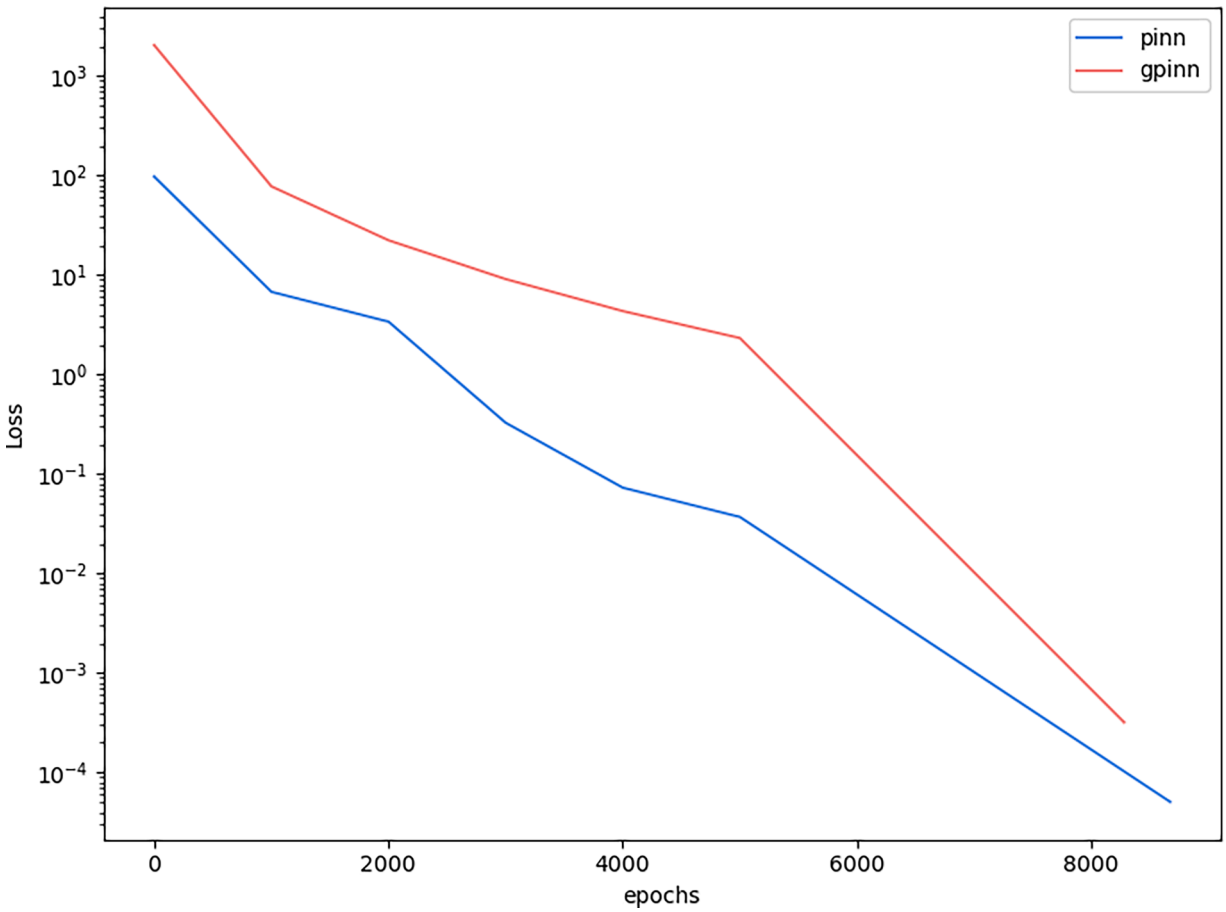


Fig. 12. Training loss of PINNs and gPINNs at different sampling points.

increased to 2000 for PINNs. In Fig. 11(b) and Fig. 11(c), the error is similar when calculated by PINNs at 2000 sampling points and gPINNs at 300 sampling points and the accuracy is similar when gPINNs are achieved with few sampling points and PINNs are achieved with many sampling points. gPINNs is better on the whole, which can be obtained by comparing Fig. 11(e), Fig. 11(f) and Fig. 11(h), Fig. 11(i).

