AIMS Mathematics, 9(10): 27418–27437.
DOI: 10.3934/math.20241332
Received: 15 July 2024
Revised: 05 September 2024
Accepted: 18 September 2024
Published: 23 September 2024

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

Research article

Gradient-enhanced fractional physics-informed neural networks for solving forward and inverse problems of the multiterm time-fractional Burger-type equation

Shanhao Yuan[1], Yanqin Liu[2,*], Yibin Xu[1], Qiuping Li[2], Chao Guo[2] and Yanfeng Shen[3]

[1] School of Mathematics and Statistics, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China

[2] School of Mathematics and Big Data, Dezhou University, Dezhou 253023, China

[3] Data Recovery Key Laboratory of Sichuan Province, College of Mathematics and Information Science, Neijiang Normal University, Neijiang 641100, China

* **Correspondence:** Email: yqliumath@163.com; Tel: +8613853467235.

Abstract: In this paper, we introduced the gradient-enhanced fractional physics-informed neural networks (gfPINNs) for solving the forward and inverse problems of the multiterm time-fractional Burger-type equation. The gfPINNs leverage gradient information derived from the residual of the fractional partial differential equation and embed the gradient into the loss function. Since the standard chain rule in integer calculus is invalid in fractional calculus, the automatic differentiation of neural networks does not apply to fractional operators. The automatic differentiation for the integer order operators and the finite difference discretization for the fractional operators were used to construct the residual in the loss function. The numerical results demonstrate the effectiveness of gfPINNs in solving the multiterm time-fractional Burger-type equation. By comparing the experimental results of fractional physics-informed neural networks (fPINNs) and gfPINNs, it can be seen that the training performance of gfPINNs is better than fPINNs.

Keywords: multiterm time-fractional Burger-type equation; fPINNs; gfPINNs; forward problem; inverse problem
Mathematics Subject Classification: 26A33, 35R11, 65M22, 68T07

1. Introduction

In recent years, fractional partial differential equations (FPDEs) have been widely used in natural science and engineering technology [1–4]. The advantage of FPDEs lies in their ability to better

describe materials and processes that exhibit memory and genetic properties [5, 6]. However, the solutions of FPDEs are much more complex. Many researchers have exploited diverse techniques for the investigation of FPDEs such as the finite difference method (FDM) [7], finite element method [8], spectral method [9], virtual element method [10], etc. The development of effective numerical methods to approximate FPDEs has been the goal of some researchers.

In recent years, neural networks (NNs) have been successfully applied to solve problems in various fields [11–13]. Due to the high expressiveness of NNs in functional approximation [14–16], using NNs to solve differential and integral equations has become an active and important research field. Physics-informed neural networks (PINNs) [17–20] are machine learning models that combine deep learning with physical knowledge. PINNs embed PDEs into the loss function of the NNs, enabling the NNs to learn solutions to PDEs. The PINNs algorithm is meshless and simple, and can be applied to various types of PDEs, including integral differential equations, FPDEs, and random partial differential equations. Moreover, PINNs solved the inverse problem of PDEs just as easily as they solved the forward problem [17]. PINNs have been successfully applied to solve various problems in scientific computing [21–23]. Pang et al. [24] used the FDM to approximate the fractional derivatives that cannot be automatically differentiated, thus extending the PINNs to fPINNs for solving FPDEs.

Despite the success of deep learning in the past, solving a wide range of PDEs is theoretically and practically challenging as complexity increases. Therefore, many aspects of PINNs need to be further improved to achieve more accurate predictions, higher computational efficiency, and robustness of training. Lu et al. [25] proposed DeepXDE, a deep learning library for solving PDEs, introduced a new residual-based adaptive refinement method to improve the training efficiency of PINNs, and new residual points were added at the position where the residuals of the PDEs were large, so that the discontinuities of PDEs could be captured well. Zhang et al. [26] combined fPINNs with the spectral method to solve the time-fractional phase field models. It had the characteristics of reducing the approximate number of discrete fractional operators, thus improving the training efficiency and obtaining higher error accuracy. Wu et al. [27] conducted a comprehensive study on two types of sampling of PINNs, including non-adaptive uniform sampling and adaptive non-uniform sampling, and the research results could also be used as a practical guide for selecting sampling methods. Zhang et al. [28] removed the soft constraints of PDEs in the loss function, and used the Lie symmetry group to generate the labeled data of PDEs to build a supervised learning model, thus effectively predicting the large amplitude and high frequency solutions of the Klein-Gordon equation. Zhang et al. [29] introduced the symmetry-enhanced physics-informed neural network (SPINN), which incorporated the invariant surface conditions derived from Lie symmetries or non-classical symmetries of PDEs into the loss function of PINNs, aiming to improve accuracy of PINNs. Lu et al. [30] and Xie et al. [31] introduced gradient-enhanced physics-informed neural networks (gPINNs) to solve PDEs and the idea of embedding the gradient information from the residuals of PDEs into the loss functions has also proven to be effective in other methods such as Gaussian process regression [32].

