AIMS Mathematics, 10(10): 24857–24900.
DOI: 10.3934/math.20251101
Received: 01 August 2025
Revised: 14 October 2025
Accepted: 24 October 2025
Published: 30 October 2025

https://www.aimspress.com/journal/Math

Research article

A forward progressive physics informed neural network with Taylor series expansion for solving evolution partial differential equations

Wenkai Liu and Yang Liu*

School of Mathematical Sciences, Inner Mongolia University, Hohhot, 010030, China

* **Correspondence:** Email: mathliuyang@imu.edu.cn.

Abstract: The physics informed neural network (PINN) has achieved significant success in solving evolution partial differential equations (PDEs). For improving the prediction accuracy of the PINN, we developed a new PINN with Taylor series expansion (TPINN). However, the low accuracy problem for the PINN or TPINN may occur in approximating the solution of strongly nonlinear evolution PDEs or even linear wave equations. For solving this issue, we introduced a novel efficient method, called a forward progressive PINN with Taylor series expansion (FP-TPINN), where the formula obtained by the Taylor series expansion was applied to construct extra supervised learning task and the domain decomposition in time was used to further improve the accuracy of our proposed method. We carried out several numerical experiments to demonstrate that the TPINN significantly improved the accuracy of the PINN. Moreover, we used the Korteweg-de Vries (KdV) equation to indicate that the TPINN can achieve higher accuracy than the SPINN, and illustrated that the FP-TPINN performed better than the pre-training PINN (PT-PINN) and the dimension-augmented PINN (DaPINN) by solving the Allen-Cahn equation.

Keywords: domain decomposition; PINN; Taylor series expansion; FP-TPINN
Mathematics Subject Classification: 41A58, 65D15, 68T07

1. Introduction

In 2019, Raissi et al. [1] proposed an innovative method of deep learning named physics informed neural network (PINN), which has demonstrated outstanding performance and broad prospects in solving partial differential equations (PDEs). PINN uses automatic differentiation technology to embed physical information into a deep neural network, which learns the mapping relationship between the input of the network and the solution in the whole computational domain by minimizing the loss function composed of residuals generated by physical information, thus acting as an approximator for the solution. Compared to deep learning driven entirely by data, PINN can use a small amount

of data samples to obtain a model with more generalization ability. The research on PINN has gained significant attention in recent years and has shown promising progress in various areas, including heat transfer problems [2, 3], fluid mechanics [4], strongly nonlinear PDEs [5–7], nonlinear dispersive PDEs [8], forward and inverse problems of nonlinear diffusion equations [9], high-dimensional convection-diffusion-reaction equations [10], multi-body dynamic equations [11], option pricing problem [12], nonlinear time distributed-order models [13], variable-order space-fractional advection-diffusion equations [14], fractional water wave models [15], nonlinear integro-differential equations [16], nonlinear elliptic differential equations [17], blood flow model [18], singularly perturbed convection-dominated problem [19], and so on.

A lot of work was conducted to further improve the efficiency and accuracy of PINN. Among various influencing factors, the selection method of training points has been considered to affect the optimization of the network. Wu et al. [20] studied the impact of different sampling methods on the performance of PINN. There are also some studies aimed at integrating new physical information into neural networks. Zhang et al. [21, 22] studied integrating the finite group and the symmetric group into deep neural networks to enhance the model's predictive ability for the exact solution. Unlike the strong form based PINN framework, Samaniego et al. [23] developed a deep neural network method based on the principle of energy (DEM), which uses energy as the loss function. Wang et al. [24] proposed a new deep learning framework called Kolmogorov-Arnold-informed neural networks (KINN), which uses the Kolmogorov-Arnold network (KAN) instead of traditional multilayer perceptron and integrates the strong form, energy form, and inverse form of PDEs to solve forward and inverse problems. Nguyen-Thanh et al. [25] developed a deep energy method (P-DEM) based on parameter space and PINN, which provides a new approach for solving elasticity problems with strain gradient effects. To address the issue of lack of temporal causality in the PINN, Noh et al. [26] proposed a causal PINN-based surrogate model that utilizes temporal causality to overcome the limitations of the standard PINN and achieves more accurate and efficient prediction, and Jung et al. [27] developed a causal PINN framework that significantly improved the accuracy of the predicted solution. To accelerate the convergence of PINN, Jagtap et al. [28] developed adaptive activation function. The key idea of PINN is to optimize the network by minimizing the loss function constructed by initial condition, boundary condition, and governing equation. However, multi-objective functions may cause PINN to fail to converge. Xiang et al. [29] constructed a self-adaptive loss method by combining the adaptive idea with loss functions to balance the relationship between multiple loss functions. By investigating several wave motion problems, Nosrati and Niri [30] demonstrated that the multi-term objective function can hinder the convergence of PINN during the training process, and proposed logarithmic loss and the sigmoidal self-adaptive regularization multiplier to improve this phenomenon. In addition, more scholars [31–33] have combined the physics-guided framework of the PINN with the emerging KAN, allowing it to improve the efficiency and accuracy of solving PDEs by introducing physical constraints while maintaining the advantages of the KAN with fewer parameters and interpretability. The efforts [34–38] have also been made to incorporate the domain decomposition technology into PINN. The combination of the domain decomposition and PINN allows for the decomposition of a complex problem into smaller, more manageable subproblems. In [39–41], the authors have proposed embedding Taylor series expansions into neural networks to increase the physical constraint of the loss function. The improved PINN framework significantly improved the prediction accuracy of the model.

Despite the significant advancements made by the PINN method in solving PDEs, there are still

several issues that warrant further investigation. For evolution PDEs, the prediction accuracy near the initial conditions directly affects the subsequent effectiveness of PINN. The absence of time-dependent features may lead to insufficient network capture of time related complexity, resulting in training failure [42]. For tackling strongly nonlinear PDEs, the local minimum problem can potentially hinder the network's ability to accurately learn the underlying physical information from the data [6, 43]. In order to accurately capture the solutions of PDEs, a large number of residual datasets are usually required, which leads to an increase in the demand for computing resources [44]. Therefore, how to balance the number and distribution of residual points and computational efficiency is an important issue in PINN research.

In this article, we develop a method named TPINN by combining PINN and Taylor series expansion to better approximate the solution of the evolution PDEs. The proposed TPINN introduces a truncated Taylor series as an additional physical constraint, which is used to construct a new loss term that is considered as a new supervised learning task to further optimize the parameters of neural networks. The main contribution of this article is to integrate the TPINN into a forward progressive training framework, which uses domain decomposition strategy to divide the computational domain into multiple subdomains and the resampling technique to enable the network to fully learn the physical information within the entire computational domain.

The characteristics of this study are as follows:

• The TPINN framework is proposed. The differential operator in the evolution partial differential equation (PDE) is used to obtain the Taylor expansion, which not only follows the physical laws described by the original equation, but also provides a numerical constraint condition that makes it easier for the neural network to capture the physical laws described by the original equation during the training process.

• Innovative integration of the TPINN framework and the forward progressive training strategy. The Taylor expansion term provides a precise local physical constraint, forming a collaborative mechanism with the forward progressive training strategy that decomposes complex problems. The Taylor expansion ensures the correctness of the physical laws learned within each subdomain, while the progressive strategy guarantees stable propagation to the next subdomain during the solving process.

• Error estimates are provided. We analyze the truncation error of the Taylor expansion terms and the interaction between the truncation error and the approximation error of the neural network.

• Comprehensive numerical verification is conducted. We not only validated the performance of the TPINN compared to the standard PINN, but also specifically demonstrate the advantage of the forward progressive TPINN (FP-TPINN) over its variants.

The remainder of this paper is organized as follows. The basic idea of PINN for solving PDEs is first presented in Section 2. In Section 3, we introduce the construction principle of TPINN and provide a detailed description of the forward progressive training strategy. We derive the error estimates of the proposed PINN framework in Section 4. In Section 5, several numerical experiments are used to test the effectiveness of our proposed methods. We provide discussions on the temporal causality analysis and length of time intervals in Section 6. Finally, we present some conclusions about our work in Section 7.

2. Solving PDEs with standard PINN

In this section, to briefly review the original PINN method for solving PDE, we focus on the time-dependent equation. Consider the following one-dimensional evolution PDE:

$$\begin{aligned} &\frac{\partial u}{\partial t}(x,t) - \mathcal{D}[u(x,t)] = 0, \quad && x \in \Omega, t \in (0,T], \\ &u(x,0) = u_0(x), && x \in \overline{\Omega}, \\ &u(x,t) = \mathcal{B}(x,t), && x \in \partial\overline{\Omega}, t \in [0,T], \end{aligned} \tag{2.1}$$

where $\mathcal{D}[\cdot]$ is a general partial differential operator. In the standard PINN framework, the basic idea is to construct a fully connected feed-forward neural network as a to-be-trained approximator of the solution of (2.1) and update the parameters of the network by optimizing the loss function.

The general form of the neural network can be expressed as

$$\begin{aligned} u(x,t;\Theta) &= (P_m \circ \sigma \circ P_{m-1} \circ \sigma \cdots \circ \sigma \circ P_1)(x,t), \\ P_k(z) &= W_k z + b_k, 1 \le k \le m, \end{aligned} \tag{2.2}$$

where the spatial and temporal variables (x,t) are inputs of the network, σ denotes the activation function, and $\Theta = \{W_k, b_k\}_{i=1}^m$ are the trainable parameters of the network to be updated during the process of minimizing the loss function. P_k $(1 \le k \le m-1)$ and P_m represent the k-th hidden layer and the output layer, respectively. The total loss function consists of three loss terms as follows:

$$\mathcal{L}(\Theta;\mathcal{J}) = w_i\mathcal{L}_i(\Theta;\mathcal{J}_i) + w_b\mathcal{L}_b(\Theta;\mathcal{J}_b) + w_f\mathcal{L}_f(\Theta;\mathcal{J}_f), \ \ \mathcal{J} = \{\mathcal{J}_i, \mathcal{J}_b, \mathcal{J}_f\}, \tag{2.3}$$

where

$$\begin{aligned} &\mathcal{L}_i(\Theta;\mathcal{J}_i) = \frac{1}{|\mathcal{J}_i|}\sum_{x\in\mathcal{J}_i} |u(x,0;\Theta) - u_0(x)|^2, && \mathcal{J}_i = \{x_i \in \overline{\Omega}\}_{i=1}^{N_i}, \\ &\mathcal{L}_b(\Theta;\mathcal{J}_b) = \frac{1}{|\mathcal{J}_b|}\sum_{(x,t)\in\mathcal{J}_b} |u(x,t;\Theta) - \mathcal{B}(x,t)|^2, && \mathcal{J}_b = \{(x_i,t_i) \in \partial\overline{\Omega}\times[0,T]\}_{i=1}^{N_b}, \\ &\mathcal{L}_f(\Theta;\mathcal{J}_f) = \frac{1}{|\mathcal{J}_f|}\sum_{(x,t)\in\mathcal{J}_f} \left|\frac{\partial u}{\partial t}(x,t;\Theta) - \mathcal{D}[u(x,t;\Theta)]\right|^2, && \mathcal{J}_f = \{(x_i,t_i) \in \Omega\times(0,T]\}_{i=1}^{N_f}. \end{aligned} \tag{2.4}$$

Subsequently, $\mathcal{L}_i$ corresponds to the residuals on the initial dataset, $\mathcal{L}_b$ denotes the loss of the boundary conditions, and $\mathcal{L}_f$ represents the mean squared error of the PDE described in (2.1). The parameters Θ of the network are updated by minimizing these loss terms, and the setting of loss terms in PINN is to ensure that the model can simultaneously consider physical laws, initial conditions, and boundary conditions during the learning process. The weights w_i, w_b, and w_f control the proportion of different loss components in the loss function $\mathcal{L}(\Theta;\mathcal{J})$, and N_i, N_b, and N_f denote the number of training points in the datasets $\mathcal{J}_i$, $\mathcal{J}_b$, and $\mathcal{J}_f$, respectively.

Remark 2.1. *The symbols with i, b, and f as subscripts represent the parameters related to the initial conditions, boundary conditions, and governing equation of (2.1), respectively.*

3. Methodology

In this section, we enhance the accuracy of the standard PINN via the truncated Taylor series expression obtained by combining the governing equation of (2.1), and we use the time domain decomposition method and resampling techniques to further augment the effectiveness and reliability of TPINN for solving the strongly nonlinear evolution PDEs.

3.1. TPINN

To simplify the description of the algorithm, the governing equation of (2.1) is rewritten as

$$\frac{\partial u}{\partial t}(x,t) = \mathcal{D}[u(x,t)]. \tag{3.1}$$

Then, the second order partial derivative of $u(x,t)$ with respect to time t can be defined as

$$\frac{\partial^2 u}{\partial t^2}(x,t) = \frac{\partial \mathcal{D}[u(x,t)]}{\partial t}. \tag{3.2}$$

On this basis, we perform Taylor series expansion with respect to $u(x,t)$ in the time direction:

$$u(x,t) = u(x,t-\Delta t) + \Delta t \frac{\partial u}{\partial t}(x,t-\Delta t) + \frac{\Delta t^2}{2!}\frac{\partial^2 u}{\partial t^2}(x,t-\Delta t) + \cdots, \tag{3.3}$$

where $\Delta t = \frac{t}{n}$. The parameter n controls the size of the time step Δt. By substituting (3.1) and (3.2) into (3.3), a level of truncation with leading order of $O(\Delta t^k)$ can be given by

$$\begin{aligned} u(x,t) =& u(x,t-\Delta t) + \Delta t \mathcal{D}[u(x,t-\Delta t)] + \frac{\Delta t^2}{2!}\frac{\partial \mathcal{D}[u(x,t-\Delta t)]}{\partial t} \\ &+ \cdots + \frac{\Delta t^k}{k!}\frac{\partial^{k-1}\mathcal{D}[u(x,t-\Delta t)]}{\partial t^{k-1}} + O(\Delta t^k). \end{aligned} \tag{3.4}$$

The Taylor expansion term (3.4) provides a local approximation for the exact solution u. Given (3.4), we introduce $\mathcal{F}(x,t)$ to represent the residual of the Taylor expansion term, which is defined as

$$\mathcal{F} := u(x,t) - \left(u(x,t-\Delta t) + \Delta t \mathcal{D}[u(x,t-\Delta t)] + \frac{\Delta t^2}{2!}\frac{\partial \mathcal{D}[u(x,t-\Delta t)]}{\partial t} + \cdots + \frac{\Delta t^k}{k!}\frac{\partial^{k-1}\mathcal{D}[u(x,t-\Delta t)]}{\partial t^{k-1}} \right). \tag{3.5}$$

The residual $\mathcal{F}(x,t)$ provides a structured physical constraint for the network, enabling it to focus more on specific form of the solution during the learning process. Therefore, the total loss function of the TPINN framework is given by adding the mean squared error loss of (3.5) to (2.3):

$$\mathcal{L}(\Theta;\mathcal{J}) = w_i\mathcal{L}_i(\Theta;\mathcal{J}_i) + w_b\mathcal{L}_b(\Theta;\mathcal{J}_b) + w_f\mathcal{L}_f(\Theta;\mathcal{J}_f) + w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\Theta;\mathcal{J}_f), \tag{3.6}$$

where $w_{\mathcal{F}}$ is the weight of the newly added loss component and

$$\mathcal{L}_{\mathcal{F}}(\Theta;\mathcal{J}_f) = \frac{1}{|\mathcal{J}_f|}\sum_{(x,t)\in\mathcal{J}_f} |\mathcal{F}(x,t)|^2. \tag{3.7}$$

For solving some PDEs that fail to obtain the prediction result with high precision by the standard PINN method, the weight w_f is set to be zero, with the aim of considering the formula (3.5) as the primary supervised learning task to optimize the parameters Θ of the network. For solving other problems that have been made the successful application of the PINN, we define the weight $w_f = 1$. This means that the formula (3.5) is regarded as the extra supervised leaning task, which is conducive to improve the prediction accuracy of the PINN.

3.2. Forward progressive training strategy

Let us decompose the time domain $[0, T]$ into q_{max} independent subdomains as shown below:

$$[T_0 = 0, T_1], [T_1, T_2], \cdots, [T_{q-1}, T_q], \cdots, [T_{q_{max}-1}, T_{q_{max}} = T]. \tag{3.8}$$

The key idea of the forward progressive training strategy is to construct an independent neural network $u(x, t; \Theta_q)$ within the time interval $[T_{q-1}, T_q]$ and then combine it with the Taylor expansion term to train the parameters Θ_q. The predicted value at time T_q serves as the initial condition for the next time interval $[T_q, T_{q+1}]$.

For the q-th time interval $[T_{q-1}, T_q]$, the corresponding optimization problem is defined as

$$\mathcal{L}(\Theta_q; \mathcal{J}_q) = w_i \mathcal{L}_i(\Theta_q; \mathcal{J}_i^q) + w_b \mathcal{L}_b(\Theta_q; \mathcal{J}_b^q) + w_f \mathcal{L}_f(\Theta_q; \mathcal{J}_f^q) + w_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}(\Theta^q; \mathcal{J}_f^q), \tag{3.9}$$

where the datasets $\mathcal{J}_q$, $\mathcal{J}_i^q$, $\mathcal{J}_b^q$, and $\mathcal{J}_f^q$ for training the corresponding loss components are denoted as

$$\begin{aligned} \mathcal{J}_q &= \{\mathcal{J}_i^q, \mathcal{J}_b^q, \mathcal{J}_f^q\}, \\ \mathcal{J}_i^q &= \{x_i \in \overline{\Omega}\}_{i=1}^{N_i}, \\ \mathcal{J}_b^q &= \{(x_i, t_i) \in \partial\overline{\Omega} \times [T_{q-1}, T_q]\}_{i=1}^{N_b}, \\ \mathcal{J}_f^q &= \{(x_i, t_i) \in \Omega \times [T_{q-1}, T_q]\}_{i=1}^{N_f}, \end{aligned} \tag{3.10}$$

and

$$\mathcal{L}_i(\Theta_q; \mathcal{J}_i^q) = \frac{1}{|\mathcal{J}_i^q|} \sum_{x \in \mathcal{J}_i^q} |u(x, T_{q-1}; \Theta_q) - u(x, T_{q-1}; \Theta_{q-1})|^2. \tag{3.11}$$

Specifically, we define the dataset $\mathcal{J}_f^1 = \{(x_i, t_i) \in \Omega \times (T_0, T_1]\}_{i=1}^{N_f}$ and the initial value at T_0 as $u(x, T_0; \Theta_0) = u_0(x)$ in the first time interval $[T_0, T_1]$.