Similarly, we compare training losses for PINNs and gPINNs, as shown in Fig. 12 and Fig.14. Fig. 13 and Fig.15 show the common and total losses of PINNs and gPINNs.Fig. 12 shows the training losses of PINNs and gPINNS at 300 sampling points. Fig.13(a) and Fig.13(b) show the common and total losses, respectively, and Fig.13(c) and Fig.13(d) are partial enlargements of (a) and (b). Fig. 14 shows the loss function when the sampling point of PINNs is increased to 2000 while gPINNs remain unchanged. Fig. 15(a) and Fig. 15(b) show the common and total losses of PINNs and gPINNs, and Fig. 15(c) and Fig. 15(d) are partial enlargements of (a) and (b).

Consistent with the result of example 1, the loss function of gPINNs is relatively large from the loss function image. This is because gPINNs added three complex terms to the loss function, increasing the overall loss function value. However, this does not affect the performance of gPINNs. As shown in Fig.9, Fig.10, and Fig.11, gPINNs predicted better than PINNs.

We compare the difference in training computational cost between PINNs and gPINNs. As described in Section 4.1, the gPINNs method considers more derivative information in the loss function construction, which increases the construction work of the loss function, and the training time also increases, as shown in Fig. 16. The image shows that gPINNs takes more time than PINNs at different sampling points. However, experiments show that gPINNs perform better than PINNs. For example, in the case of 300 sampling points, the training time of gPINNs is longer than that of PINNs, while the prediction results are more accurate, as shown in Fig. 9. Fig. 10 shows that PINNs can achieve similar accuracy to gPINNs by increasing the number of sampling points, but at this time, we find that the time spent by PINNs also increases to be similar to gPINNs. In summary, compared to PINNs, gPINNs invest more in loss function construction and training time. These additional cost come with a higher level of model performance.

4.3. *Discuss the gPINNs method effectively*

PINNs is a machine learning model that combines deep learning and physics.gPINNs optimizes and improves the model, showing a more comprehensive consideration and higher performance. gPINNs has advantages compared to pinn, but it also has limitations.

In terms of the advantages of gPINNs compared with PINNs, firstly, in the design of the loss function, gPINNs consider the derivative information of the solution more comprehensively. gPINNs introduce three additional loss terms related to independent variables, especially related to the partial derivative of the independent variable. gPINNs can more fully reflect the rate of change of the solution in the independent variable space, providing more differential information, thus enhancing the model's adaptability to complex problems and improving the