In this paper, inspired by the above works, gfPINNs are applied to solve the forward and inverse problems of the multiterm time-fractional Burger-type equation. The integer order derivatives are handled using the automatic differentiation capability of the NNs, while the fractional derivatives of the equation are approximated using finite difference discretization [33, 34]. Subsequently, the residual information of the equation is then incorporated into the loss function of NNs and optimized to yield optimal parameters. For the inverse problems of the multiterm time-fractional Burger-type

equation, their overall form are known but the coefficient and the orders of time-fractional derivatives are unknown. The gfPINNs explicitly incorporate information from the equation by including the differential operators of the equation directly into the optimization loss function. The parameters to be identified appear in the differential operators, which are then optimized by minimizing the loss function associated with those parameters. A numerical comparison between fPINNs and gfPINNs is conducted using numerical examples. The numerical results demonstrate the effectiveness of gfPINNs in solving the multiterm time-fractional Burger-type equation.

The structure of this paper is as follows. In Section 2, we define forward and inverse problems for the multiterm time-fractional Burger-type equation. In Section 3, we introduce fPINNs and gfPINNs and give the finite difference discretization to approximate the time-fractional derivatives. In Section 4, we demonstrate the effectiveness of gfPINNs in solving the forward and inverse problems of the multiterm time-fractional Burger-type equation by numerical examples, and compare the experimental results of fPINNs and gfPINNs. Finally, we give the conclusions of this paper in Section 5.

2. Multiterm time-fractional Burger-type equation

We consider the following multiterm time-fractional Burger-type equation defined on the bounded domain Ω:

$$c_1 {}_0^C D_t^\alpha u(x,t) + c_2 {}_0^C D_t^\gamma u(x,t) + u(x,t)\frac{\partial u(x,t)}{\partial x} = \upsilon \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \tag{2.1}$$

where $(x,t) \in \Omega \times [0,T]$ and the initial and boundary conditions are given as

$$\begin{cases} u(x,t) = 0, & x \in \partial\Omega, \\ u(x,0) = g(x), & x \in \Omega, \end{cases} \tag{2.2}$$

where $u(x,t)$ is the solution of the equation, $f(x,t)$ is the forcing term whose values are only known at scattered spatio-temporal coordinates, υ is the kinematic viscosity of fluid, $g(x)$ is a sufficiently smooth function, the fractional orders α and γ have been restricted to (0, 1) and (1, 2), respectively, ${}_0^C D_t^\theta u(x,t)$ is the Caputo time-fractional derivative of order θ ($\theta > 0, n-1 \le \theta < n$) of $u(x,t)$ with respect to t [35, 36]:

$${}_0^C D_t^\theta u(x,t) = \begin{cases} \frac{1}{\Gamma(n-\theta)} \int_\alpha^t (t-s)^{n-1-\theta} \frac{\partial^n u(x,s)}{\partial s^n} ds, & \theta \notin \mathbf{z}^+, \\ \frac{\partial^\theta u(x,t)}{\partial t^\theta}, & \theta \in \mathbf{z}^+, \end{cases} \tag{2.3}$$

where $\Gamma(\cdot)$ is the gamma function.

The forward and inverse problems of solving the multiterm time-fractional Burger-type equation are described as follows. For the forward problem, under the given preconditions of the fractional orders α and γ, the forcing term f, and the initial and boundary conditions, the solution $u(x,t)$ is solved. For the inverse problem, under the given preconditions of the initial and boundary conditions, the forcing term f, and additional concentration measurements at the final time $u(x,t) = h(x,t)$, the fractional orders α and γ, the flow velocity υ, and the solution $u(x,t)$ are solved.