Despite the fact that the standard PINN can use different sampling methods, such as random sampling and Latin hypercube sampling, to select training points, all training points are selected before training and fixed throughout the entire training process. To further improve the prediction accuracy of the TPINN method, we incorporate the resampling technique [43] during the training process of each time subdomain in the forward progressive training strategy. The resampling strategy can randomly reselect new training points and combine them with existing sample points to form new training data during the training process, which allows the parameters of the network to be adjusted in a timely manner based on the dynamic changes of the problem or new information feedback, making subsequent analysis or simulation more accurate and reliable. More details about the resampling algorithm, such as the frequency of resampling and the number of resampling points, are presented in Algorithm 1. The procedure of the proposed FP-TPINN is detailed in Algorithm 2.

Algorithm 1. The resampling strategy

Input: Training data sets $\mathcal{J}_i, \mathcal{J}_b, \mathcal{J}_f$:
Iteration step number $\mathcal{I}$;
Output: the prediction $u(x, t; \Theta)$
Step 1: Apply Adam algorithm to optimize the loss function $\mathcal{L}(\Theta; \mathcal{J})$
for $i = 1, \cdots, \mathcal{I}$ **do**
1. Compute the loss function $\mathcal{L}(\Theta; \mathcal{J})$.
2. Update Θ by Adam optimizer
3. **if** $i\%\ell = 0$ and i $< \zeta$ ($\zeta < \frac{\mathcal{I}}{2}$) (or $\delta \times \frac{\mathcal{I}}{10} < i < (\delta + 1) \times \frac{\mathcal{I}}{10}$, $\delta = 1, 3, 5, 7, 9$) **then**
Resample $N_{rf} = 0.8N_f$ points in the spatio-temporal domain and N_{rb} points on the boundary
Update N_f points in $\mathcal{J}_f$ and N_b points in $\mathcal{J}_b$.
end
end
Step 2: Continue to use the L-BFGS method to update the parameters Θ.

Algorithm 2. TPINN with the forward progressive training strategy (FP-TPINN)

Step 1: Divide the time interval $[0, T]$ into q_{max} subdomains:
$[T_0 = 0, T_1], [T_1, T_2], \cdots, [T_{q-1}, T_q], \cdots, [T_{q_{max}-1}, T_{q_{max}} = T]$.
Step 2: **for** $q = 1, \cdots, q_{max}$ **do**
1. Select the training data $\mathcal{J}_q$.
2. Generate the initial value $u(x, T_{q-1}; \Theta_{q-1})$ at time T_{q-1}.
3. Initialize the parameters Θ_q of the network.
4. Compute the loss function $\mathcal{L}(\Theta_q; \mathcal{J}_q)$.
5. Set the number of resampling points in the spatio-temporal domain $\Omega \times [T_{q-1}, T_q]$ to N_{rf} and the number of resampling points on the boundary to N_{rb}.
6. Update Θ_q by the Algorithm 1.
7. Obtain the prediction $u(x, t; \Theta_q)$.
end

4. Error estimates

In this section, we advance to the error estimates of the proposed Taylor-series-enhanced loss term. We assume that the following conditions hold.

Assumption 4.1. *Suppose* $u \in C^{k+1}([0, T]; H^m(\Omega))$.

Assumption 4.2. *Let the differential operator* $\mathcal{D}$ *be Lipschitz continuous. It holds that*

$$\|\mathcal{D}[u] - \mathcal{D}[v]\|_{L^2} \leq L_{\mathcal{D}} \|u - v\|_{H^m}, \tag{4.1}$$

where $L_{\mathcal{D}}$ *is a constant.*

The core of our proposed TPINN method is to utilize Taylor series expansion as a new constraint term to enhance the optimization capability of network parameters. Equation (3.4) is rewritten as

$$u(x,t) = u(x,t-\Delta t) + \sum_{j=1}^{k} \frac{\Delta t^j}{j!} \frac{\partial^{j-1}\mathcal{D}[u(x,t-\Delta t)]}{\partial t^{j-1}} + R_{k+1}(x,t), \tag{4.2}$$

where $R_{k+1}(x,t)$ denotes the remainder term. Then, we analyze the interaction between the truncation error of the Taylor expansion and the approximation error of the network.

Lemma 4.1. *For the remainder term $R_{k+1}(x,t)$, we have*

$$|R_{k+1}(x,t)| \leq C_1 \frac{\Delta t^{k+1}}{(k+1)!} \sup_{\tau\in[t-\Delta t,t]} \left| \frac{\partial^k \mathcal{D}[u(x,\tau)]}{\partial t^k} \right|, \tag{4.3}$$

where C_1 is a positive constant.

Proof. According to Eq (4.2), the remainder term $R_{k+1}(x,t)$ is

$$R_{k+1}(x,t) = \frac{\partial^k \mathcal{D}[u(x,\xi)]}{\partial t^k} \frac{(\Delta t)^{k+1}}{(k+1)!}. \tag{4.4}$$

Then, we can obtain

$$|R_{k+1}(x,t)| \leq \frac{\Delta t^{k+1}}{(k+1)!} \left| \frac{\partial^k \mathcal{D}[u(x,\xi)]}{\partial t^k} \right| \leq \frac{\Delta t^{k+1}}{(k+1)!} \sup_{\tau\in[t-\Delta t,t]} \left| \frac{\partial^k \mathcal{D}[u(x,\tau)]}{\partial t^k} \right|. \tag{4.5}$$

Let $C_1 = 1$ to draw a conclusion. □

For convenience, the network $u(x,t;\Theta)$ is rewritten as $u_\Theta(x,t)$. We define the approximation error between the network u_Θ and the exact solution u of a neural network as

$$\epsilon_a(x,t) = u_\Theta(x,t) - u(x,t). \tag{4.6}$$

Substituting the network u_Θ into $\mathcal{F}(x,t)$, we obtain

$$\mathcal{F}_\Theta(x,t) = u_\Theta(x,t) - \mathcal{T}_k[u_\Theta(x,t)], \tag{4.7}$$

where

$$\mathcal{T}_k[u_\Theta] = u_\Theta(x,t-\Delta t) + \sum_{j=1}^{k} \frac{\Delta t^j}{j!} \frac{\partial^{j-1}\mathcal{D}[u_\Theta(x,t-\Delta t)]}{\partial t^{j-1}}. \tag{4.8}$$

Theorem 4.1. *It holds that*

$$|\mathcal{F}_\theta(x,t)| \leq C_2 \|\epsilon_a\|_{W^{k,\infty}} + C_3 \frac{\Delta t^{k+1}}{(k+1)!}, \tag{4.9}$$

where C_2 and C_3 are positive constants.

Proof. We decompose $\mathcal{F}_\theta$ into the following three parts:

$$\mathcal{F}_\theta = [u_\theta - u] - [\mathcal{T}_k[u_\theta] - \mathcal{T}_k[u]] + [u - \mathcal{T}_k[u]]. \tag{4.10}$$

Based on Assumption 4.2 and the approximation error ϵ_a, we have

$$|\mathcal{T}_k[u_\theta] - \mathcal{T}_k[u]| \leq L_k \|\epsilon_a\|_{W^{k,\infty}}, \tag{4.11}$$

where L_k is a constant. Then, by setting $C_3 = C_1 \cdot \sup\left|\frac{\partial^k \mathcal{D}[u]}{\partial t^k}\right|$ and considering the remainder term $R_{k+1}(x,t)$, we get

$$R_{k+1}(x,t) = u - \mathcal{T}_k[u] \leq C_3 \frac{\Delta t^{k+1}}{(k+1)!}. \tag{4.12}$$

There exists a constant $C_2 = 1 + L_k$ such that

$$|\mathcal{F}_\theta| \leq C_2 \|\epsilon_a\|_{W^{k,\infty}} + C_3 \frac{\Delta t^{k+1}}{(k+1)!}. \tag{4.13}$$

The conclusion is obtained. □

5. Numerical results

In this section, we provide some numerical experiments to illustrate the performance of the proposed TPINN and FP-TPINN methods. The relative L^2 norm error is applied to evaluate the performance of our proposed methods and the standard PINN:

$$\|u(x,t) - u(,x,t;\Theta)\|_{L^2} = \frac{\sqrt{\sum_{i=1}^{N} |u(x_i,t_i) - u(,x_i,t_i;\Theta)|^2}}{\sqrt{\sum_{i=1}^{N} |u(x_i,t_i)|^2}}. \tag{5.1}$$

The tanh function is selected as the activation function, and Table 1 lists the other training configurations used in each numerical experiment and illustrates the methods involved in each example. For convenience, we set an initial value for each variable at the beginning of each experiment, which runs through the entire experimental process. Unless otherwise stated, all prediction results are given under these initial settings and the parameter k is set to 2. When comparing the accuracy of prediction results, individual variables change while other variables remain unchanged. In this work, the hyperparameters used in the model compared to our proposed method, such as network structure, optimizer, activation function, and the number of configuration points and initial and boundary points, are consistent with those used in our model.

Table 1. Training configurations used for all numerical examples.

Example	Method	Network structure	Optimizer	Iterative step	Learning rate
5.1 KdV	Taylor expansion	[2, 30, 30, 30, 30, 1]	L-BFGS	-	-
5.2 Potential Burgers	Taylor expansion	[2, 10, 10, 10, 10, 2]	L-BFGS	-	-
5.3 Reaction	Taylor expansion, time-domain decomposition	[2, 10, 10, 10, 10, 1]	L-BFGS	-	-
5.4 AC	Taylor expansion, time-domain decomposition, resampling	[2, 50, 50, 50, 50, 50, 1]	Adam + L-BFGS	10000	0.001
5.5 Wave	Taylor expansion, time-domain decomposition, resampling	[2, 100, 100, 100, 100, 1]	Adam + L-BFGS	10000	0.001
5.6 Improved Boussinesq	Taylor expansion	[2, 20, 20, 20, 20, 1]	L-BFGS	-	-
5.7 Flow mixing	Taylor expansion	[3, 10, 10, 10, 10, 1]	L-BFGS	-	-
5.8 Diffusion-reaction	Taylor expansion	[6, 20, 20, 20, 20, 20, 20, 20, 20, 1]	L-BFGS	-	-

5.1. Korteweg-de Vries (KdV) equation

We discuss a KdV equation given by

$$u_t + uu_x + u_{xxx} = 0,\ (x,t) \in (0,1) \times (0,1], \tag{5.2}$$

with initial condition

$$u(x,0) = 12\mathrm{sech}^2(x), x \in [0,1], \tag{5.3}$$

and boundary conditions

$$\begin{aligned} u(0,t) &= 12\mathrm{sech}^2(-4t), \quad t \in [0,1], \\ u(1,t) &= 12\mathrm{sech}^2(1-4t), \quad t \in [0,1], \end{aligned} \tag{5.4}$$

where the analytical solution is $u(x,t) = 12\mathrm{sech}^2(x-4t)$. Then, the total loss function to update the parameters of the network is defined as

$$\mathcal{L}(\Theta;\mathcal{J}) = w_u\mathcal{L}_u(\Theta;\mathcal{J}_u) + w_f\mathcal{L}_f(\Theta;\mathcal{J}_f) + w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}}(\Theta;\mathcal{J}_f),\ \ \mathcal{J} = \{\mathcal{J}_u, \mathcal{J}_f\}, \tag{5.5}$$

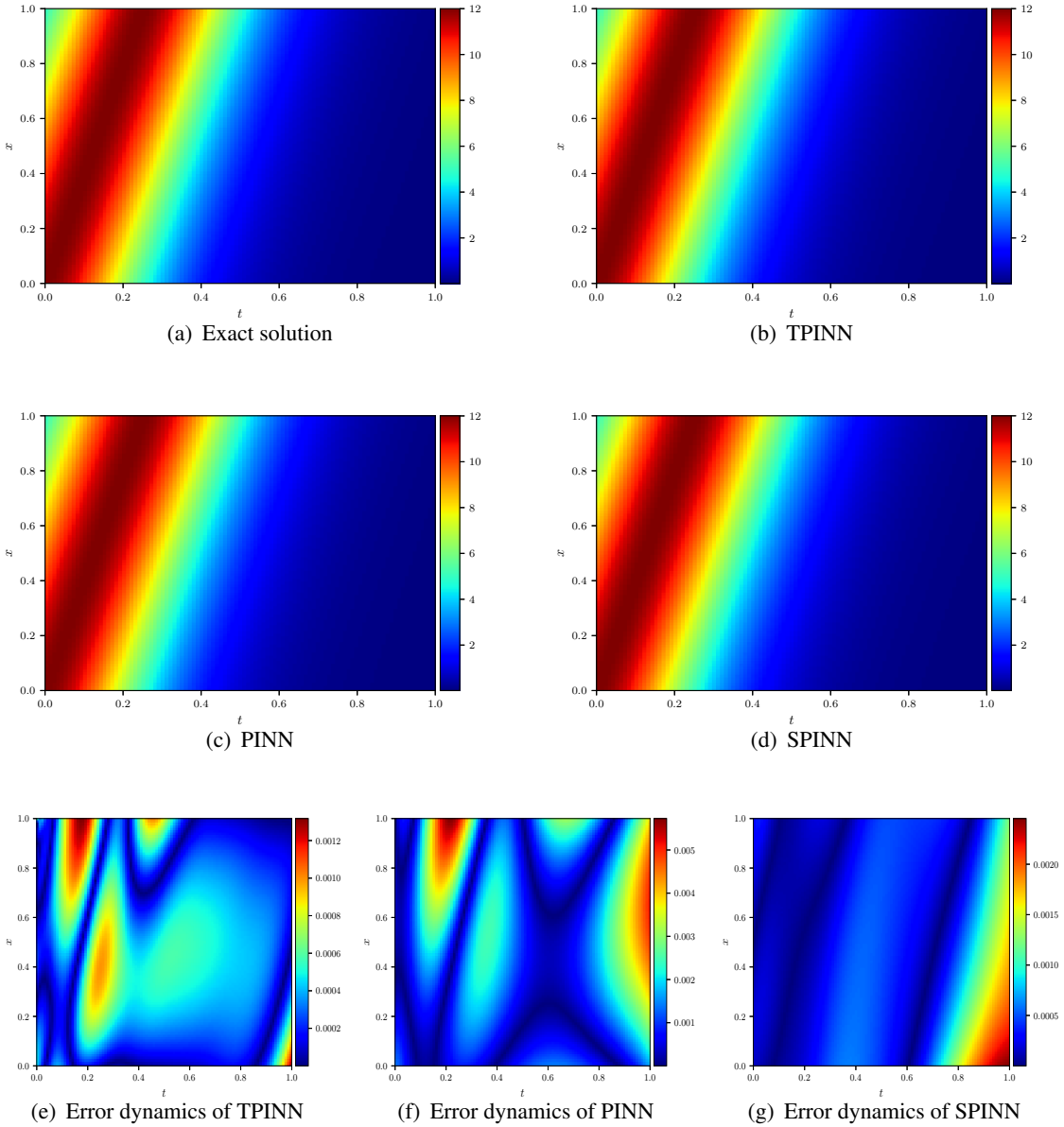
where

$$\mathcal{L}_u(\Theta;\mathcal{J}_u) = \frac{1}{|\mathcal{J}_u|}\sum_{(x,t)\in\mathcal{J}_u} |u(x,t;\Theta) - u(x,t)|^2, \mathcal{J}_u = \{(x_i,t_i,u_i)\}_{i=1}^{N_u}. \tag{5.6}$$

$\mathcal{L}_u(\Theta;\mathcal{J}_u)$ denotes the mean squared error on the initial and boundary dataset $\mathcal{J}_u$ and w_u represents the weight.

In this example, we randomly sample $N_f = 3000$ collocation points from the spatiotemporal domain and $N_u = 100$ points from initial and boundary data to compose the training data. The weights of the loss components are set to $w_u = 1$, $w_f = 0$, and $w_{\mathcal{F}} = 100$ and the parameter n is set as 100.

To provide an intuitive evaluation, we plot the prediction in the whole spatiotemporal domain and the absolute errors of the PINN, the SPINN [45], and the TPINN in Figure 1, and provide a comparison between the exact solution and the corresponding predictions in Figure 2 to display more detailed observation, which validates that the proposed TPINN is superior to the standard PINN and the SPINN.



(a) Exact solution (b) TPINN

(c) PINN (d) SPINN

(e) Error dynamics of TPINN (f) Error dynamics of PINN (g) Error dynamics of SPINN

Figure 1. KdV equation: (top) comparison between the exact solution and the solutions solved by the PINN, the SPINN, and the TPINN; (bottom) absolute errors of the PINN, the SPINN, and the TPINN.

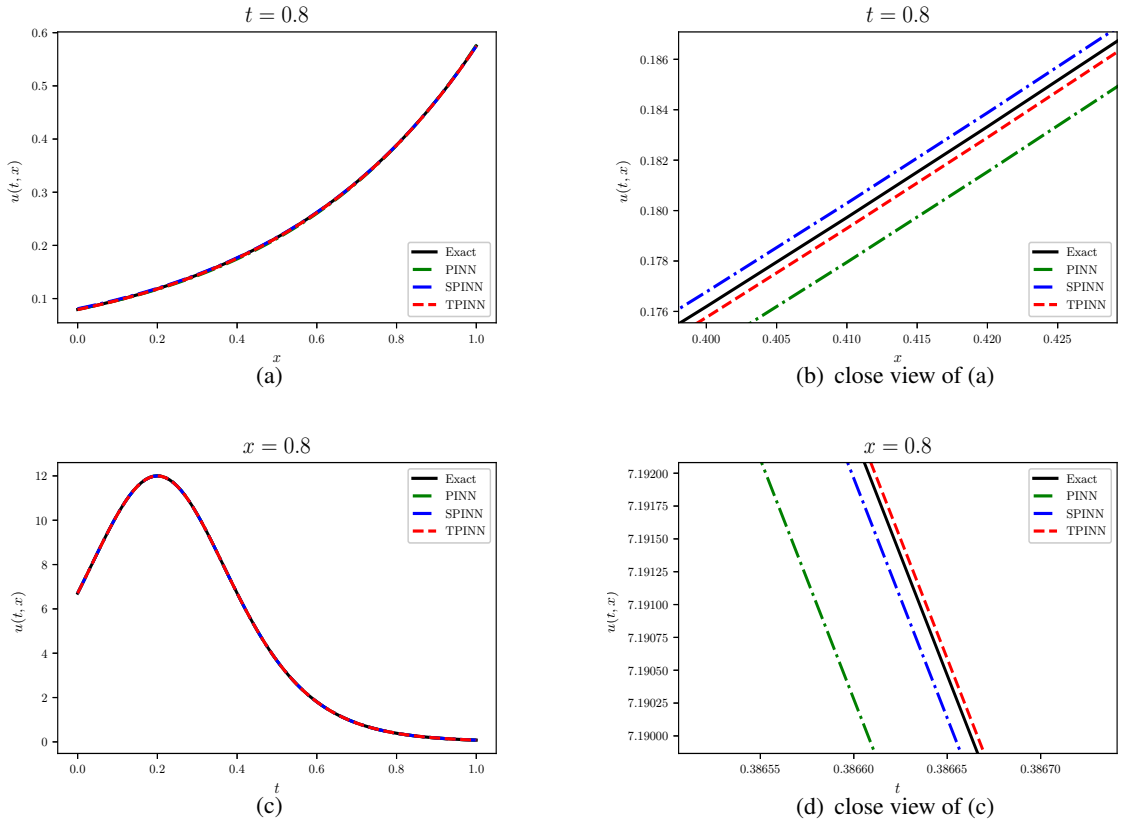


Figure 2. KdV equation: comparison of the exact solution and the solution solved by the PINN, the SPINN, and the TPINN.