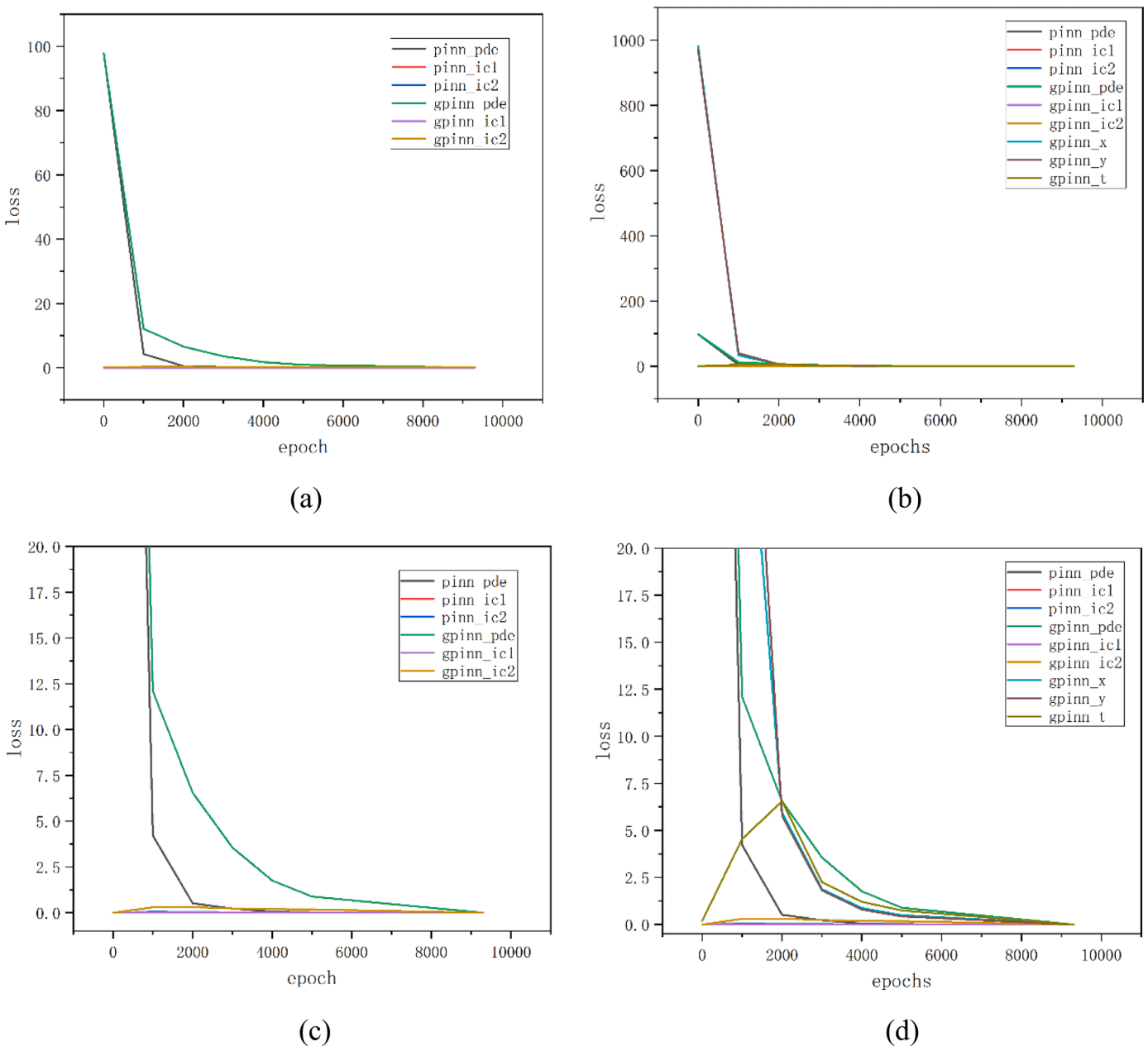


Fig. 13. Each loss of PINNs and gPINNs.

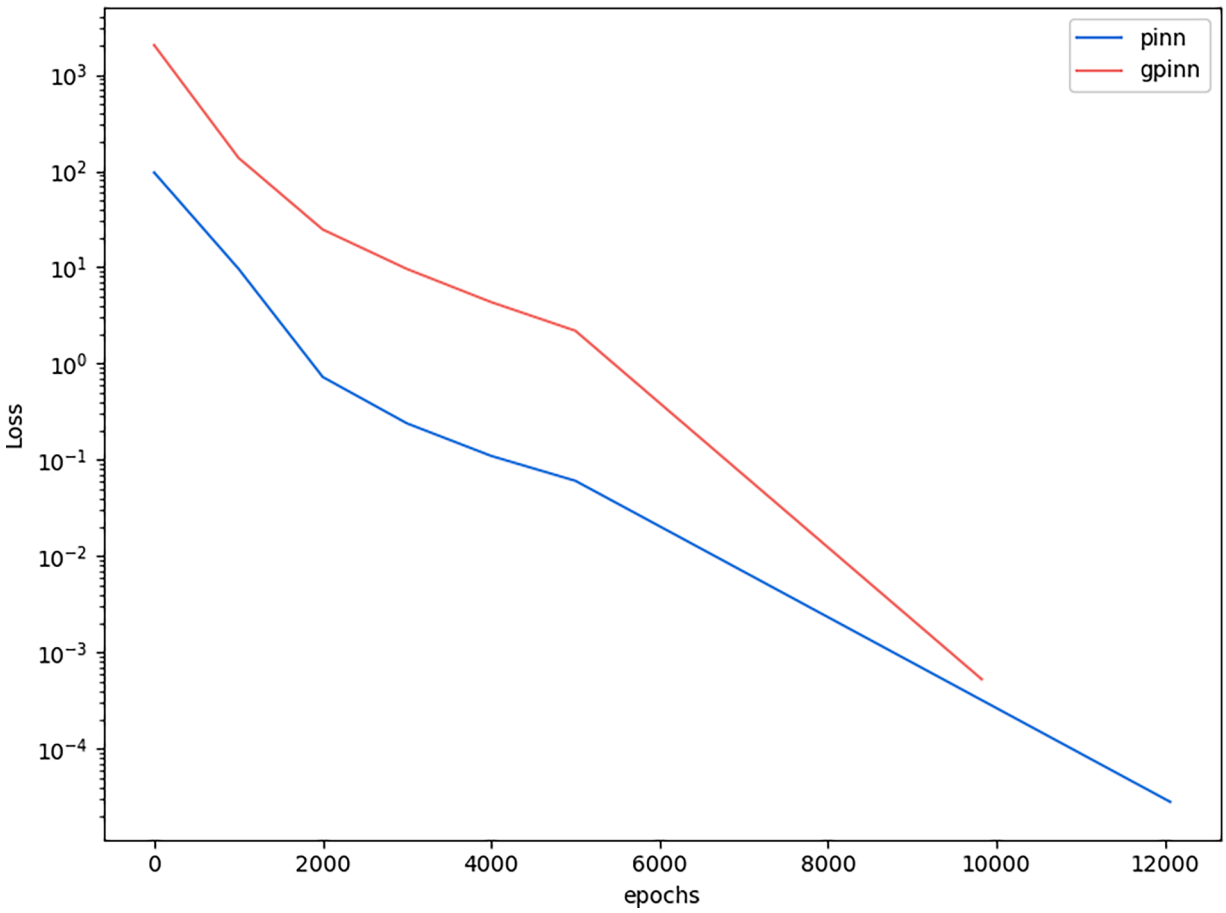


Fig. 14. Training loss of PINNs and gPINNs at different sampling points.

accuracy and generalization ability of training. Secondly, it is worth noting that during the training process, because gPINNs need to perform more derivative calculations, this makes the model more nuanced in the optimization. gPINNs' loss function not only considers the derivative information of the solution but also makes the model more sensitive to small changes through more automatic differential calculations, which is expected to better fit the real physical equation in the training process, thus improving the accuracy of the model. Two examples from Chapter 4 show the higher training accuracy of gPINNs compared to PINNs.

However, the advantages of gPINNs come with some limitations. First, due to the introduction of more derivative terms, the overall loss function value is relatively large. Second, gPINNs spend more training time than PINNs. To overcome these drawbacks, it is meaningful to combine gPINNs with parallel training methods [40,41]. By splitting the model into multiple stages and processing each stage in parallel on multiple devices, the training process of gPINNs can be significantly accelerated while reducing training costs. The introduction of a structured pruning method in gPINNs can significantly shorten the training time and improve the training efficiency of the model. In conclusion, gPINNs have made significant progress in improving the accuracy and generalization ability of models by considering the derivative information of solutions more comprehensively. However, it also faces challenges of longer training time and higher computational cost. In practical applications, it is necessary to weigh the advantages and limitations according to specific problems, to make a more informed decision on whether to adopt gPINNs.

5. Conclusions

In this paper, the gPINNs method is employed to solve the two-dimensional wave equation with boundary hard constraints embedded in the network construction process. Two examples of wave equations are given, which are calculated using PINNs and gPINNs, respectively. The conclusion is that gPINNs is more accurate than PINNs at the same data point, by comparing the L2 relative errors calculated by PINNs and gPINNs using different residuals. With fixed (x,y)=(1/2,1/2) and fixed time t=[0.25, 0.5, 0.75], PINNs and gPINNs are applied to calculate the wave equation, and compared with the real solution of the equation. It is observed that the gPINNs method is better than PINNs under the same conditions. In the second example, assuming 300 sampling points for both algorithms at the beginning, the gPINNs method can fit the equation well, while PINNs cannot predict the equation. Then increase the sample points of PINNs to 2000, and the sample points of gPINNs remain unchanged, PINNs can also fit the equation well. This undoubtedly proves that gPINNs can fit the equation well in the state of fewer data sets, while PINNs need more data sets to fit the equation. The training time of gPINNs is usually longer than PINNs in the calculation process, which may be the reason why the loss term is increased. Reducing the training time of gPINNs can be the content of our follow-up research.