In Table 2, we present the training results given by the PINN, the SPINN, and the TPINN. One can observe that the TPINN enhances the prediction accuracy of the standard PINN and the SPINN. When $N_f = 2000$ and 3000, the computational cost of the TPINN is significantly lower than that of the PINN. In order to investigate the impact of adding the terms of expansion on the TPINN method, we present the prediction accuracy of the TPINN at different k in Table 3. The results indicate that even if n is small, increasing the expansion terms can effectively improve the performance of the TPINN. Moreover, we also find that increasing n can achieve higher prediction accuracy when the number k of expansion terms remains constant. We present the evolution diagram of the L^2 norm of the loss gradients for the PINN and the TPINN during the training process in Figure 3. From the graph, it can be seen that the gradient of the TPINN rapidly decreases to a lower level, indicating that the model can update parameters more efficiently, with better convergence and stability. Finally, we present the numerical results of the TPINN, the PINN, and the SPINN with different network architectures in Table 4 and the performance of the TPINN with different weights in Table 5, which demonstrate that the proposed TPINN has a good performance.

Table 2. KdV equation: the computational cost and the relative L^2 errors of the TPINN, the PINN, and the SPINN with different parameters n for different numbers of training points N_f.

	N_f	PINN	TPINN			SPINN
			$n = 60$	$n = 80$	$n = 100$	
Relative L^2 error	1000	4.8385e-04	1.7407e-04	1.3037e-04	9.8527e-05	1.2761e-04
	2000	9.7769e-04	8.6517e-05	5.7600e-05	1.3270e-04	6.5901e-05
	3000	3.0576e-04	8.2006e-05	9.5781e-05	7.1979e-05	1.0684e-04
Time(s)	1000	7.84	9.07	8.86	10.17	6.79
	2000	18.06	11.94	13.42	10.40	10.12
	3000	25.85	16.65	18.53	23.28	13.81

Table 3. KdV equation: the relative L^2 errors of the TPINN with $N_f = 3000$ for different parameters k and n.

TPINN	n			
	$n = 5$	$n = 10$	$n = 20$	$n = 50$
$k = 1$	1.1597e-02	5.5453e-03	4.3747e-03	1.4520e-03
$k = 2$	7.2001e-03	2.9954e-03	2.3783e-04	1.7504e-04
$k = 3$	7.0655e-03	1.0455e-03	2.4589e-04	7.9653e-05

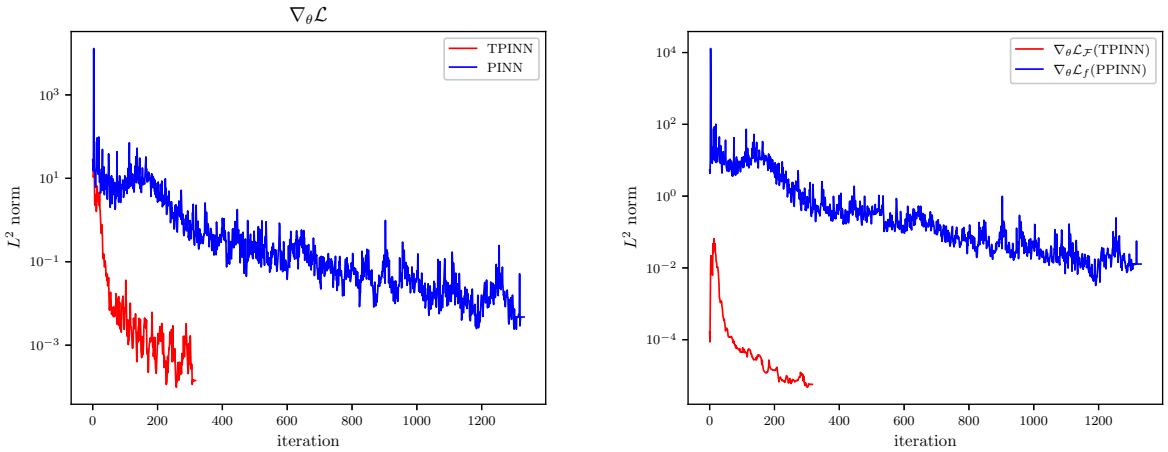


Figure 3. KdV equation: the evolution diagram of the loss gradients from the TPINN versus the standard PINN.

Table 4. KdV equation: the computational cost and the relative L^2 errors of the TPINN, the PINN, and the SPINN with different network architectures.

Network structure			PINN	TPINN ($w_{\mathcal{F}} = 100$)			SPINN
Depth	Width			$n = 60$	$n = 80$	$n = 100$	
2	10	Relative L^2 error	2.8013e-04	2.9920e-05	4.4091e-05	3.3040e-05	1.1597e-04
		Time(s)	4.47	5.42	8.99	8.99	2.47
2	20	Relative L^2 error	6.4498e-04	6.7469e-05	3.6693e-05	5.6842e-05	7.6321e-05
		Time(s)	7.82	5.67	4.83	4.81	4.96
3	10	Relative L^2 error	5.7579e-05	3.0591e-05	5.5185e-05	1.0372e-04	1.8765e-04
		Time(s)	7.45	9.81	6.93	5.78	3.15
3	20	Relative L^2 error	1.9099e-04	7.6522e-05	1.1167e-04	8.9386e-05	4.2747e-05
		Time(s)	9.33	9.31	10.29	8.36	6.48
4	20	Relative L^2 error	1.6469e-04	7.9599e-05	1.5130e-04	7.2142e-05	8.2036e-05
		Time(s)	8.53	15.00	19.60	16.30	12.02
4	40	Relative L^2 error	1.9830e-04	1.0089e-04	1.2367e-04	5.8203e-05	4.7907e-05
		Time(s)	30.14	27.32	23.31	25.53	22.80
6	30	Relative L^2 error	3.8655e-04	9.4959e-05	8.4570e-05	6.4763e-05	1.0390e-04
		Time(s)	35.87	42.12	28.78	35.94	21.96
6	40	Relative L^2 error	1.8299e-04	1.1659e-04	7.3174e-05	8.6014e-05	8.2550e-05
		Time(s)	41.34	52.77	44.32	39.95	20.57

Table 5. KdV equation: the computational cost and the relative L^2 errors of the TPINN with different weights.

	TPINN ($n = 80$)			
	$w_{\mathcal{F}} = 1$	$w_{\mathcal{F}} = 20$	$w_{\mathcal{F}} = 100$	$w_{\mathcal{F}} = 200$
Relative L^2 error	1.7110e-04	1.4781e-04	7.1979e-05	9.5486e-05
Time(s)	19.72	19.93	23.28	15.33

5.2. Potential Burgers equations

In this example, we investigate the potential Burgers equations described as

$$\begin{aligned} &v_x - u = 0, && (x,t) \in (0.1, 1.1) \times (0, 1], \\ &v_t - u_x + \frac{1}{2}u^2 = 0, && (x,t) \in (0.1, 1.1) \times (0, 1], \end{aligned} \tag{5.7}$$

with initial conditions

$$u(x,0) = -\frac{4(x+2)}{x(4+x)}, \quad v(x,0) = -2\ln\left(\frac{x}{3} + \frac{x^2}{12}\right), \quad x \in [0.1, 1.1], \tag{5.8}$$

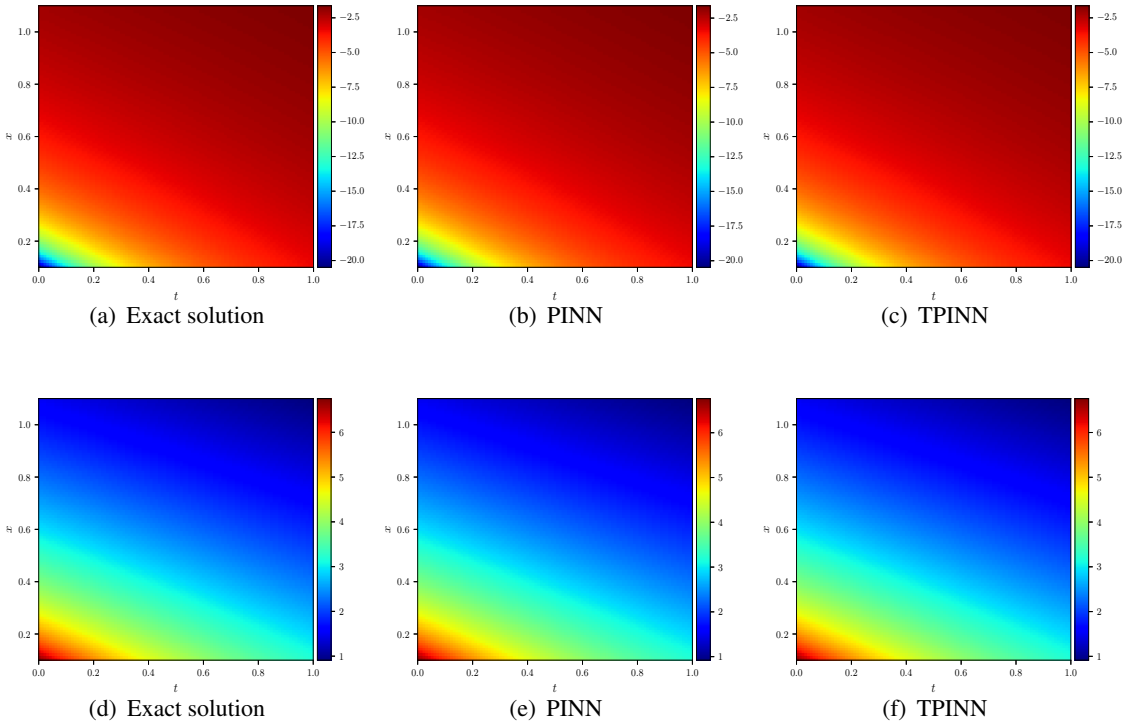
and boundary conditions

$$\begin{aligned} u(0.1,t) &= \frac{-840}{200t+41}, \quad v(0.1,t) = -2\ln\left(\frac{t}{6} + \frac{41}{1200}\right), \quad t \in [0,1], \\ u(1.1,t) &= \frac{-1240}{200t+561}, \quad v(1.1,t) = -2\ln\left(\frac{t}{6} + \frac{187}{400}\right), \quad t \in [0,1], \end{aligned} \tag{5.9}$$

with the analytical solutions $u(x,t) = \frac{-4(x+2)}{2t+4x+x^2}$ and $v(x,t) = -2\ln\left(\frac{t}{6} + \frac{x}{3} + \frac{x^2}{12}\right)$.

The structure of the total loss function is the same as the above example, and the training data is composed of $N_f = 2000$ randomly distributed collocation points in the spatiotemporal domain and $N_u = 100$ points for training the initial and boundary conditions. The weights of the loss components are the same as the numerical examples above and the parameter n is set as 60.

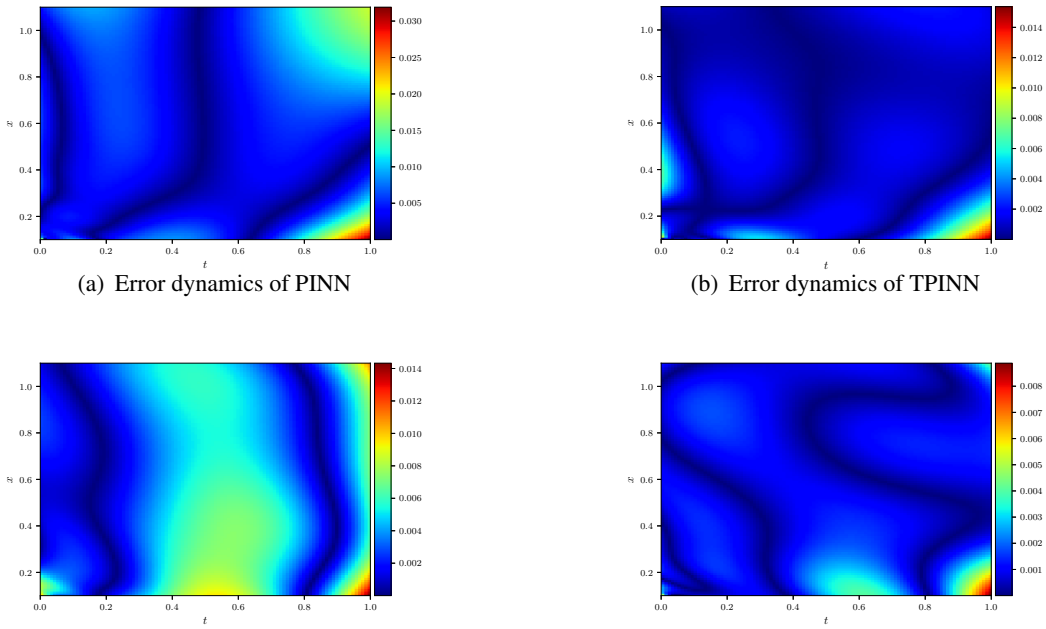
We plot the density diagrams of the exact and predicted solutions in Figure 4 and present the absolute error distributions of the predicted results for $u(x,t)$ and $v(x,t)$ over the entire spatiotemporal domain in Figure 5. Moreover, an intuitive comparison between the analytical solution and the training result is shown in Figure 6. It is obvious to find that the prediction given by the proposed TPINN achieves better resolution than that given by the standard PINN. In addition, Table 6 summarizes the test errors of the exact solutions u and v.



(a) Exact solution (b) PINN (c) TPINN

(d) Exact solution (e) PINN (f) TPINN

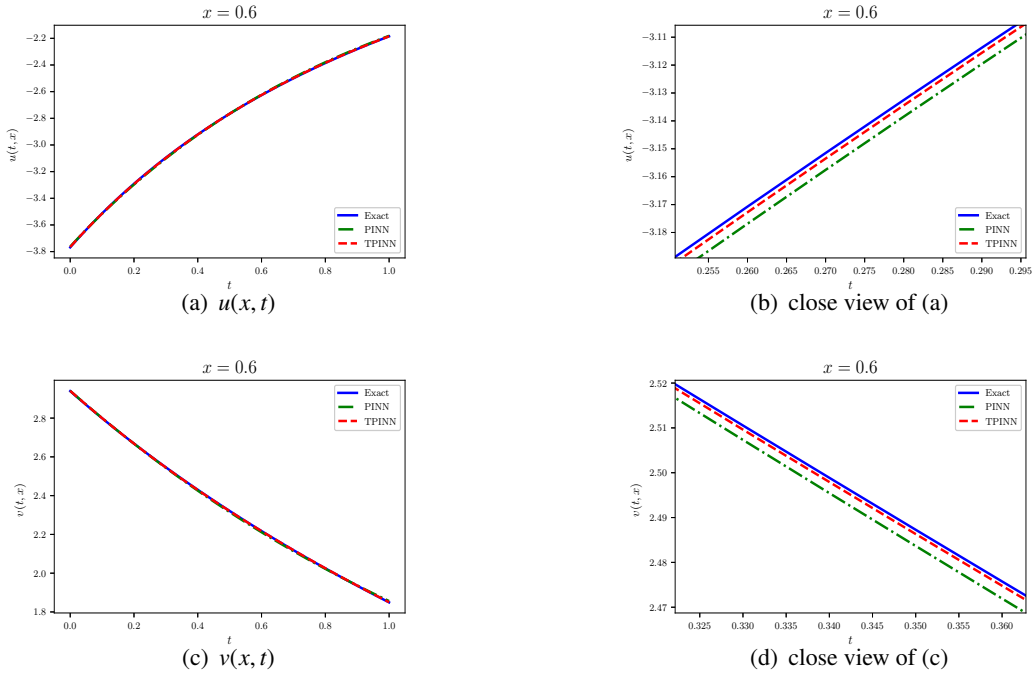
Figure 4. Potential Burgers equations: the density diagrams of the exact and predicted solutions for $u(x,t)$ (top) and $v(x,t)$ (bottom).



(a) Error dynamics of PINN (b) Error dynamics of TPINN

(c) Error dynamics of PINN (d) Error dynamics of TPINN

Figure 5. Potential Burgers equations: absolute errors of the PINN and TPINN for the exact solutions $u(x,t)$ (top) and $v(x,t)$ (bottom).



(a) $u(x,t)$ (b) close view of (a)

(c) $v(x,t)$ (d) close view of (c)

Figure 6. Potential Burgers equations: comparison of the exact solution and the solutions solved by the PINN and TPINN.

Table 6. Potential Burgers equations: the relative L^2 errors of the PINN and TPINN with different parameters n for different numbers of training points N_f.