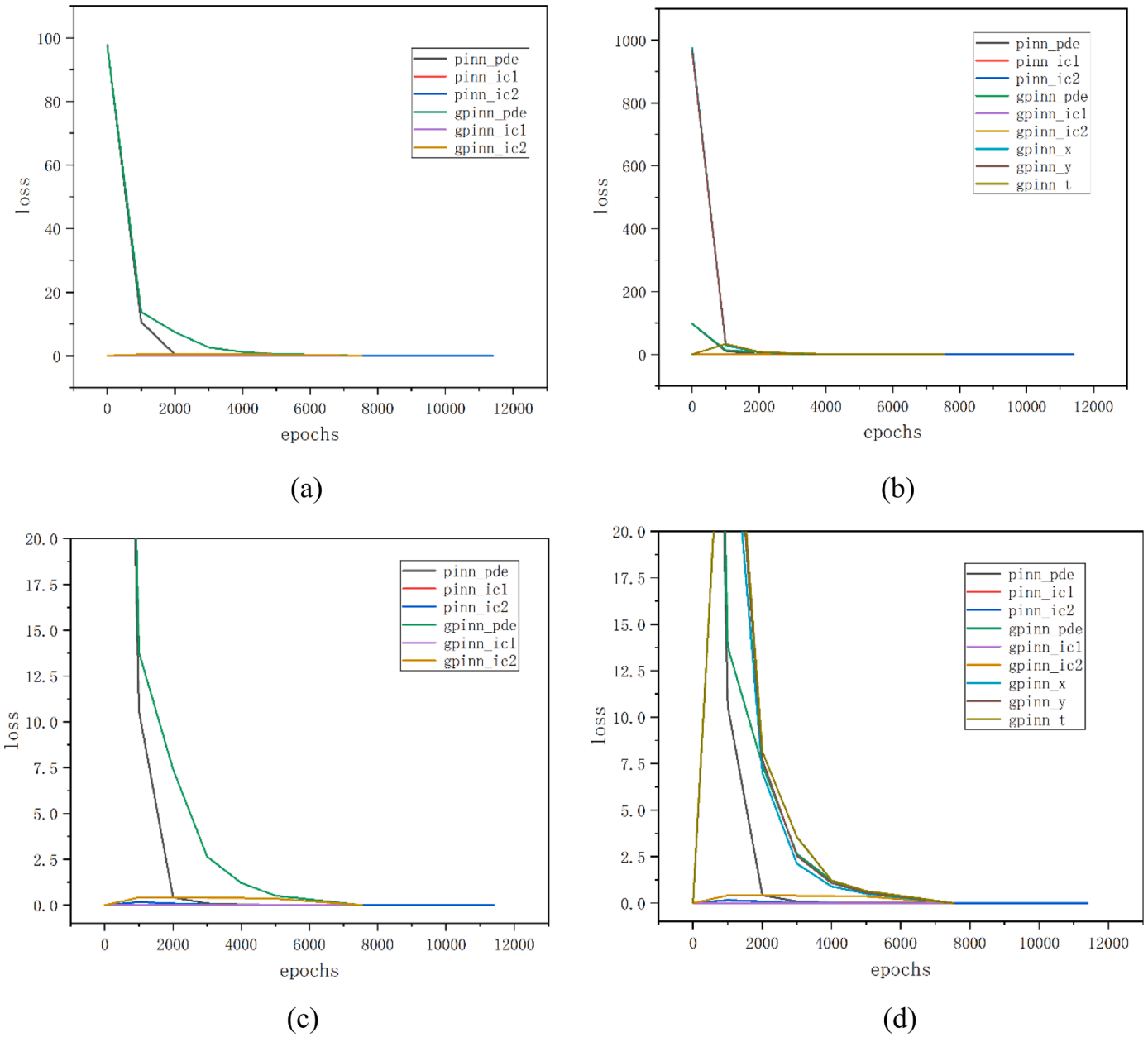


Fig. 15. Each loss of PINNs and gPINNs.