Solution	N_f	PINN	TPINN		
			$n = 10$	$n = 30$	$n = 60$
u	1000	1.3852e-03	7.5867e-04	2.8645e-04	4.2065e-04
	2000	1.6522e-03	1.0243e-03	6.7773e-04	4.8766e-04
	4000	9.5945e-04	5.1565e-04	6.8839e-04	5.8296e-04
v	1000	5.9970e-04	7.4379e-04	2.6666e-04	3.9838e-04
	2000	1.6236e-03	9.4908e-04	2.8057e-04	4.8761e-04
	4000	7.8131e-04	4.4148e-04	6.6338e-04	4.4545e-04

5.3. Reaction equation

Here, we develop an efficient technique combining the time domain decomposition with the TPINN, which is called TPINN-q_{max}, to explore the impact of the number q_{max} of the time intervals on the performance of TPINN. For this purpose, we consider the reaction equation with a large reaction coefficient as follows:

$$u_t - \rho u(1-u) = 0, (x,t) \in (0, 2\pi) \times (0, 1], \tag{5.10}$$

with initial condition

$$u(x, 0) = u_0(x), x \in [0, 2\pi], \tag{5.11}$$

and boundary condition

$$u(0, t) = u(2\pi, t), t \in [0, 1], \tag{5.12}$$

where $u_0(x) = \exp(-\frac{(x-\pi)^2}{\pi^2})$, $\rho > 5$ is the reaction coefficient, and the analytical solution is

$$u(x,t) = \frac{u_0(x)\exp(\rho t)}{u_0(x)\exp(\rho t) + 1 - u_0(x)}. \tag{5.13}$$

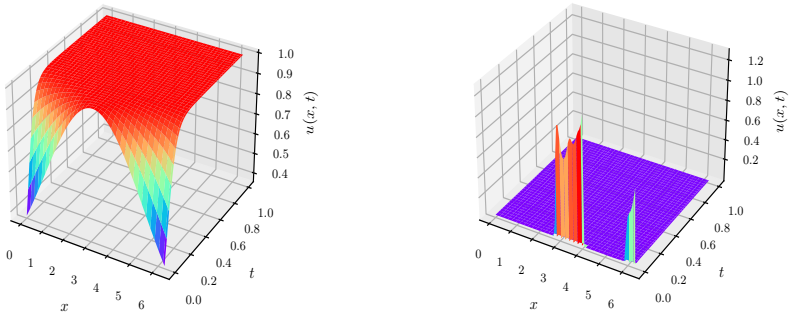
In Table 7, we list the length of the time intervals used in our proposed methods and the size of the datasets. Due to the failure of the standard PINN in accurately solving the reaction equation, the weight w_f is set to zero and all other weights are set to $w_i = w_b = w_{\mathcal{F}} = 1$. We set the parameter $n = 20$ in the proposed TPINN frameworks.

Table 7. Reaction equation: description of the datasets and the length of the time intervals of the PINN and TPINN-q_{max}.

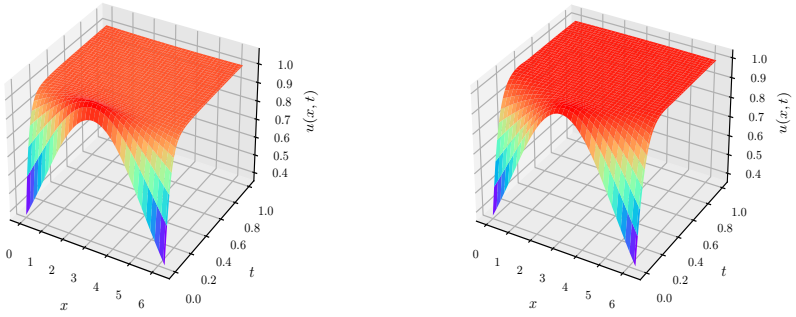
Method	Time interval	N_f	N_i	N_b
PINN	[0, 1.0]	6000	50	100
TPINN	[0, 1.0]	6000	50	100
TPINN-2	[0, 0.4]	3000	50	40
	[0.4, 1.0]	3000	50	60
TPINN-3	[0, 0.4]	2000	50	40
	[0.4, 0.7]	2000	50	30
	[0.7, 1.0]	2000	50	30
TPINN-4	[0, 0.4]	1500	50	40
	[0.4, 0.6]	1500	50	20
	[0.6, 0.8]	1500	50	20
	[0.8, 1.0]	1500	50	20

We present the distribution of the exact and predicted solutions in Figure 7 and absolute error density diagrams in Figure 8. Moreover, a more intuitive demonstration of our proposed methods is provided in Figure 9. To quantitatively illustrate that the Taylor expansion term can more effectively guide the optimization process, Figure 10 presents comparison of L^2 norm of the loss gradients for the TPINN and the PINN. The results indicate that although the gradient norm of the total loss function in the PINN decreases, the gradient norm of its PDE residual loss remains almost constant at around 10^0 and fails to decrease further. Due to the presence of sharp variations in the solution, the network fails to accurately capture the physical laws described by the PDE during the training process. The Taylor expansion term provides a constraint within a local region for the network, making the predicted values and their derivatives closely related between adjacent points. This is a rigid and discrete physical law. TPINN embeds this causal temporal relationship into the loss function. During the training process, the norm of $\nabla_\theta \mathcal{L}_\mathcal{F}$ in the TPINN can smoothly decrease, indicating that the network parameters have been effectively optimized and can accurately capture physical laws.

We present the resulting errors of the TPINN at different reaction coefficients ρ in Table 8, which illustrates that as the parameter n is increased, the accuracy of the proposed TPINN to predict the exact solution is improved. Figure 11 presents the variation trend of the relative L^2 errors obtained by the PINN, TPINN, and TPINN-2 as the reaction coefficient ρ is increased. We can observe that the TPINN-2 performs better than the standard PINN and the proposed TPINN. The results presented in Table 9 indicate that we need to set the number of time intervals to $q_{max} = 3$ or even $q_{max} = 4$ in order to achieve a better predictive effect at $\rho = 100$ or 200. For this example, we test the sensitivity of TPINN-2 to the window length in Table 10 and observe that the window length of 0.4 results in higher prediction accuracy.

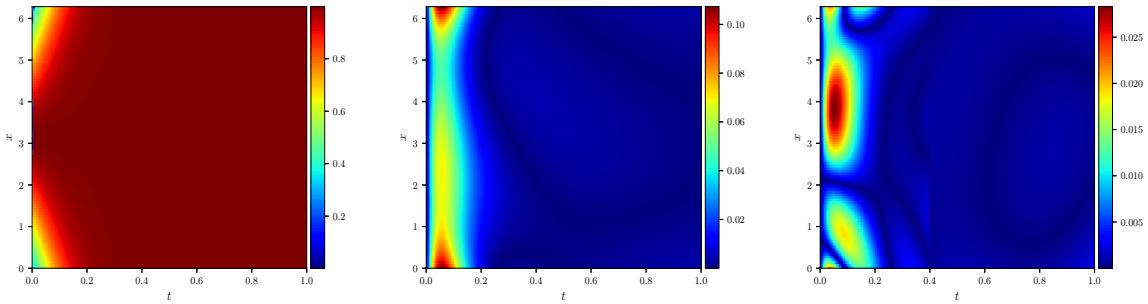


(a) Exact solution (b) PINN



(c) TPINN (d) TPINN-2

Figure 7. Reaction equation: comparison between the exact solution and the solutions solved by the PINN, TPINN, and TPINN-2 at $\rho = 20$.



(a) Error dynamics of PINN (b) Error dynamics of TPINN (c) Error dynamics of TPINN-2

Figure 8. Reaction equation: absolute errors of the PINN, TPINN, and TPINN-2 with the coefficient $\rho = 20$.

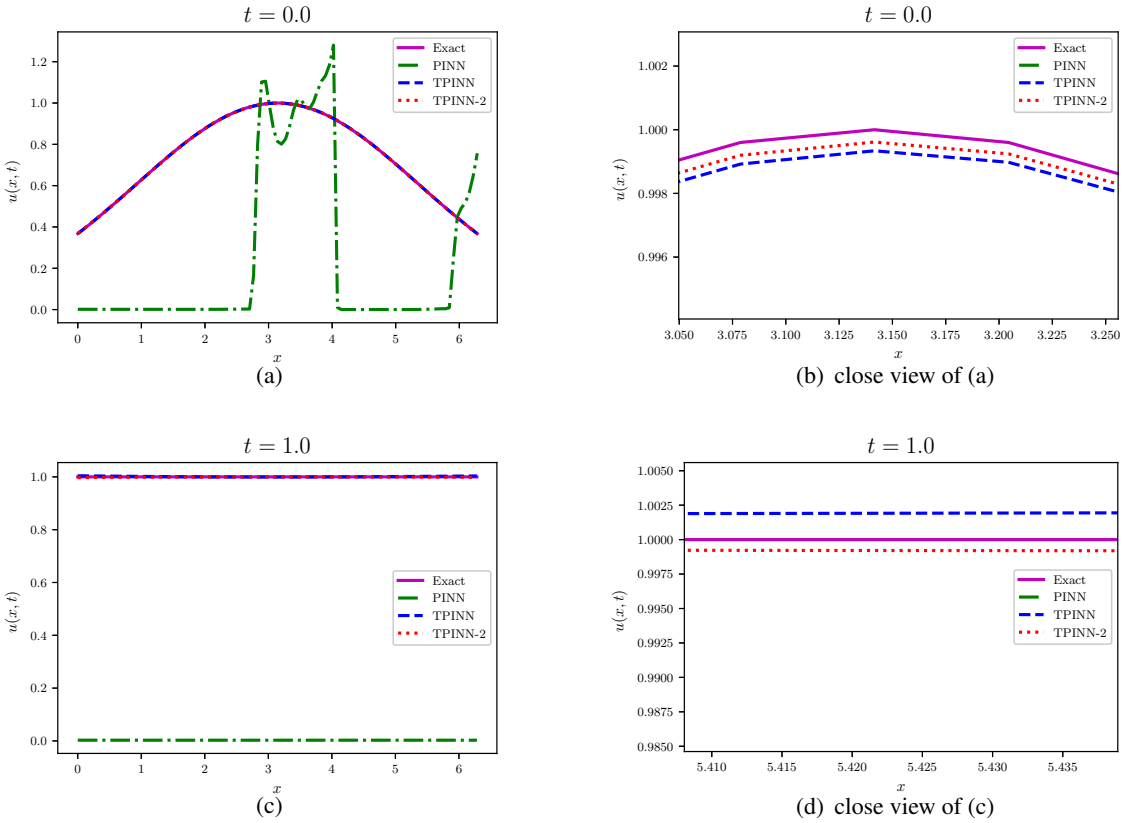


Figure 9. Reaction equation: the snapshots of the exact solution and the solutions solved by PINN, TPINN, and TPINN-2.

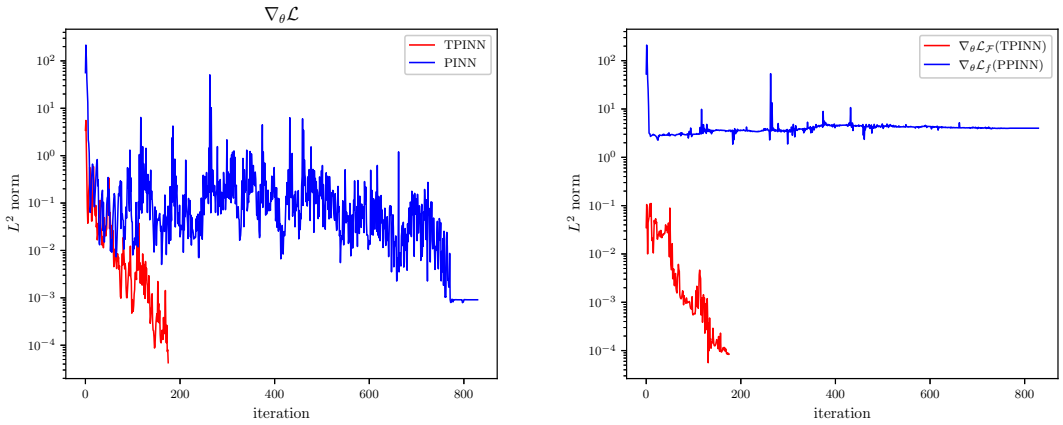


Figure 10. Reaction equation: the evolution diagram of the loss gradients from the TPINN versus the standard PINN.

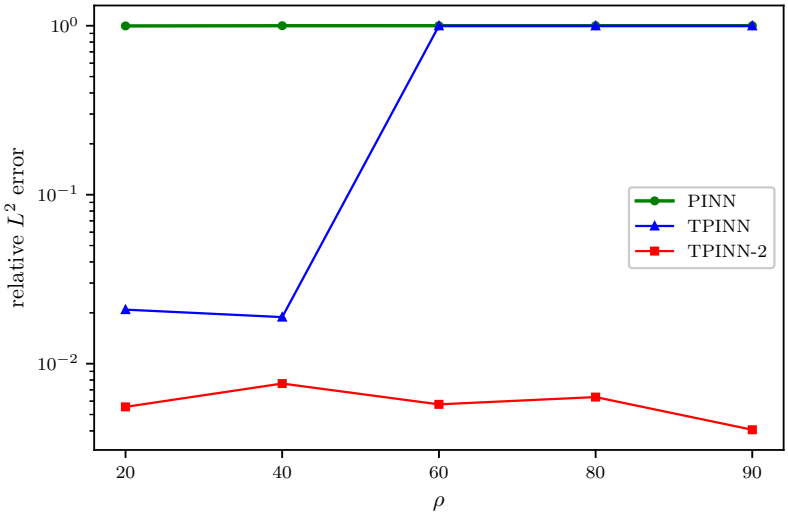


Figure 11. Reaction equation: relative L^2 errors of the PINN, TPINN, and TPINN-2 with different reaction coefficients ρ.

Table 8. Reaction equation: the relative L^2 errors of the PINN and TPINN with different parameters n for different coefficients ρ.

ρ	PINN	TPINN		
		$n = 10$	$n = 20$	$n = 30$
20	9.9691e-01	1.3084e-02	2.0875e-02	1.3640e-02
35	9.9965e-01	9.9767e-01	8.0152e-03	7.3878e-03
40	9.9972e-01	9.9784e-01	1.8859e-02	6.0290e-03
45	9.9976e-01	9.9877e-01	9.9639e-01	7.9848e-03
50	9.9980e-01	9.9784e-01	9.9829e-01	7.4312e-03

Table 9. Reaction equation: the relative L^2 errors of the TPINN with different numbers of time intervals for different parameters n and ρ.

n	ρ	TPINN	TPINN-2	TPINN-3	TPINN-4
20	100	9.9727e-01	7.7157e-01	3.6382e-03	4.1501e-03
40	200	9.9746e-01	9.9742e-01	9.9853e-01	7.9976e-03

Table 10. Reaction equation: the relative L^2 errors of the TPINN-2 with different progressive window lengths at $n = 20$ and $\rho = 20$.

First interval	Second interval	TPINN-2
[0, 0.2]	[0.2, 1.0]	6.6424e-03
[0, 0.4]	[0.4, 1.0]	5.5491e-03
[0, 0.6]	[0.6, 1.0]	6.1065e-03
[0, 0.8]	[0.8, 1.0]	1.3310e-02

5.4. Allen-Cahn (AC) equation

For illustrating the performance of the FP-TPINN method in solving strongly nonlinear evolution PDEs, we aim to solve the following AC equation:

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, (x, t) \in (-1, 1) \times (0, 1], \tag{5.14}$$

with initial condition

$$u(x, 0) = x^2 \cos(\pi x), x \in [-1, 1], \tag{5.15}$$

and boundary conditions

$$\begin{aligned} u(-1, t) &= u(1, t), \quad t \in [0, 1], \\ u_x(-1, t) &= u_x(1, t), \quad t \in [0, 1]. \end{aligned} \tag{5.16}$$

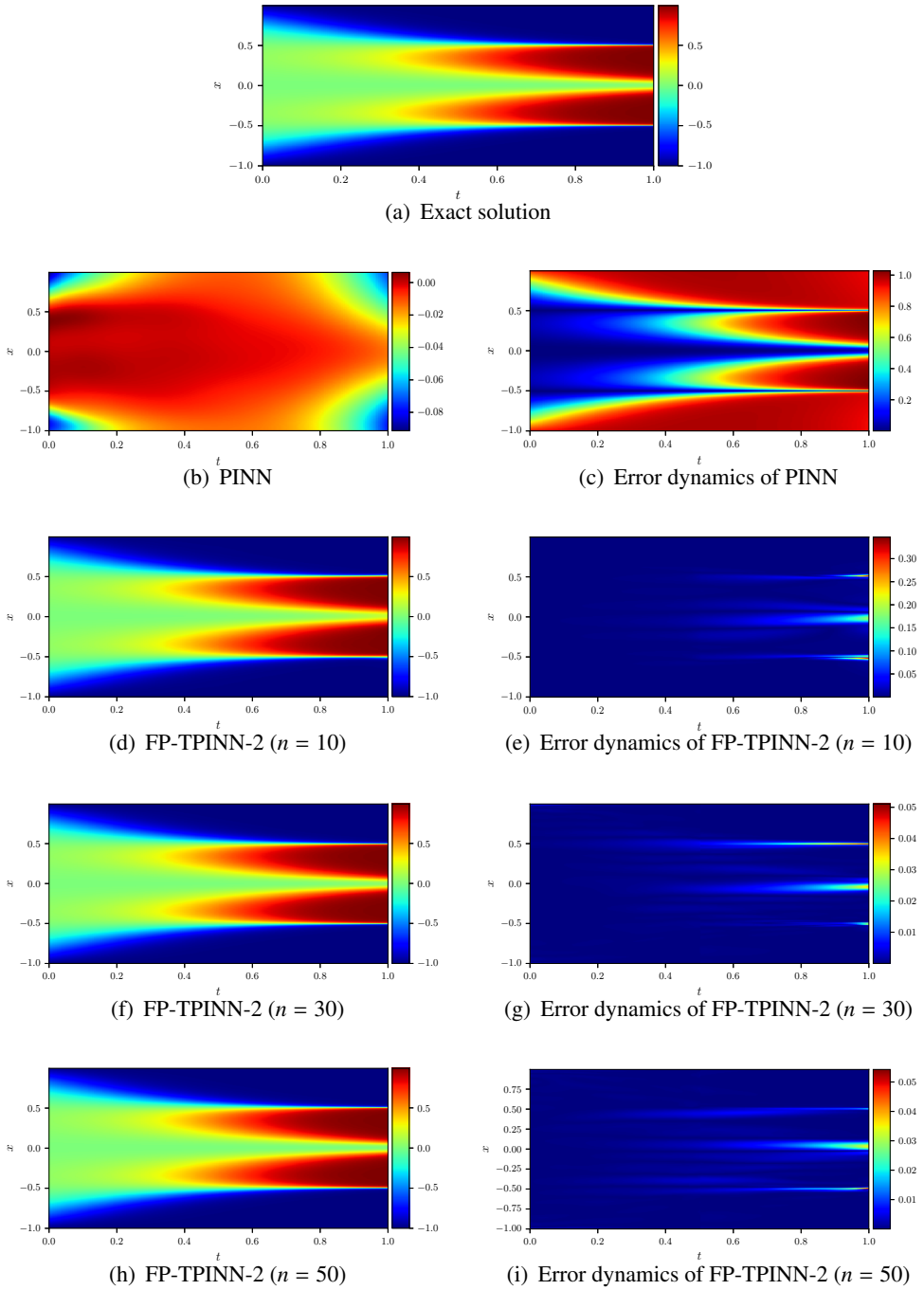
We set $\ell = 200$, $\zeta = 4000$ and consider n, w_i w_b, w_f, and $w_{\mathcal{F}}$ to 50, 100, 1, 0, 200, respectively. Table 11 lists the number of the training points used in the proposed FP-TPINN-2 framework. In this example, we replace $u(x, t - \Delta t)$ in (3.4) with $\frac{1}{5}(u_t - 0.0001u_{xx} + 5u^3)$ at time $t - \Delta t$.