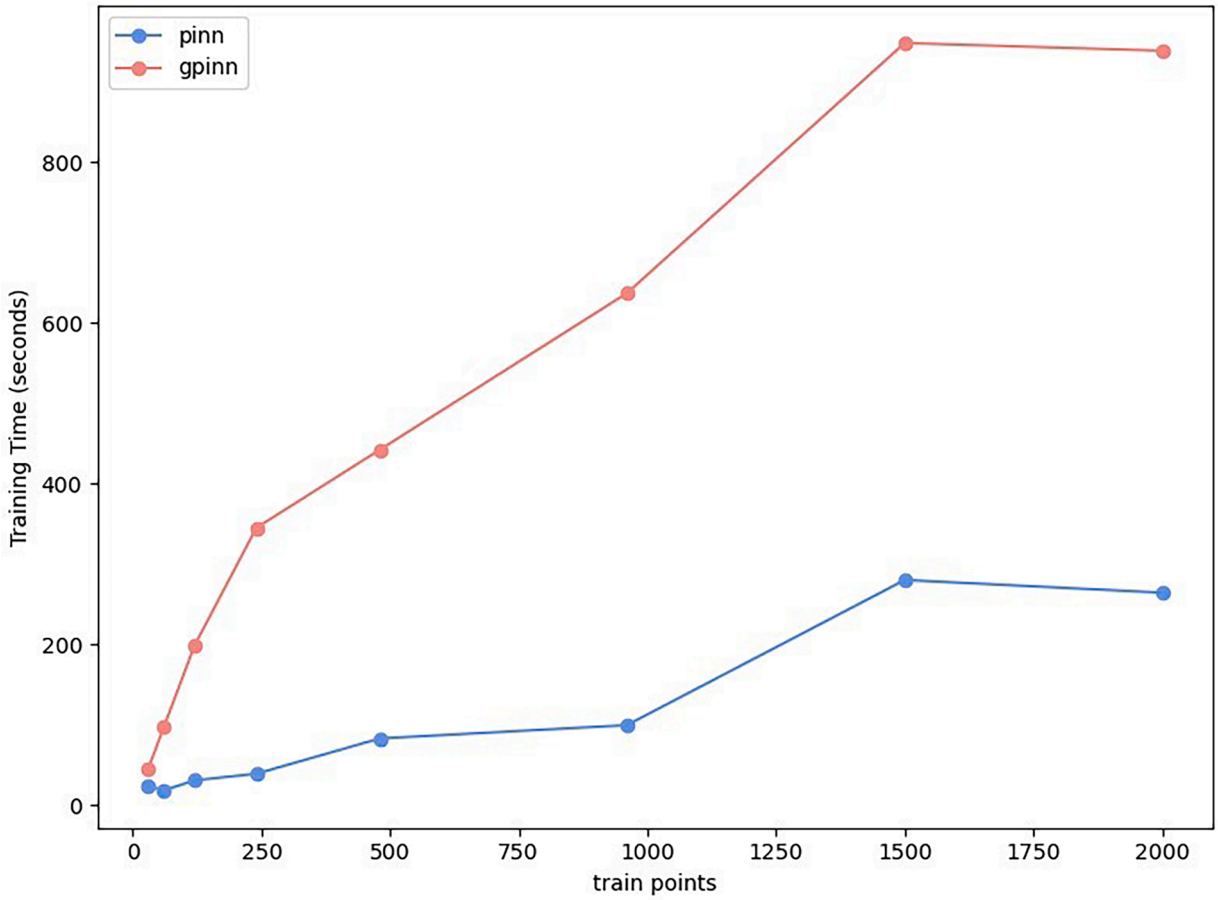


Fig. 16. Training time of PINNs and gPINNs at different sampling points.