Table 11. AC equation: description of the datasets and the length of time intervals.

Method	Time interval	N_f	N_i	N_b	N_{rf}	N_{rb}
PINN	[0, 1]	2000	200	200	-	-
TPINN	[0, 1.0]	2000	200	200	-	-
FP-PINN-2	[0,0.505]	1000	200	100	800	100
	[0.505,1.0]	1000	200	100	800	100
FP-TPINN-2	[0, 0.505]	1000	200	100	800	100
	[0.505, 1.0]	1000	200	100	800	100

We plot the dynamic change of the prediction and the absolute error in Figure 12 and present a comparison between the exact solution and the prediction in Figure 13, which reveals that the predicted solutions provided by our proposed methods are closer to the exact solution.

In Table 12, we conduct ablation studies and test the performance of the TPINN, the PINN combined with the forward progressive training strategy (FP-PINN), and the FP-TPINN with $q_{max} = 2$ time intervals (FP-TPINN-2) separately. We additionally present the results of the PT-PINN method [43] for this example in Table 12.



(a) Exact solution

(b) PINN

(c) Error dynamics of PINN

(d) FP-TPINN-2 (n = 10)

(e) Error dynamics of FP-TPINN-2 (n = 10)

(f) FP-TPINN-2 (n = 30)

(g) Error dynamics of FP-TPINN-2 (n = 30)

(h) FP-TPINN-2 (n = 50)

(i) Error dynamics of FP-TPINN-2 (n = 50)

Figure 12. AC equation: predictions and absolute errors of the PINN and FP-TPINN-2 with different parameters n.

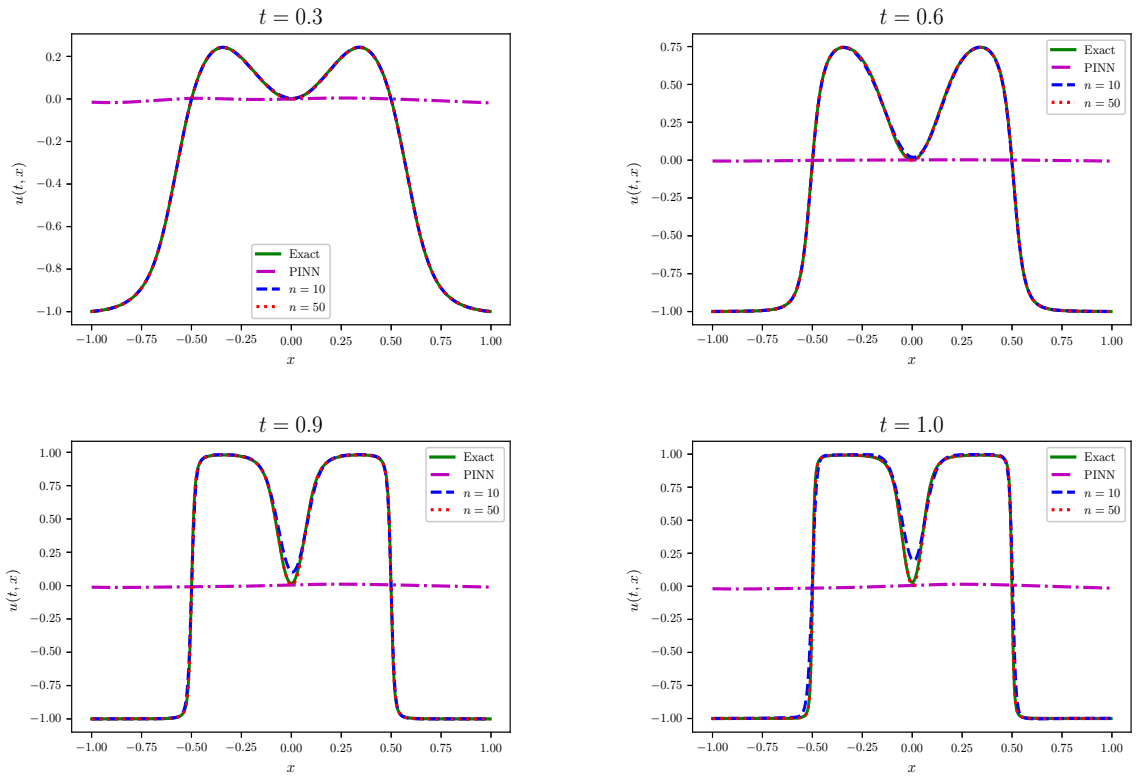


Figure 13. AC equation: the snapshots of the exact solutions and the solutions solved by the PINN and FP-TPINN-2 with the parameters $n = 10, 50$.

Table 12. AC equation: the relative L^2 errors of the PINN, PT-PINN, and FP-TPINN-2 with different parameters n.

Method	N_f	Relative L^2 error
PINN	2000	9.8507e-01
TPINN ($n = 10$)	2000	9.9770e-01
TPINN ($n = 30$)	2000	9.9950e-01
TPINN ($n = 50$)	2000	9.9959e-01
FP-PINN-2	2000	7.8545e-02
FP-TPINN-2 ($n = 10$)	2000	2.1993e-02
FP-TPINN-2 ($n = 30$)	2000	4.0156e-03
FP-TPINN-2 ($n = 50$)	2000	3.6964e-03
PINN	4000	5.1999e-01
TPINN ($n = 50$)	4000	9.9962e-01
FP-TPINN-2 ($n = 50$)	4000	2.6815e-03
PT-PINN	4000	5.1456e-03

When solving the AC equation with strong nonlinearity, the TPINN failed to effectively predict the exact solution, and the FP-PINN-2 has good prediction accuracy. Moreover, the proposed FP-TPINN-2 is also applied to solve another AC model existed in [46], and the corresponding test errors of the DaPINN [46] and the FP-TPINN-2 are listed in Table 13. It is observed that our proposed FP-TPINN method can achieve better prediction accuracy than either the PT-PINN method or the DaPINN method. Finally, the sensitivity analysis of the FP-TPINN-2 to progressive window length is reported in Table 14, which controls the accuracy of the initial condition pushing outward. A larger window length tends to result in poor fitting results. When solving the AC equation, the first window length of the FP-TPINN-2 performs better at 0.4.

Table 13. AC equation reported in [46]: the relative L^2 errors of the PINN, DaPINN and FP-TPINN-2.

Method	N_f	Relative L^2 error
PINN	2000	5.4717e-01
DaPINN (Fourier)	2000	1.5390e-02
DaPINN (x^3)	2000	1.4418e-02
FP-TPINN-2 ($n = 50$)	2000	4.2976e-03

Table 14. AC equation: the relative L^2 errors of the FP-TPINN-2 with different progressive window lengths at $n = 50$.

First interval	Second interval	FP-TPINN-2
[0, 0.2]	[0.2, 1.0]	6.2421e-03
[0, 0.4]	[0.4, 1.0]	4.3043e-03
[0, 0.6]	[0.6, 1.0]	6.7707e-03
[0, 0.8]	[0.8, 1.0]	3.0110e-02

5.5. Wave equation

Here, we consider a one-dimensional wave equation as follows:

$$u_{tt} - C^2 u_{xx} = 0, (x, t) \in (0, 1) \times (0, 1], \tag{5.17}$$

with initial conditions

$$\begin{aligned} &u(x, 0) = \sin(\pi x) + \sin(A\pi x), \quad x \in [0, 1], \\ &u_t(x, 0) = 0, \qquad\qquad\qquad\quad x \in [0, 1], \end{aligned} \tag{5.18}$$

and boundary conditions

$$u(0, t) = u(1, t) = 0, t \in [0, 1], \tag{5.19}$$

where $A = 3$, $C = 2$, and the analytical solution $u(x, t) = \sin(\pi x)\cos(C\pi t) + \sin(A\pi x)\cos(AC\pi t)$.

To apply the FP-TPINN method, the governing equation (5.17) is rewritten as

$$v_t - C^2 u_{xx} = 0, \tag{5.20}$$

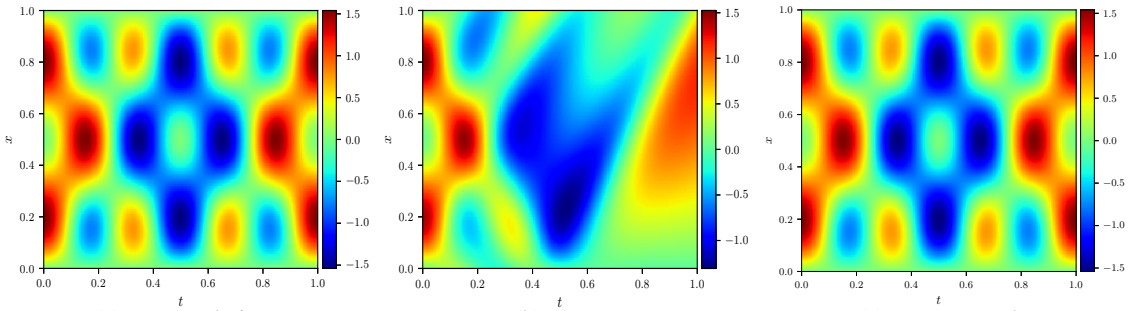
where $v = u_t$.

Table 15 shows the range of sampling intervals for FP-TPINN-2 and the size of the datasets. We set the parameter $n = 50$. The weights w_i, w_b, w_f, and $w_{\mathcal{F}}$ are set as 1, 100, 0, and 200, respectively.

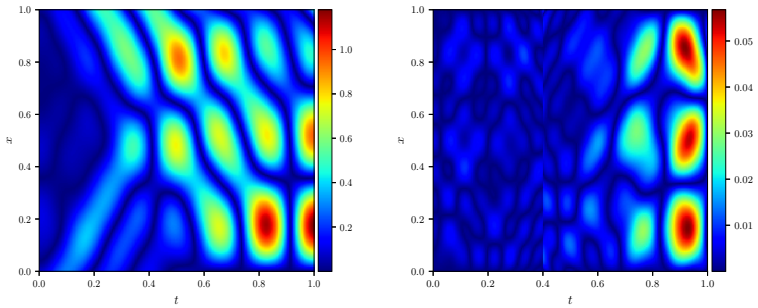
Table 15. Wave equation: description of the datasets and the length of time intervals of the PINN and FP-TPINN-2.

Method	Time interval	N_f	N_i	N_b	N_{rf}	N_{rb}
PINN	[0, 1.0]	2000	200	200	-	-
TPINN	[0, 1.0]	2000	200	200	-	-
FP-TPINN-2	[0, 0.4]	1000	200	100	800	0
	[0.4, 1.0]	1000	200	100	800	0

We present in Figure 14 the density diagrams of the predictions obtained by the PINN and the proposed FP-TPINN-2 and the corresponding error density diagrams, and we show in Figure 15 a comparison of the PINN and the proposed FP-TPINN-2 to provide an intuitive assessment, which confirms that the proposed method has better resolution than the standard PINN.



(a) Exact solution (b) PINN (c) FP-TPINN-2



(d) Error dynamics of PINN (e) Error dynamics of FP-TPINN-2

Figure 14. Wave equation: (top) comparison between the exact solution and the solutions solved by the PINN and FP-TPINN-2; (bottom) absolute errors of the PINN and FP-TPINN-2.

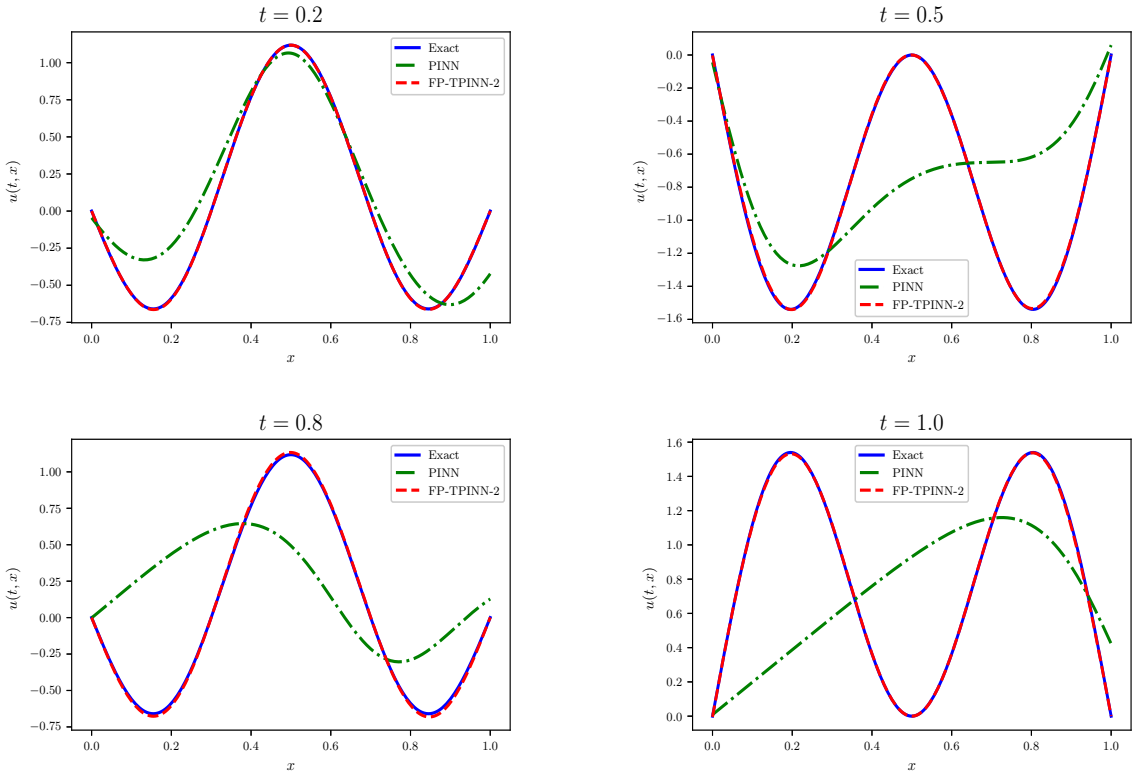


Figure 15. Wave equation: the snapshots of the exact solutions and the solutions solved by the PINN and FP-TPINN-2.

Table 16 summarizes the relative L^2 errors, memory costs and training time of the PINN and the proposed FP-TPINN-2 with different numbers of training points N_f. We find that the PINN spend a lot of training time with the number of collocation points $N_f = 3000$, which is almost twice that of the FP-TPINN.

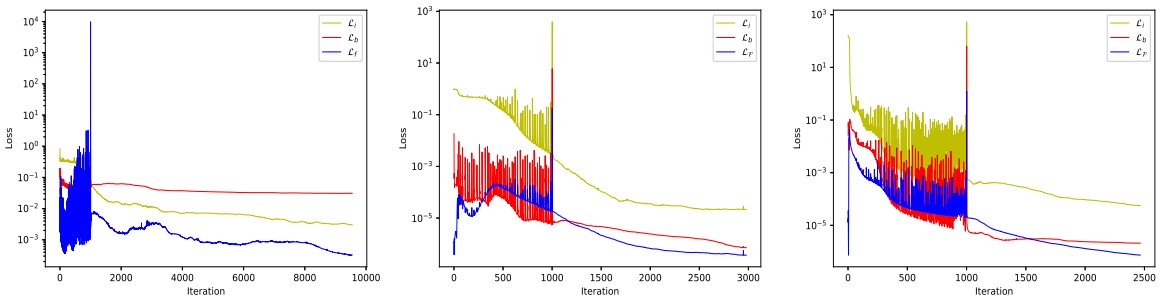
Table 16. Wave equation: the training results of the PINN and FP-TPINN-2 with different parameters n for different numbers of training points N_f.

	N_f	PINN	FP-TPINN-2 ($n = 30$)	FP-TPINN-2 ($n = 50$)
Relative L^2 error	1600	5.0319e-01	5.6001e-02	1.8498e-02
	2000	5.0724e-01	8.5455e-02	1.8544e-02
	3000	5.0711e-01	5.5911e-02	2.0080e-02
Training time(s)	1600	386	323	347
	2000	410	410	465
	3000	1073	616	607
Memory cost(MB)	1600	276	407	405
	2000	279	374	418
	3000	271	451	452

Table 17 reports the comparative analysis of the TPINN and the FP-TPINN. It can be observed that the TPINN can also solve the wave equation with stiff behavior, and the FP-TPINN can further improve the accuracy of the TPINN. According to the dynamic changes shown in Figure 16, we find that although the loss term $\mathcal{L}_b$ has a decreasing trend during the training process of the PINN, the magnitude of the decrease is small, which causes the PINN to spend too much time minimizing the loss function during the training process. However, it can be observed that the existence of the loss term $\mathcal{L}_{\mathcal{F}}$ effectively optimizes the parameters of the network during the training process of the first interval ($[0, 0.4]$) and the second interval ($[0.4, 1.0]$) of the TPINN, resulting in a comparatively steady downward trend for each loss term.

Table 17. Wave equation: relative L^2 errors of the TPINN and the FP-TPINN-2 with different parameters n for different numbers of training points N_f.

N_f	TPINN-2 ($n = 30$)	TPINN-2 ($n = 50$)	FP-TPINN-2 ($n = 30$)	FP-TPINN-2 ($n = 50$)
1600	5.3308e-02	2.4325e-02	5.6001e-02	1.8498e-02
2000	5.6372e-02	8.2527e-02	8.5455e-02	1.8544e-02
3000	5.7562e-02	7.6930e-02	5.5911e-02	2.0080e-02



(a) PINN (b) FP-TPINN-2 ([0, 0.4]) (c) FP-TPINN-2 ([0.4, 1.0])

Figure 16. Wave equation: the dynamic variation of the loss term of the PINN and FP-TPINN-2 ($n = 50$) with respect to the number of iterations.