CRediT authorship contribution statement

Guizhong Xie: Writing – review & editing, Writing – original draft, Conceptualization. **Beibei Fu:** Writing – review & editing, Writing – original draft, Data curation, Conceptualization. **Hao Li:** Funding acquisition, Formal analysis. **Wenliao Du:** Validation, Supervision, Funding acquisition. **Yudong Zhong:** Project administration, Investigation, Funding acquisition. **Liangwen Wang:** Resources, Project administration, Methodology. **Hongrui Geng:** Visualization, Software, Methodology. **Ji Zhang:** Visualization, Validation, Methodology. **Liang Si:** Software, Resources, Project administration.

Declaration of competing interest

The authors declared no potential conflicts of interest with respect to the research, author- ship, and/or publication of this article.

Data availability

Data will be made available on request.

Acknowledgements

This work was partly supported by the National Science Foundation of China (12302270, 52075500, 52304301 and 52175256), and partly supported by key scientific and technological project of Henan Province (232102221040, 232102221041, and 232102221033), partly supported by Henan Province Science and Technology Research and Development Plan Joint Fund Project (225200810029), and Major Science and Technology Project of Longmen Laboratory (231100220300). We especially thank Shuangqiang Luo for his careful guidance and valuable suggestions in the revision of our paper.

References

[1] Zeng SJ, Zhang Z, Zou QS. Adaptive deep neural networks methods for high-dimensional partial differential equations[J]. J Comput Phys 2022;463:111232.
[2] Shin J, Lee JY. Energy-conserving successive multi-stage method for the linear wave equation with forcing terms[J]. J Comput Phys 2023;489:112255.
[3] Moseley B., Markham A., Nissen-Meyer T. Solving the wave equation with physics-informed deep learning[J]. arXiv preprint arXiv:2006.11894, 2020.
[4] Song C, Alkhalifah T, Waheed UB. Solving the frequency-domain acoustic VTI wave equation using physics-informed neural networks[J]. Geophys J Int 2021;225(2):846–59.
[5] Song C, Wang Y. Simulating seismic multifrequency wavefields with the Fourier feature physics-informed neural network[J]. Geophys J Int 2023;232(3):1503–14.
[6] Sun L, Feng Y, Liu Y, et al. The modified weak galerkin finite element method for solving brinkman equations[J]. J Math Res Appl 2019;39(06):657–76.
[7] Guo CY, Xiao XF, Feng XL, et al. An immersed finite element method for elliptic interface problems on surfaces[J]. Comput Math Appl 2023;131:54. -67.
[8] Yuan YR, Li CF, H.L S. An upwind-block-centered finite difference method for a semiconductor device of heat conduction and its numerical analysis[J]. J Comput Appl Math 2023;425:114978.
[9] Hashemi AS, Heydari M, Loghmani GB. Iterative compact finite difference method for the numerical study of fully wet porous fins with different profile shapes[J]. Appl Numer Math 2023;186:358–77.
[10] Lei WD, Ji DF, Zhu GP. Time-domain boundary element method with von Mises model for solving 2-D elastoplastic dynamic problems[J]. J Braz Soc Mech Sci Eng 2019;41(6):269.
[11] Lei WD, Ji DF, Li HJ, et al. On an analytical method to solve singular integrals both in space and time for 2-D elastodynamics by TD-BEM[J]. Appl Math Model 2015;39(20):6307–18.
[12] Xie GZ, Li K, Zhong YD, et al. A systematic derived sinh based method for singular and nearly singular boundary integrals[J]. Eng Anal Bound Elem 2021;123:147–53.