5.6. *Improved Boussinesq equation*

In this example, let us investigate the following improved Boussinesq equation:

$$u_{tt} - u_{xx} - u_{xxtt} - (u^2)_{xx} = 0, (x, t) \in (x_l, x_r) \times J, \tag{5.21}$$

with initial conditions

$$u(x, 0) = u_0(x), \quad u_t(x, 0) = g(x), \quad x \in [x_l, x_r], \tag{5.22}$$

and periodic boundary conditions

$$\begin{aligned} u(x_l, t) &= u(x_r, t), \qquad t \in J, \\ u_x(x_l, t) &= u_x(x_r, t), \qquad t \in J, \\ u_{xx}(x_l, t) &= u_{xx}(x_r, t), \quad t \in J. \end{aligned} \tag{5.23}$$

In this example, we consider the single solitary wave case with the analytical solution defined by

$$u(x,t) = \alpha \text{sech}^2\left(\sqrt{\frac{\alpha}{6}}\frac{x-\beta t - x_0}{\beta}\right), \tag{5.24}$$

where $\alpha = 0.5$, $x_0 = 0, \beta = \sqrt{1+\frac{2\alpha}{3}}$, $J = (0, 1]$, and $[x_l, x_r] = [-100, 100]$. The corresponding initial conditions are given by

$$u_0(x) = \alpha \text{sech}^2\left(\sqrt{\frac{\alpha}{6}}\frac{x-x_0}{\beta}\right), \tag{5.25}$$

and

$$g(x) = 2\alpha\sqrt{\frac{\alpha}{6}}\text{sech}^2\left(\sqrt{\frac{\alpha}{6}}\frac{x-x_0}{\beta}\right)\tanh\left(\sqrt{\frac{\alpha}{6}}\frac{x-x_0}{\beta}\right). \tag{5.26}$$

Due to the significant increase in training time caused by the derivative of u_{xxtt} with respect to time t, we adopt a different calculation method from Subsection 5.5 to define $\mathcal{F}(x,t)$:

$$\mathcal{F}(x,t) = u(x,t) - \left(u(x,t-\Delta t) + \Delta t\frac{\partial u}{\partial t}(x,t-\Delta t) + \frac{\Delta t^2}{2!}\mathcal{D}[u(x,t-\Delta t)]\right), \tag{5.27}$$

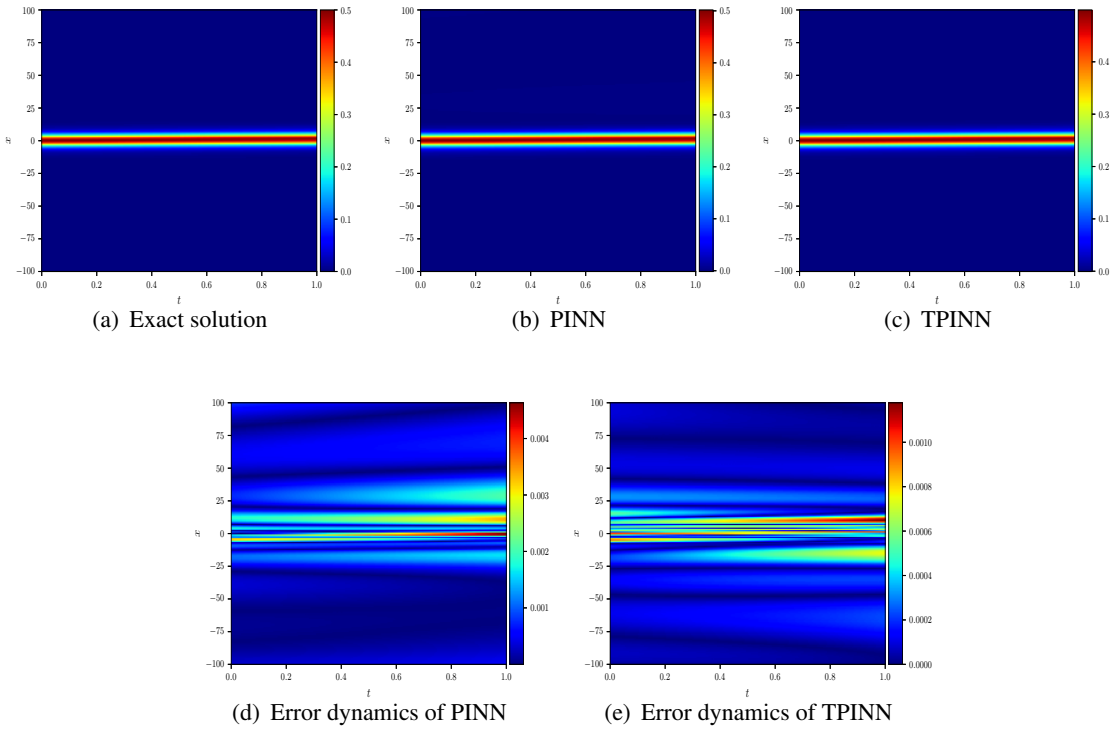
where $\mathcal{D}[u(x,t)] = u_{xx} + u_{xxtt} + (u^2)_{xx}$.

For this example, we use the training data that consists of $N_f = 4000$ residual points sampled from the spatiotemporal domain, $N_i = 100$ initial points, and $N_b = 100$ boundary points. The weights of loss components are set to $w_i = w_b = w_f = w_{\mathcal{F}} = 1$ and the parameter n is set as 30.

The training results are summarized in Figure 17, and the snapshots shown in Figure 18 demonstrate that the results given by the TPINN are closer to the exact solution. We report the relative L^2 errors, final values of the loss terms, training time and memory costs given by the standard PINN, and the proposed TPINN in Table 18, indicating that the TPINN method significantly improves the prediction accuracy. The proposed TPINN introduces the loss term $\mathcal{L}_{\mathcal{F}}$ as an additional supervised learning task, which allows the network to better learn the physical laws described by evolution PDEs, but also increases the difficulty of training the network. Therefore, the training time required by the TPINN is slightly higher than that of the PINN, increasing from 19 seconds to 54 seconds. However, our proposed TPINN reduces the error from 1.0098e-02 to 2.7457e-03, with an error reduction of approximately 73%, significantly improving the prediction accuracy. Meanwhile, all loss components have been reduced by an order of magnitude. We believe that this is acceptable in scientific computing problems that focus on high precision. The dynamic changes of each loss term in the PINN and TPINN are displayed in Figure 19. Comparing the changes in the PINN and TPINN, we can observe that during the training process of the TPINN, the value of the loss term $\mathcal{L}_{\mathcal{F}}$ is almost always lower than that of other loss terms, and it has a positive impact on the downward trend of the loss $\mathcal{L}_i$, $\mathcal{L}_b$, and $\mathcal{L}_f$ without disrupting their overall trend of change.

Table 18. Improved Boussinesq equation: the results given by the PINN and TPINN.

Method	PINN	TPINN (n = 10)	TPINN (n = 30)
Relative L^2 error	1.0098e-02	6.1460e-03	2.7457e-03
$\mathcal{L}$	1.2066e-06	4.5959e-07	2.0984e-07
$\mathcal{L}_i$	6.1505e-07	2.0954e-07	7.6047e-08
$\mathcal{L}_b$	7.6545e-08	1.1244e-08	1.1404e-09
$\mathcal{L}_f$	5.1496e-07	2.3881e-07	1.3265e-07
$\mathcal{L}_{\mathcal{F}}$	-	1.3063e-12	2.2622e-14
Time(s)	19	43	54
Memory	604 MB	927 MB	931 MB



(a) Exact solution (b) PINN (c) TPINN

(d) Error dynamics of PINN (e) Error dynamics of TPINN

Figure 17. Improved Boussinesq equation: (top) comparison between the exact solution and the solutions solved by the PINN and TPINN; (bottom) absolute error distributions of the PINN and TPINN.

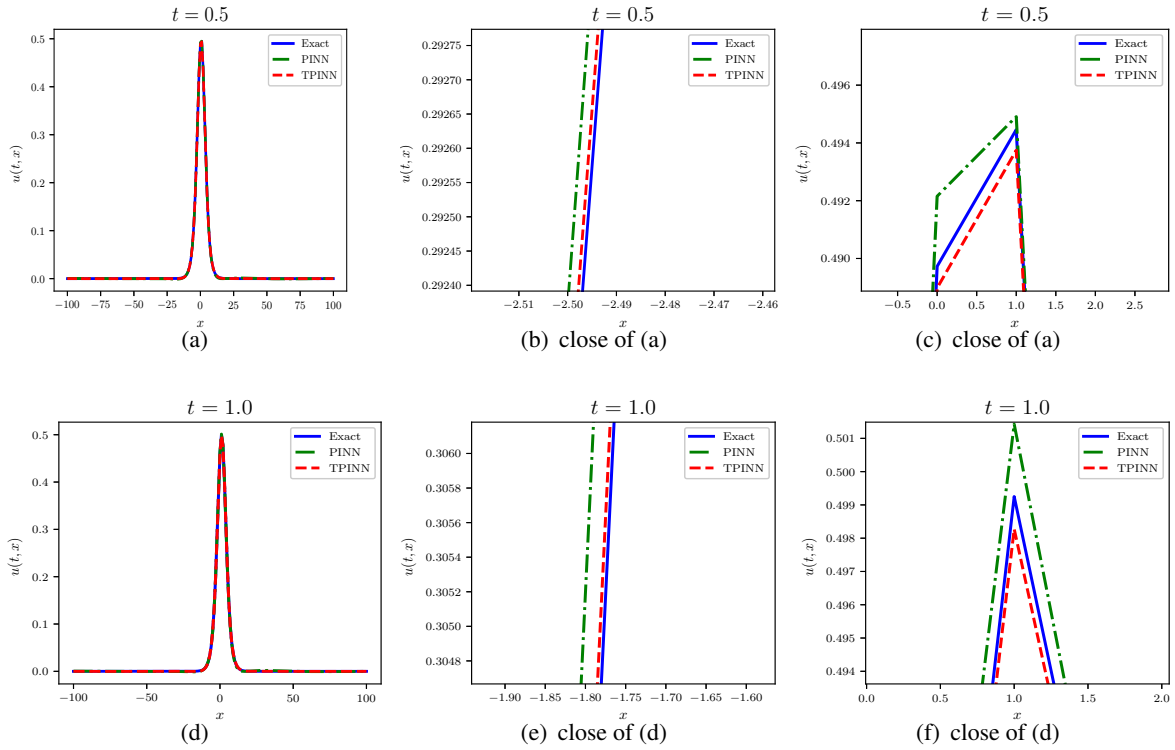


Figure 18. Improved Boussinesq equation: the snapshots of the exact solution and the solutions solved by PINN and TPINN.

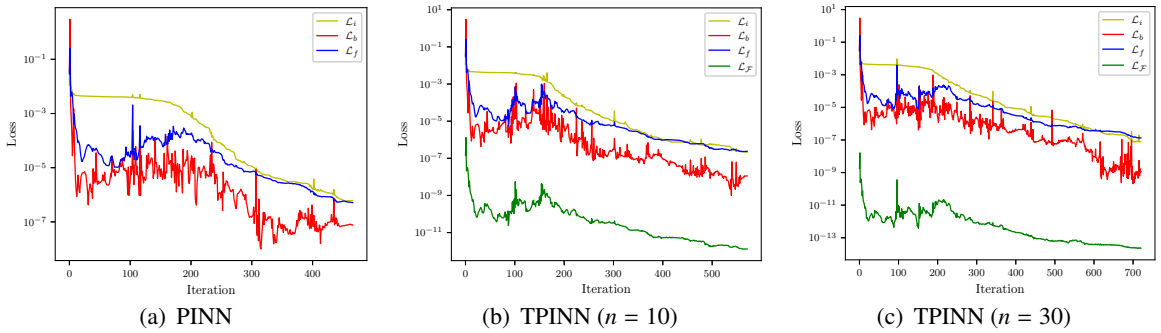


Figure 19. Improved Boussinesq equation: the dynamic variations of the loss terms for the PINN and TPINN with respect to the number of iterations.

5.7. Flow mixing problem

To further test the effectiveness of the proposed TPINN, the flow mixing problem with the Dirichlet boundary condition is as follows:

$$u_t + au_x + bu_y = 0,\ (x, y) \in (-4, 4) \times (-4, 4),\ t \in [0, 4], \tag{5.28}$$

where

$$\begin{aligned} a(x,y) &= -\frac{\nu_t}{\nu_{tmax}}\frac{y}{r}, \qquad b(x,y) = \frac{\nu_t}{\nu_{tmax}}\frac{x}{r}, \\ \nu_t &= \operatorname{sech}^2(r)\tanh(r), \quad r = \sqrt{x^2+y^2}. \end{aligned} \tag{5.29}$$

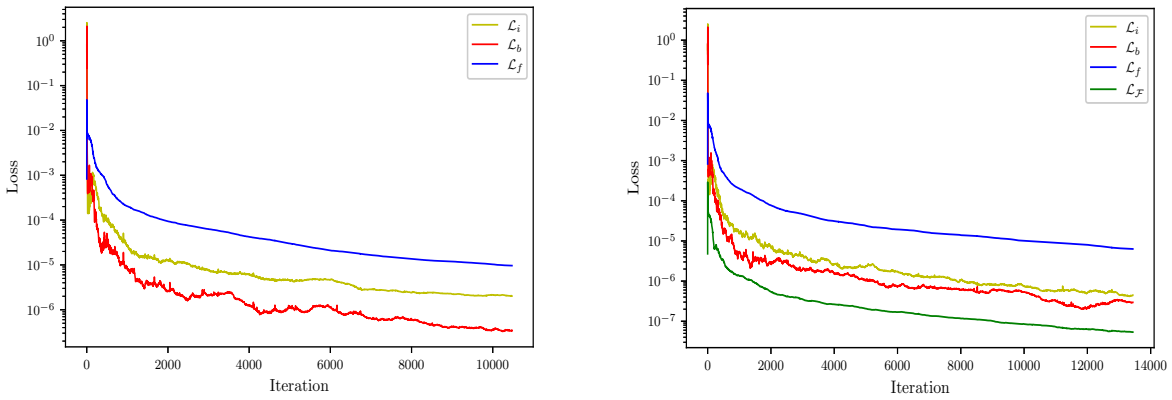
This problem describes the motion process of two fluids with different characteristics, a clod front and a warm front, gradually mixing at a specified speed ν_t in the spatial domain [47, 48]. The analytical solution is

$$u(x,y,t) = -\tanh\left(\frac{y}{2}\cos(\omega t) - \frac{x}{2}\sin(\omega t)\right), \tag{5.30}$$

where $\omega = \frac{1}{r}\frac{\nu_t}{\nu_{tmax}}$ and $\nu_{tmax} = 0.385$. The initial condition at $t = 0$ is derived from the solution (5.30).

To train the network, we randomly sample $N_f = 2000$ points within the solution domain, $N_b = 200$ data points on the boundary, and $N_i = 100$ initial data points. The weights of each loss is set to $w_i = w_b = w_f = w_{\mathcal{F}} = 1$ and the parameter n is set as 30.

The final mean squared error of each loss term for the PINN and TPINN presented in Table 19 and Figure 20 shows the dynamic change during the training process. It can be seen that the loss terms $\mathcal{L}_i$, $\mathcal{L}_b$, $\mathcal{L}_f$ and the error of the total loss $\mathcal{L}$ obtained by the TPINN are all lower than those of the PINN, indicating that the existence of the loss term $\mathcal{L}_{\mathcal{F}}$ further optimizes the parameters of the network, resulting in the prediction being closer to the exact solution. We present in Figure 21 the density diagrams of the exact solution and the predicted solutions provided by the PINN and TPINN throughout the entire spatial domain. In Figure 22, the error density diagrams of the PINN and TPINN at different times are plotted. The distribution interval of absolute errors given by the TPINN is smaller than that of the PINN, indicating that our proposed method achieves a better prediction.

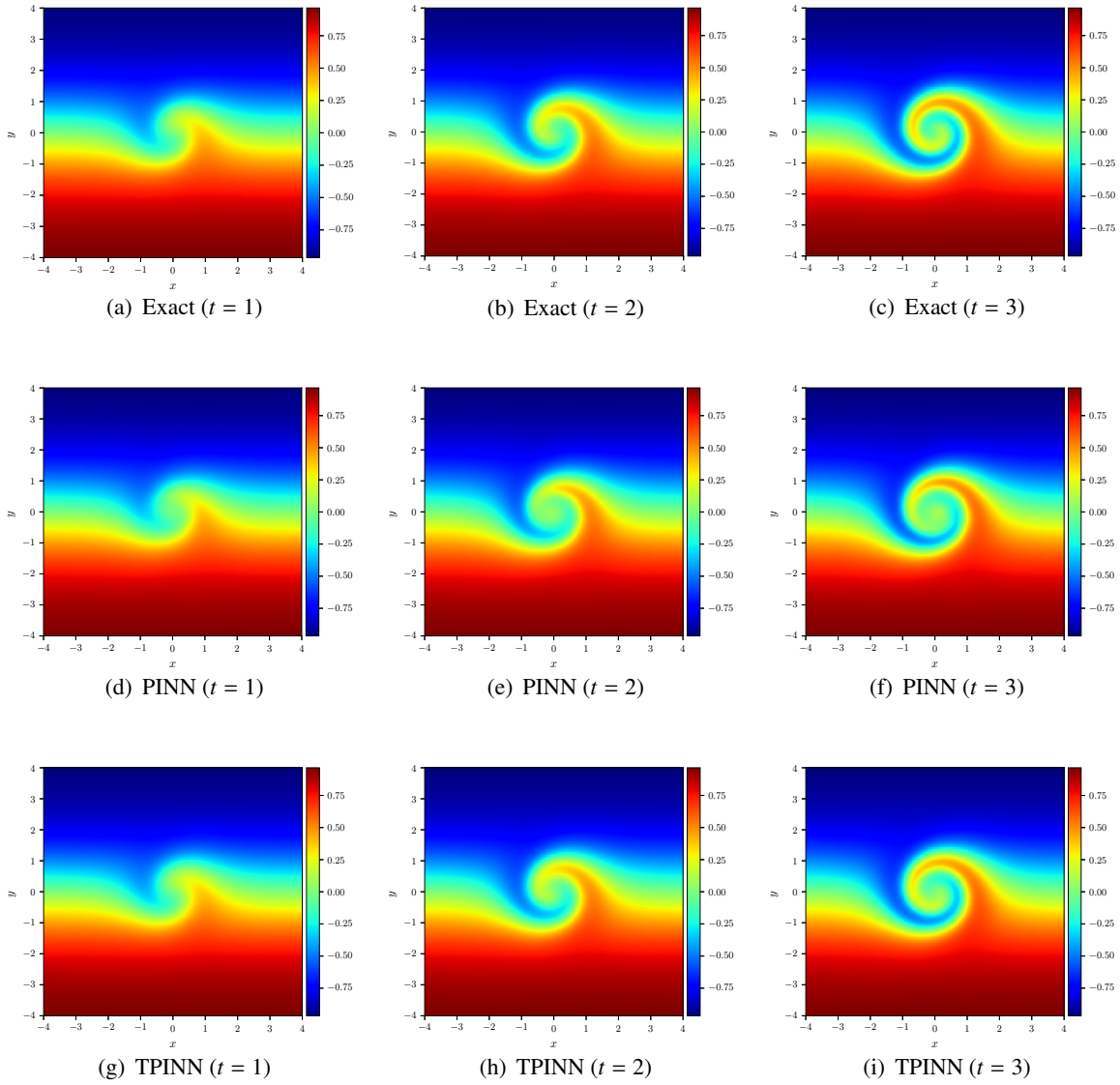


(a) PINN (b) TPINN

Figure 20. Flow mixing problem: the dynamic variations of the loss terms for the PINN and TPINN with respect to the number of iterations.

Table 19. Flow mixing problem: the training results for the PINN and TPINN.

Method	Relative L^2 error	$\mathcal{L}$	$\mathcal{L}_i$	$\mathcal{L}_b$	$\mathcal{L}_f$	$\mathcal{L}_{\mathcal{F}}$	Time(s)	Memory
PINN	1.8752e-02	1.2011e-05	2.0188e-06	3.4129e-07	9.6513e-06	-	177	380 MB
TPINN	1.0699e-02	7.1101e-06	4.4728e-07	2.9558e-07	6.3135e-06	5.3712e-08	268	437 MB

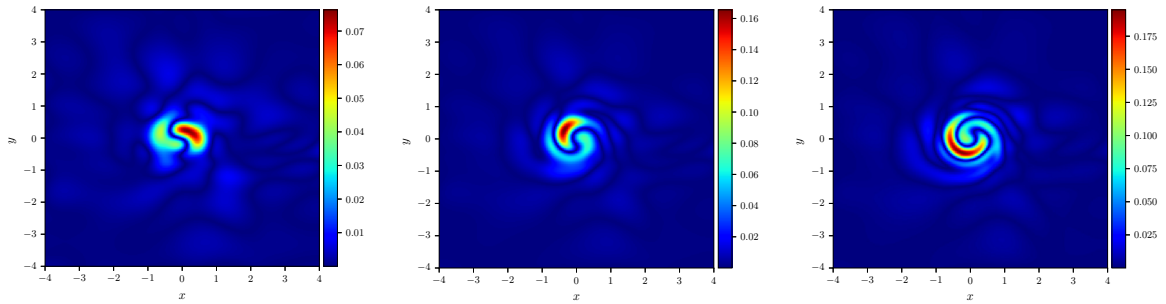


(a) Exact ($t = 1$) (b) Exact ($t = 2$) (c) Exact ($t = 3$)

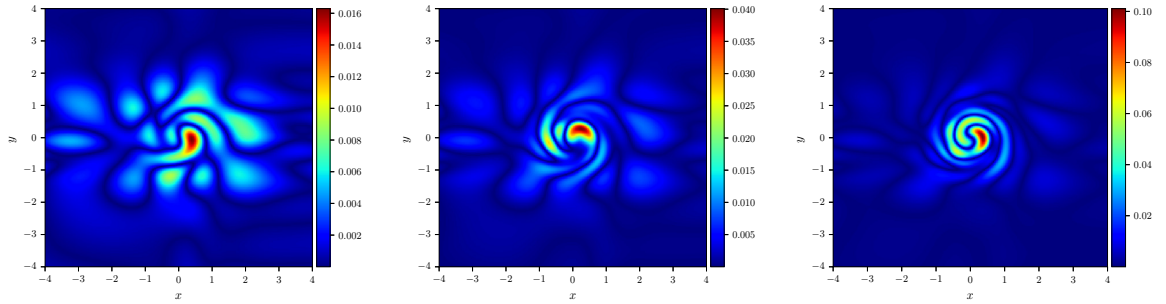
(d) PINN ($t = 1$) (e) PINN ($t = 2$) (f) PINN ($t = 3$)

(g) TPINN ($t = 1$) (h) TPINN ($t = 2$) (i) TPINN ($t = 3$)

Figure 21. Flow mixing problem: comparison between the exact solution (top) and the solutions solved by the PINN (middle) and TPINN (bottom) at different times.



(a) Error dynamics of PINN ($t = 1$) (b) Error dynamics of PINN ($t = 2$) (c) Error dynamics of PINN ($t = 3$)



(d) Error dynamics of TPINN ($t = 1$) (e) Error dynamics of TPINN ($t = 2$) (f) Error dynamics of TPINN ($t = 3$)

Figure 22. Flow mixing problem: the error dynamic diagrams of the PINN and TPINN at different times.

5.8. High-dimensional diffusion-reaction equation

In this example, we aim to demonstrate the performance of the TPINN method in solving high dimensional PDEs. Consider the following five-dimensional diffusion-reaction equation [49]:

$$u_t - \Delta u - u^2 = f(\boldsymbol{x}, t), (\boldsymbol{x}, t) \in \Omega \times (0, T], \tag{5.31}$$

with initial condition

$$u(\boldsymbol{x}, 0) = \prod_{j=1}^{5} \sin(\pi x_j), \boldsymbol{x} \in \overline{\Omega}, \tag{5.32}$$

and boundary condition

$$u(\boldsymbol{x}, t) = e^t \prod_{j=1}^{5} \sin(\pi x_j), (\boldsymbol{x}, t) \in \partial\overline{\Omega} \times [0, T], \tag{5.33}$$

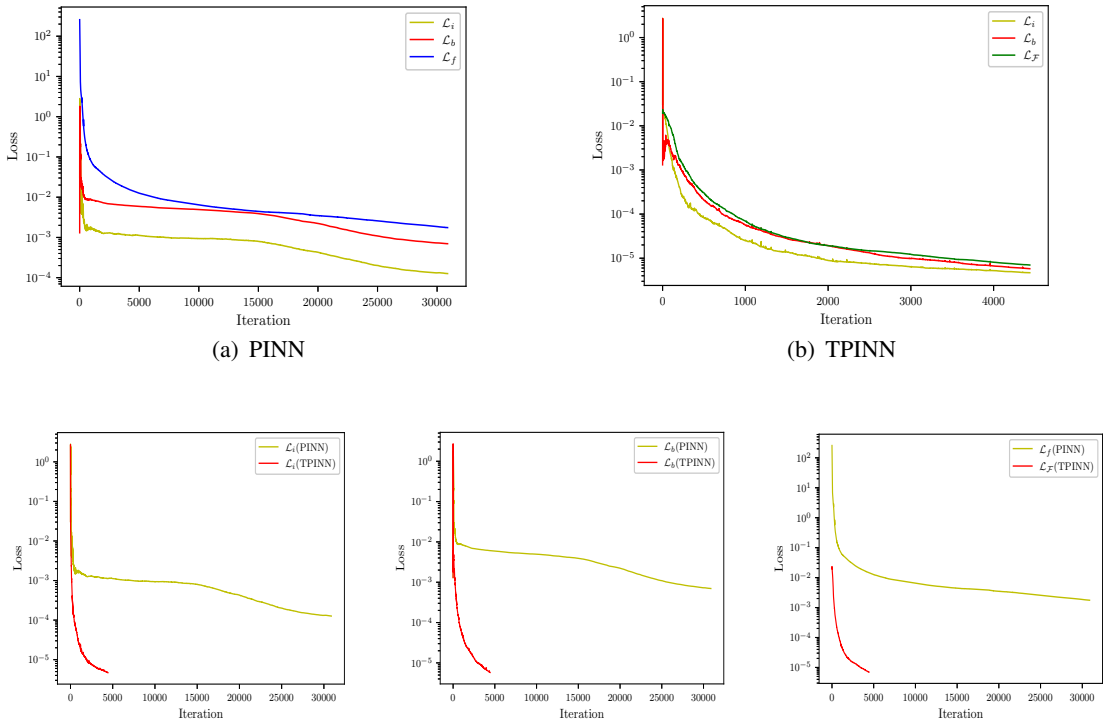
where $\Omega = (0, 1)^5 \subset \mathbb{R}^5$ and the final time $T = 1$. The exact solution is defined as $u(\boldsymbol{x}, t) = e^t \prod_{j=1}^{5} \sin(\pi x_j)$ and the forcing term $f(\boldsymbol{x}, t)$ is derived from the exact solution.

The training data consists of $N_f = 2000$ points randomly sampled from the solution domain, $N_i = 600$ initial points, and $N_b = 500$ boundary points. The parameter n is set to 80 and the weights w_i, w_b, w_f, and $w_{\mathcal{F}}$ are set as 1, 1, 0, and 1, respectively.

Table 20 presents the relative L^2 error, the final value of each loss term, training time, and memory cost, indicating that the TPINN achieves higher prediction accuracy and spends less computational time than the PINN. We also find that the final value of each loss term in the TPINN is lower than that in the PINN. For this purpose, we provide the dynamic variation graph of each loss term during the training process in Figure 23, which reveals that compared to the PINN, the presence of the loss term $\mathcal{L}_{\mathcal{F}}$ in the TPINN results in a faster rate of decrease for the loss terms. Finally, Figure 24 displays a more intuitive comparison.

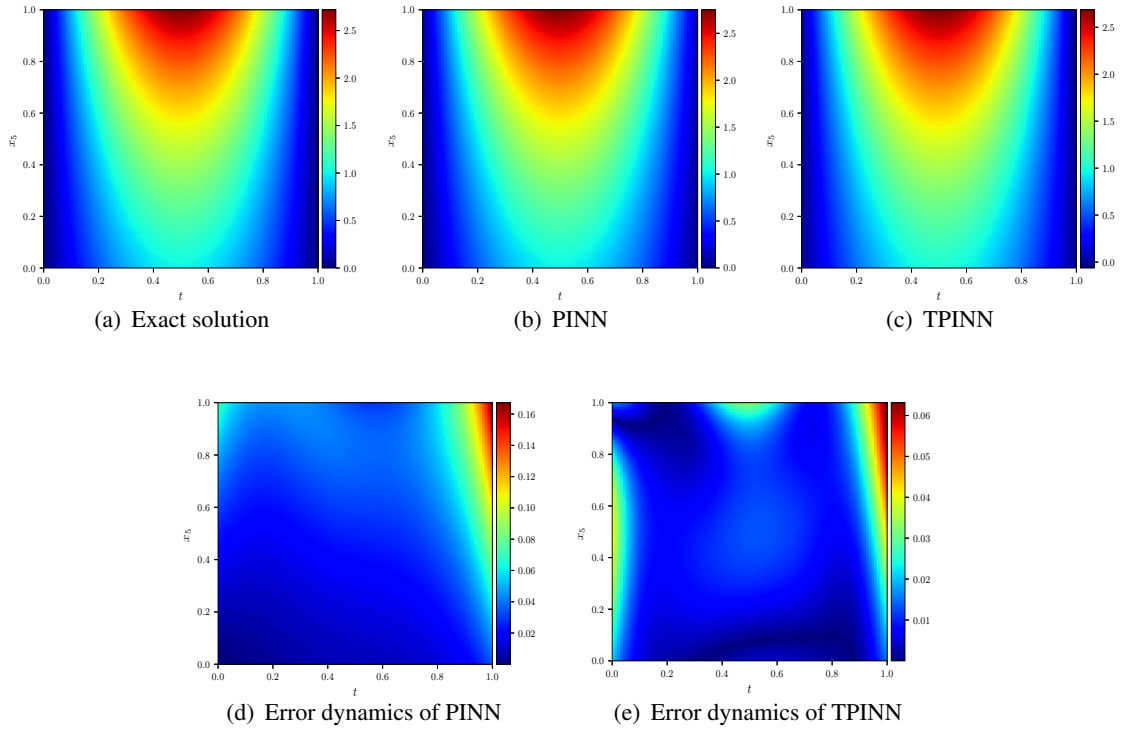
Table 20. Diffusion-reaction equation: the training results for the PINN and TPINN.

Method	Relative L^2 error	$\mathcal{L}$	$\mathcal{L}_i$	$\mathcal{L}_b$	$\mathcal{L}_f$	$\mathcal{L}_{\mathcal{F}}$	Time(s)	Memory
PINN	7.2253e-02	2.5797e-03	1.2648e-04	6.9888e-04	1.7544e-03	-	492	449 MB
TPINN	2.8781e-02	1.7486e-05	4.6917e-06	5.7837e-06	-	7.0110e-06	361	997 MB



(a) PINN (b) TPINN

Figure 23. Diffusion-reaction: the dynamic variations of the loss terms for the PINN and TPINN with respect to the number of iterations.



(a) Exact solution (b) PINN (c) TPINN

(d) Error dynamics of PINN (e) Error dynamics of TPINN

Figure 24. Diffusion-reaction: point-wise density diagrams on the slice of $x_1 = x_2 = x_3 = x_4 = 0.5$; (top) comparison between the exact solution and the solutions solved by the PINN and TPINN; (bottom) absolute errors of the PINN and TPINN.

6. Discussion

The numerical experimental results presented in Section 5 demonstrate that the proposed TPINN and FP-TPINN methods achieve better prediction accuracy than that of the PINN under the same training configuration, such as the number of training points, network structure, and optimizer. Next, we will further analyze the temporal causal relationship in our proposed methods. Moreover, it may not be clear how to divide the length of time intervals and the scenarios applicable to TPINN or FP-TPINN. Therefore, we will provide a detailed explanation in this section.

Temporal causality analysis: The proposed PINN-based framework in this work enhances compliance with the initial condition and temporal causality from two aspects. (1) The local dependency of the loss function based on Taylor expansion improves initial condition adherence and causality. We use the loss term $\mathcal{L}_{\mathcal{F}}$ introduced by Taylor series expansion as a new supervised learning task within the time interval $[T_{q-1}, T_q]$. (2) The FP-TPINN decomposes the time domain into continuous subintervals and enforces the solution of the previous subdomain as the initial condition for the next subdomain, thus rigidly defining the temporal causal relationship from the structure of network training. This strategy not only ensures strict propagation of the initial condition in the global time domain, but also transforms complex global optimization problems into a series of simpler local problems. There are essential differences between other causality-aware PINN frameworks based on

time domain decomposition and our proposed FP-TPINN. The authors in [43] divide the entire time domain $(0, T]$ into k intervals as $(0, T_1]$, $(0, T_2]$, $\cdots$, $(0, T_k]$, $(0, T_{k+1} = T]$ $(T_{k-1} < T_k)$, where the predicted solution on the subdomain $(0, T_{k-1}]$ is provided as extra supervised learning data to the next subdomain $(0, T_k]$. The solutions predicted by all previous intervals ($[0, T_{q-1}]$) are considered as a constraint for training neural networks on interval $[T_{q-1}, T_q]$ [6]. In the proposed FP-TPINN framework, we take the predicted value at time T_{q-1} in the previous interval $[T_{q-2}, T_{q-1}]$ as the initial condition for the next interval $[T_{q-1}, T_q]$. Moreover, we use Taylor expansion term instead of the PDE residual in each interval and combine them with initial and boundary conditions to train the neural network. Compared to the sequence-to-sequence learning method [42], we adopt the resampling method at each interval to train the network.

The standard for dividing the length of time intervals: We control the size of interval division based on the complexity of the variation trend exhibited by the solution in the spatiotemporal domain. In Subsection 5.3, when the entire time interval is divided into two intervals, the length of the first time interval is smaller than that of the second interval. From Figure 7(a), it can be seen that the exact solution of (5.10) has a steep trend of change at the initial stage, which makes it difficult to obtain the forecasting results with high accuracy. Therefore, we choose a relatively small interval to obtain a better prediction. When dividing the entire time domain into three or four subdomains, we choose to divide the time interval [0.6,1.0] equally into two or three intervals, respectively. A small time interval can achieve better training results. The solution of the AC equation (5.14) has a small amplitude of change near the initial time, while the solution of the wave equation (5.17) exhibits a more complex trend of change near the initial time. Therefore, we set the first time interval to [0, 0.505] and [0, 0.4] in Subsections 5.4 and 5.5, respectively, which can provide a more accurate initial condition for the second interval.

Application scenarios of the TPINN and FP-TPINN: The key idea of the proposed TPINN is to add a new supervised learning task to train the network by utilizing Taylor expansion, which aims to improve the accuracy of the PINN. For time-dependent PDEs that can be solved by the PINN, the TPINN can be used to obtain a better prediction. In addition, for the evolution PDEs with stiff behavior, such as the reaction equations with a large model parameter and the wave equation, the prediction given by the PINN deviates far from the exact solution, while the TPINN can still accurately predict the exact solution. On the basis of the TPINN, the FP-TPINN method is constructed by combining the time domain decomposition and the resampling strategy, further improving the performance of the TPINN. The proposed FP-TPINN has a good performance in solving evolution PDEs with strong nonlinearity, such as the AC equation, where the TPINN fails to produce accurate prediction.

7. Conclusions

In this paper, we propose a novel PINN framework for the forward progressive training based on the Taylor expansion term. The core contribution of this work lies in the first combination of the loss function constructed by Taylor expansion with the progressive training strategy. Although Taylor expansion has been applied in the PINN, their utilization as the central component of the loss function, integrated with a temporally progressive training mechanism, has not been reported. Specifically, we construct the TPINN framework using the Taylor series expansion, and develop the FP-TPINN to further improve the performance of the proposed TPINN. Our training strategy

divides the entire problem into multiple independent subproblems, and the k-th subproblem provides initial condition for the $(k+1)$-th subproblem. This information exchange ensures that the overall solution remains consistent on a global scale. By simulating the KdV equation, the potential Burgers equations, the reaction equation with large reaction coefficient, the improved Boussinesq equation, the flow mixing problem, and the five-dimensional diffusion-reaction equation, we demonstrate that the TPINN significantly improves the accuracy of the standard PINN. For the AC equation and the wave equation, the proposed FP-TPINN provides more accurate prediction results than that of the PINN. We also examine the capability of the TPINN for solving time-independent PDEs such as two-dimensional Poisson in Appendix 7. The analysis in the numerical experiments indicates that there is an inherent correlation between the computational cost and accuracy improvement of the TPINN method. Specifically, for the KdV equation and the diffusion-reaction equation, our proposed TPINN has efficient convergence efficiency and requires less training time than the PINN. In more complex scenarios such as the improved Boussinesq equation and the flow mixing problem, the TPINN only takes about 2 to 3 times the computation time, reducing the error from the order of 10^{-2} to the order of 10^{-3}. It is acceptable to exchange linear growth of time cost for rapid improvement in accuracy.