[13] Xie GZ, Zhong YD, Li H, et al. Near singularity cancellation in weakly singular integrals of three-dimensional boundary element method[J]. Eng Anal Bound Elem 2020;118:49–54.
[14] Zhong YD, Xie GZ, Geng HR, et al. Thermal analysis for plate structures using a transformation BEM based on complex poles[J]. Comput Math Appl 2024;161:32–42.
[15] AMES WF. Numerical methods for partial differential equations[M]. Academic Press; 1992.
[16] Gu Y, Zhang CZ, Zhang PJ, et al. Enriched physics-informed neural networks for 2D in-plane crack analysis: theory and MATLAB code[J]. Int J Solids Struct 2023;276:112321.
[17] Gu Y, Zhang CZ, Golub MV. Physics-informed neural networks for analysis of 2D thin-walled structures[J]. Eng Anal Bound Elem 2022;145:161–72.
[18] Karniadakis GE, Kevrekidis IG, Lu L, et al. Physics-informed machine learning[J]. Nat Rev Phys 2021;3(6):422–40.
[19] Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations[J]. IEEE Trans Neural Networks 1998;9(5):987–1000.
[20] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations[J]. J Comput Phys 2019;378:686–707.
[21] Wang SF, Teng YJ, Perdikaris P. Understanding and mitigating gradient pathologies in physics-informed neural networks[J]. SIAM J Sci Comput 2021;43(5):A3055–81.
[22] R.L. Burden, D.J. Faires, Numerical analysis, 1985.
[23] Li JH, Li B. Gradient-optimized physics-informed neural networks (GOPINNs): a deep learning method for solving the complex modified KdV equation[J]. Nonlinear Dyn 2022;107:781–92.
[24] Fang TS, Li B. Solving complex nonlinear problems based on gradient-optimized physics-informed neural networks[J]. Acta Physica Sinica 2023;72(10):100202.
[25] Li JH, Li B. Mix-training physics-informed neural networks for the rogue waves of nonlinear Schrödinger equation[J]. Chaos, Solitons Fractals 2022;164:112712.
[26] Wu CX, Zhu M, Tan QY, et al. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks[J]. Comput Methods Appl Mech Eng 2023;403:115671.
[27] McClenny LD, Braga-Neto UM. Self-adaptive physics-informed neural networks[J]. J Comput Phys 2023;474:111722.
[28] Yu J, Lu L, Meng XH, et al. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems[J]. Comput Methods Appl Mech Eng 2022;393:114823.
[29] Samaniego E, Anitescu C, Goswami S, et al. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications[J]. Comput Methods Appl Mech Eng 2019;362:112790.
[30] Lu L, Pestourie R, Yao WJ, et al. Physics-informed neural networks with hard constraints for inverse design[J]. SIAM J Sci Comput 2021;43(6):B1105–32.
[31] Zhang ZY, Zhang H, Zhang LS, et al. Enforcing continuous symmetries in physics-informed neural network for solving forward and inverse problems of partial differential equations[J]. J Comput Phys 2023;492:112415.
[32] Zhu W, Khademi W, Charalampidis EG, et al. Neural networks enforcing physical symmetries in nonlinear dynamical lattices: the case example of the Ablowitz–Ladik model[J]. Physica D 2022;434:133264.
[33] Linka K, Schafer A, Meng XH, et al. Bayesian Physics-Informed Neural Networks for real-world nonlinear dynamical systems[J]. Comput Methods Appl Mech Eng 2022;402:115346.
[34] Pang GF, Lu L, Karniadakis EG. fPINNs: fractional Physics-Informed Neural Networks[J]. SIAM J Sci Comput 2019;41(4):A2603–26.
[35] Gu YQ, Yang HZ, Zhou C. SelectNet: self-paced learning for high-dimensional partial differential equations[J]. J Comput Phys 2021;441:110444.
[36] Lu L, Meng XH, Mao ZP, et al. DeepXDE: a deep learning library for solving differential equations[J]. SIAM Rev 2021;63(1):208–28.
[37] Kharazmi E, Zhang ZQ, George Karniadakis EM. hp-VPINNs: variational physics-informed neural networks with domain decomposition[J]. Comput Methods Appl Mech Eng 2021;374:113547.
[38] P.D. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv: 2014, abs/1412.6980.
[39] Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Math Program 1989;45(1):503–28.
[40] Huang Y, Cheng Y, Bapna A, et al. GPipe: easy scaling with micro-batch pipel ine parallelism[J]. In: proceeding of Computer Science>Computer Vision and Pattern Recognition; 2019.
[41] Shoeybi M., Patwary M., Puri R., et al. Megatron-lm: training multi-billion parameter language models using model parallelism[J]. arXiv preprint arXiv:1909.08053, 2019.