Although our proposed methods have demonstrated excellent predictive ability in solving PDEs with strong nonlinearity or stiff behavior, there are also some limitations and assumptions that will be the focus of our future research work. Our proposed methods increase computational cost when dealing with PDEs with high-order spatial derivatives. Therefore, for problems with fourth-order derivatives, it is necessary to consider designing specialized network structures or training strategies to ensure prediction accuracy and efficiency. In addition, the performance of our proposed methods mainly depend on the assumption that sufficient training data is provided to initialize the model or define the loss function. If the data is extremely sparse, effective predictions may not be obtained. Based on the above limitations, we plan to study more effective numerical schemes based on strong forms in future work to efficiently handle the problems with high-order derivative,such as the Cahn-Hilliard equation, and consider systematic comparison and analysis with energy based deep learning frameworks, such as the DEM method.

Author contributions

Wenkai Liu: Methodology, Software, Formal analysis, Validation, Writing–original draft; Yang Liu: Methodology, Validation, Funding acquisition, Writing–review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgements

This work is supported by the Young innovative talents project of Grassland Talents Project, Program for Innovative Research Team in Universities of Inner Mongolia Autonomous Region (NMGIRT2413).

Conflict of interest

The authors declare that they have no conflicts of interest.

References

1. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informend neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045
2. R. Laubscher, Simulation of multi-species flow and heat transfer using physics-informed neural networks, *Phys. Fluids*, **33** (2021), 087101. https://doi.org/10.1063/5.0058529
3. S. Z. Cai, Z. C. Wang, S. F. Wang, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks for heat transfer problems, *J. Heat Transfer.*, **143** (2021), 060801. https://doi.org/10.1115/1.4050542
4. S. Z. Cai, Z. P. Mao, Z. C. Wang, M. L. Yin, G. E. karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, *Acta Mech. Sin.*, **37** (2021), 1727–1738. https://doi.org/10.1007/s10409-021-01148-1
5. J. He, X. X. Li, H. Q. Zhu, An adaptive discrete physics-informed neural network method for solving the Cahn-Hilliard equation, *Eng. Anal. Bound. Elem.*, **155** (2023), 826–838. https://doi.org/10.1016/j.enganabound.2023.06.031
6. R. Mattey, S. Ghosh, A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations, *Comput. Methods Appl. Mech. Eng.*, **390** (2022), 114474. https://doi.org/10.1016/j.cma.2021.114474
7. J. Song, M. Zhong, G. E. Karniadakis, Z. Y. Yan, Two-stage initial-value iterative physics-informed neural networks for simulating solitary waves of nonlinear wave equations, *J. Comput. Phys.*, **505** (2024), 112917. https://doi.org/10.1016/j.jcp.2024.112917
8. G. M. Bai, U. Koley, S. Mishra, R. Molinaro, Physics informed neural networks (PINNs) for approximating nonlinear dispersive PDEs, *J. Comput. Math.*, **39** (2021), 816–847. https://doi.org/10.4208/jcm.2101-m2020-0342
9. Z. Y. Zhang, H. Zhang, Y. Liu, J. Y. Li, C. B. Li, Generalized conditional symmetry enhanced physics-informed neural network and application to the forward and inverse problems of nonlinear diffusion equations, *Chaos Solitons Fract.*, **168** (2023), 113169. https://doi.org/10.1016/j.chaos.2023.113169
10. J. G. Pan, X. F. Xiao, L. Guo, X. L. Feng, A high resolution physics-informed neural networks for high-dimensional convection-diffusion-reaction equations, *Appl. Soft Comput.*, **148** (2023), 110872. https://doi.org/10.1016/j.asoc.2023.110872
11. Z. Tang, S. D. Dong, X. S. Yang, J. J. Zhang, Application of a parallel physics-informed neural network to solve the multi-body dynamic equations for full-scale train collisions, *Appl. Soft Comput.*, **142** (2023), 110328. https://doi.org/10.1016/j.asoc.2023.110328

12. X. Wang, J. Li, J. C. Li, A deep learning based numerical PDE method for option pricing, *Comput. Econom.*, **62** (2023), 149–164. https://doi.org/10.1007/s10614-022-10279-x

13. W. K. Liu, Y. Liu, H. Li, Y. N. Yang, Multi-output physics-informed neural network for one- and two-dimensional nonlinear time distributed-order models, *Netw. Heterog. Media*, **18** (2023), 1899–1918. https://doi.org/10.3934/nhm.2023080

14. S. P. Wang, H. Zhang, X. Y. Jiang, Physics-informed neural network algorithm for solving forward and inverse problems of variable-order space-fractional advection-diffusion equations, *Neurocomputing*, **535** (2023), 64–82. https://doi.org/10.1016/j.neucom.2023.03.032

15. W. K. Liu, Y. Liu, H. Li, Time difference physics-informed neural network for fractional water wave models, *Results Appl. Math.*, **17** (2023), 100347. https://doi.org/10.1016/j.rinam.2022.100347

16. L. Yuan, Y. Q. Ni, X. Y. Deng, S. Hao, A-PINN: auxiliary physics informed neural networks for forward and inverse problems of nonlinear integro-differential equations, *J. Comput. Phys.*, **462** (2022), 111260. https://doi.org/10.1016/j.jcp.2022.111260

17. Y. Huang, W. R. Hao, G. Lin, HomPINNs: homotopy physics-informed neural networks for learning multiple solutions of nonlinear elliptic differential equations, *Comput. Math. Appl.*, **121** (2022), 62–73. https://doi.org/10.1016/j.camwa.2022.07.002

18. Y. Q. Liu, L. Cai, Y. P. Chen, P. F. Ma, Q. Zhong, Variable separated physics-informed neural networks based on adaptive weighted loss functions for blood flow model, *Comput. Math. Appl.*, **153** (2024), 108–122. https://doi.org/10.1016/j.camwa.2023.11.018

19. F. J. Cao, F. Gao, X. B. Guo, D. F. Yuan, Physics-informed neural networks with parameter asymptotic strategy for learning singularly perturbed convection-dominated problem, *Comput. Math. Appl.*, **150** (2023), 229–242. https://doi.org/10.1016/j.camwa.2023.09.030

20. C. X. Wu, M. Zhu, Q. Y. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.*, **403** (2023), 115671. https://doi.org/10.1016/j.cma.2022.115671

21. Z. Y. Zhang, S. J. Cai, H. Zhang, A symmetry group based supervised learning method for solving partial differential equations, *Comput. Methods Appl. Mech. Eng.*, **414** (2023), 116181. https://doi.org/10.1016/j.cma.2023.116181

22. Z. Y. Zhang, J. Y. Li, L. L. Guo, Invariant deep neural networks under the finite group for solving partial differential equations, *J. Comput. Phys.*, **523** (2025), 113680. https://doi.org/10.1016/j.jcp.2024.113680

23. E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, et al., An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications, *Comput. Methods Appl. Mech. Eng.*, **362** (2020), 112790. https://doi.org/10.1016/j.cma.2019.112790

24. Y. Z. Wang, J. Sun, J. S. Bai, C. Anitescum, M. S. Eshaghi, X. Y. Zhuang, et al., Kolmogorov-Arnold-informed neural network: a physics-informed deep learning framework for solving forward and inverse problems based on Kolmogorov-Arnold networks, *Comput. Methods Appl. Mech. Eng.*, **433** (2025), 117518. https://doi.org/10.1016/j.cma.2024.117518

25. V. M. Nguyen-Thanh, C. Anitescu, N. Alajlan, T. Rabczuk, X. Y. Zhuang, Parametric deep energy approach for elasticity accounting for strain gradient effects, *Comput. Methods Appl. Mech. Eng.*, **386** (2021), 114096. https://doi.org/10.1016/j.cma.2021.114096

26. H. K. Noh, M. Choi, J. H. Lim, Real-time full-field estimation of transient responses in time-dependent partial differential equations using causal physics-informed neural networks with sparse measurements, *Eng. Anal. Bound. Elem.*, **179** (2025), 106363. https://doi.org/10.1016/j.enganabound.2025.106363

27. J. Jung, H. Kim, H. Shin, M. Choi, CEENs: causality-enforced evolutional networks for solving time-dependent partial differential equations, *Comput. Methods Appl. Mech. Eng.*, **427** (2024), 117036. https://doi.org/10.1016/j.cma.2024.117036

28. A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *J. Comput. Phys.*, **404** (2020), 109136. https://doi.org/10.1016/j.jcp.2019.109136

29. Z. X. Xiang, W. Peng, X. Liu, W. Yao, Self-adaptive loss balanced physics-informed neural networks, *Neurocomputing*, **496** (2022), 11–34. https://doi.org/10.1016/j.neucom.2022.05.015

30. H. Nosrati, M. E. Niri, Manipulating the loss calculation to enhance the training process of physics-informed neural networks to solve the 1D wave equation, *Eng. Comput.*, **40** (2024), 1741–1769. https://doi.org/10.1007/s00366-023-01881-0

31. X. Rao, Y. N. Liu, X. P. He, H. Hoteit, Physics-informed Kolmogorov-Arnold networks to model flow in heterogeneous porous media with a mixed pressure-velocity formulation, *Phys. Fluids*, **37** (2025), 076654. https://doi.org/10.1063/5.0279122

32. H. Shuai, F. X. Li, Physics-informed Kolmogorov-Arnold networks for power system dynamics, *IEEE Open Access J. Power Energy*, **12** (2025), 46–58. https://doi.org/10.1109/OAJPE.2025.3529928

33. F. Mostajeran, S. A. Faroughi, Scaled-cPIKANs: spatial variable and residual scaling in Chebyshev-based physics-informed Kolmogorov-Arnold networks, *J. Comput. Phys.*, **537** (2025), 114116. https://doi.org/10.1016/j.jcp.2025.114116

34. E. Kharazmi, Z. Q. Zhang, G. E. M. Karniadakis, *hp*-VPINNs: variational physics-informed neural networks with domain decomposition, *Comput. Methods Appl. Mech. Eng.*, **374** (2021), 113547. https://doi.org/10.1016/j.cma.2020.113547

35. B. Moseley, A. Markham, T. Nissen-Meyer, Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations, *Adv. Comput. Math.*, **49** (2023), 62. https://doi.org/10.1007/s10444-023-10065-9

36. Z. Y. Hu, A. D. Jagtap, G. E. Karniadakis, K. Kawaguchi, Augmented physics-informed neural networks (APINNs): a gating network-based soft domain decomposition methodology, *Eng. Appl. Artif. Intell.*, **126** (2023), 107183. https://doi.org/10.1016/j.engappai.2023.107183

37. C. Liu, H. A. Wu, *cv*-PINN: efficient learning of variational physics-informed neural network with domain decomposition, *Extreme Mech. Lett.*, **63** (2023), 102051. https://doi.org/10.1016/j.eml.2023.102051

38. T. Bandai, T. A. Ghezzehei, Forward and inverse modeling of water flow in unsaturated soils with discontinuous hydraulic conductivities using physics-informed neural networks with domain decomposition, *Hydrol. Earth Syst. Sci.*, **26** (2022), 4469–4495. https://doi.org/10.5194/hess-26-4469-2022

39. W. D. Zhai, D. W. Tao, Y. Q. Bao, Parameter estimation and modeling of nonlinear dynamical systems based on Runge-Kutta physics-informed neural network, *Nonlinear Dyn.*, **111** (2023), 21117–21130. https://doi.org/10.1007/s11071-023-08933-6

40. Y. B. Mao, Y. L. Gu, L. Sha, H. J. Shao, Q. X. Wang, T. Abdelzaher, Phy-Taylor: partially physics-knowledge-enhanced deep neural networks via NN editing, *IEEE Trans. Neural Netw. Learn. Syst.*, **36** (2025), 447–461. https://doi.org/10.1109/TNNLS.2023.3325432

41. J. Mau, K. Moon, Incorporating Taylor series and recursive structure in neural networks for time series prediction, 2024, arXiv: 2402.06441.

42. A. Krishnapriyan, A. Gholami, S. D. Zhe, R. Kriby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, In: *Advances in neural information processing systems (NeurIPS 2021)*, **34** (2021), 26548–26560.

43. J. W. Guo, Y. Z. Yao, H. Wang, T. X. Gu, Pre-training strategy for solving evolution equations based on physics-informed neural networks, *J. Comput. Phys.*, **489** (2023), 112258. https://doi.org/10.1016/j.jcp.2023.112258

44. L. Lu, X. H. Meng, Z. P. Mao, G. E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, *SIAM Rev.*, **63** (2021), 208–228. https://doi.org/10.1137/19M1274067

45. Z. Y. Zhang, H. Zhang, L. S. Zhang, L. L. Guo, Enforcing continuous symmetries in physics-informed neural network for solving forward and inverse problems of partial differential equations, *J. Comput. Phys.*, **492** (2023), 112415. https://doi.org/10.1016/j.jcp.2023.112415

46. W. L. Guan, K. H. Yang, Y. S. Chen, S. L. Liao, Z. Guan, A dimension-augmented physics-informed neural network (DaPINN) with high level accuracy and efficiency, *J. Comput. Phys.*, **491** (2023), 112360. https://doi.org/10.1016/j.jcp.2023.112360

47. P. H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, Y. S. Ong, CAN-PINN: a fast physics-informed neural network based on coupled-automatic-numerical differentiation method, *Comput. Methods Appl. Mech. Eng.*, **395** (2022), 114909. https://doi.org/10.1016/j.cma.2022.114909

48. P. Tamamidis, D. N. Assanis, Evaluation of various high-order-accuracy schemes with and without flux limiters, *Int. J. Numer. Methods Fluids*, **16** (1993), 931–948. https://doi.org/10.1002/fld.1650161006

49. Y. H. Zang, G. Bao, X. J. Ye, H. M. Zhou, Weak adversarial networks for high-dimensional partial differential equations, *J. Comput. Phys.*, **411** (2020), 109409. https://doi.org/10.1016/j.jcp.2020.109409

Appendix. Two-dimensional Poisson equation

In this section, we aim to investigate the performance of our proposed TPINN method in solving the time-independent problem. Consider the following Poisson equation with Dirichlet

boundary condition:

$$\nabla^2 u(x, y) = f(x, y), (x, y) \in (0, 1) \times (0, 1), \tag{7.1}$$

where the exact solution $u(x, y) = (0.1 \sin(2\pi x) + \tanh(x)) \sin(2\pi y)$ and the source term can be derived from the exact solution. For this example, we define the Taylor expansion term as

$$\mathcal{F} = u_x(x, y) - \Big[u_x(x - \Delta x, y) + \Delta x(f(x - \Delta x, y) - u_{yy}(x - \Delta x, y)\Big], \tag{7.2}$$

where $\Delta x = \frac{x}{n}$. Then, the loss function can be formulated as

$$\mathcal{L}(\Theta; \mathcal{J}) = w_b \mathcal{L}_b(\Theta; \mathcal{J}_b) + w_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}(\Theta; \mathcal{J}_f). \tag{7.3}$$

The neural network consists of 4 hidden layers and 20 neurons on each hidden layer. The training set contains 2000 collocation points sampled from the solution domain and 320 boundary points. The weights w_b and $w_{\mathcal{F}}$ are set to 1 and 100, respectively. The Adam algorithm is selected as the optimizer, with a learning rate of 0.001, and the number of iterations is set to 10000.

In Table 21, we present the relative errors and computational cost of the TPINN, the PINN, and the *hp*-variational PINN (VPINN) [34]. The results show that our proposed method outperforms the PINN in accuracy with almost the same computational time. Although the proposed TPINN has slightly lower accuracy than the VPINN, it exhibits higher computational efficiency. Furthermore, it can be observed that our proposed method still maintains good prediction performance under different numbers of training points, proving its stability. To more intuitively illustrate the performance of the TPINN, we plot the predicted solutions and corresponding error distributions of the TPINN, the PINN, and the VPINN in Figure 25.

Table 21. Poisson equation: the computational cost and the relative L^2 errors of the TPINN ($n = 80$), the PINN, and the VPINN with different number of training points N_f.

	N_f	TPINN	PINN	VPINN
Relative L^2 error	1000	5.1578e-02	1.0860e-01	1.3482e-02
	2000	3.4024e-02	6.9169e-02	1.3691e-02
	3000	3.9881e-02	1.6236e-01	1.3630e-02
Time(s)	1000	23	24	171
	2000	48	45	178
	3000	54	55	174

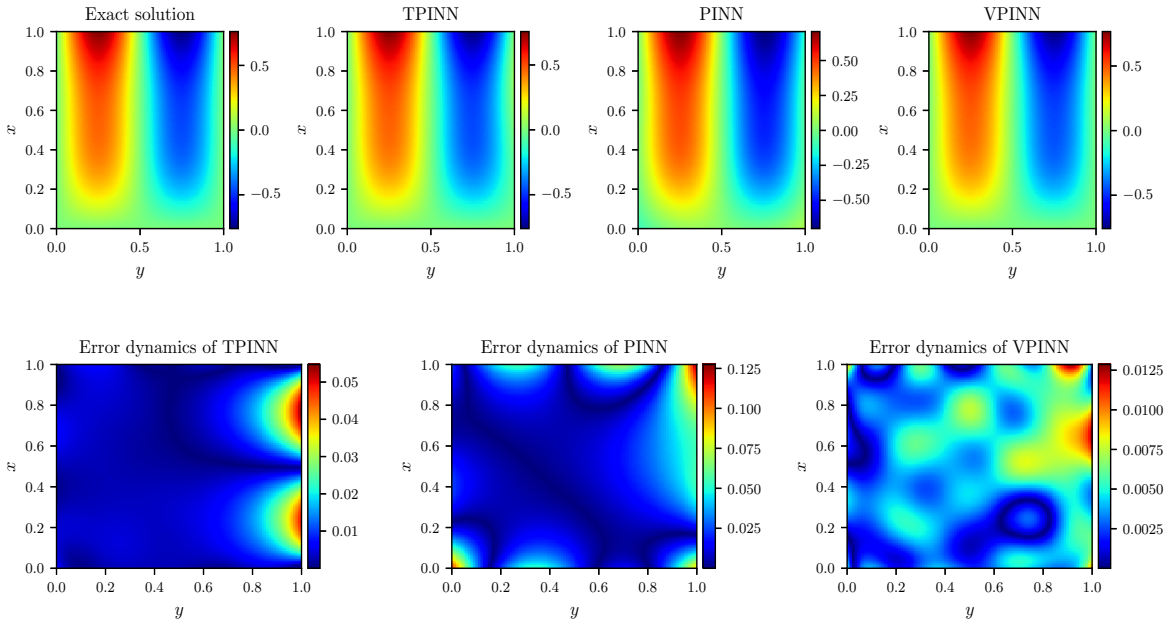


Figure 25. Poisson equation: (top) comparison between the exact solution and the solutions solved by the TPINN, the PINN, and the VPINN; (bottom) absolute errors of the TPINN, the PINN, and the VPINN.