3. Methodology

3.1. fPINNs

This subsection introduces the idea of fPINNs and we consider both the forward and inverse problems, along with their corresponding NNs. We first consider the forward problem of the multiterm time-fractional Burger-type equation in the following form:

$$\begin{cases} \mathscr{L}\{u(x,t)\} = f(x,t), & (x,t) \in \Omega \times [0,T], \\ u(x,t) = 0, & x \in \partial\Omega, \\ u(x,0) = g(x), & x \in \Omega, \end{cases} \tag{3.1}$$

where $\mathscr{L}\{\cdot\}$ is a nonlinear operator and $\mathscr{L}\{u(x,t)\} = c_1 {}_0^C D_t^\alpha u(x,t) + c_2 {}_0^C D_t^\gamma u(x,t) + u(x,t)\frac{\partial u(x,t)}{\partial x} - \upsilon\frac{\partial^2 u(x,t)}{\partial x^2}$. We divide the nonlinear operator $\mathscr{L}\{\cdot\}$ into two parts, $\mathscr{L} = \mathscr{L}_{AD} + \mathscr{L}_{nonAD}$. The first part is an integer derivative operator, which can be automatically differentiated (AD) using the chain rule. We have

$$\mathscr{L}_{AD}\{\cdot\} = \begin{cases} u(x,t)\frac{\partial u(x,t)}{\partial x} - \upsilon\frac{\partial^2 u(x,t)}{\partial x^2}, & \alpha \in (0,1), \gamma \in (1,2), \\ c_2\frac{\partial^2 u(x,t)}{\partial t^2} + u(x,t)\frac{\partial u(x,t)}{\partial x} - \upsilon\frac{\partial^2 u(x,t)}{\partial x^2}, & \alpha \in (0,1), \gamma = 2, \\ c_1\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} - \upsilon\frac{\partial^2 u(x,t)}{\partial x^2}, & \alpha = 1, \gamma \in (1,2), \end{cases} \tag{3.2}$$

and the second category consists of operators that lack automatic differentiation capabilities:

$$\mathscr{L}_{nonAD}\{\cdot\} = \begin{cases} c_1 {}_0^C D_t^\alpha u(x,t) + c_2 {}_0^C D_t^\gamma u(x,t), & \alpha \in (0,1), \gamma \in (1,2), \\ c_1 {}_0^C D_t^\alpha u(x,t), & \alpha \in (0,1), \gamma = 2, \\ c_2 {}_0^C D_t^\gamma u(x,t), & \alpha = 1, \gamma \in (1,2). \end{cases} \tag{3.3}$$

For $\mathscr{L}_{nonAD}$, we can discretize it using FDM and denote by $\mathscr{L}_{FDM}$ the discretization version of $\mathscr{L}_{nonAD}$.

During the NNs training process, our goal is to optimize its parameters in order to ensure that the approximate solution of the equation closely satisfies the initial and boundary conditions. The approximate solution is chosen as

$$\tilde{u}(x,t) = t\rho(x)u_{NN}(x,t) + g(x), \tag{3.4}$$

where u_{NN} represents the output of the NNs. The NNs acts as a surrogate model, approximating the relationship between spatio-temporal coordinates and the solution of the equation. It is defined by its weights and biases, forming the parameter vector μ; see Figure 1 for a simple NN. This is fully connected with a single hidden layer consisting of three neurons. In this network, x and t are two inputs, which go through a linear transformation to obtain $x_1 = w_1 x + w_4 t + b_1$, $x_2 = w_2 x + w_5 t + b_2$, and $x_3 = w_3 x + w_6 t + b_3$ in the hidden layer, and then, they go through a nonlinear transformation to get $Y_i = f(x_i)$ for $i = 1, 2, 3$. We choose the hyperbolic tangent function $\tanh(\cdot)$. Y_i to go through a linear transformation to obtain the output of the NNs, $u_{NN}(x,t;\mu) = w_7 Y_1 + w_8 Y_2 + w_9 Y_3 + b_4$. The vector of parameters μ is comprised of the weights w_i and biases b_i. $\rho(0) = \rho(1) = 0$ and the auxiliary function $\rho(x)$ is preselected. $g(x)$ is the initial condition function such that it satisfies the initial and boundary conditions automatically.

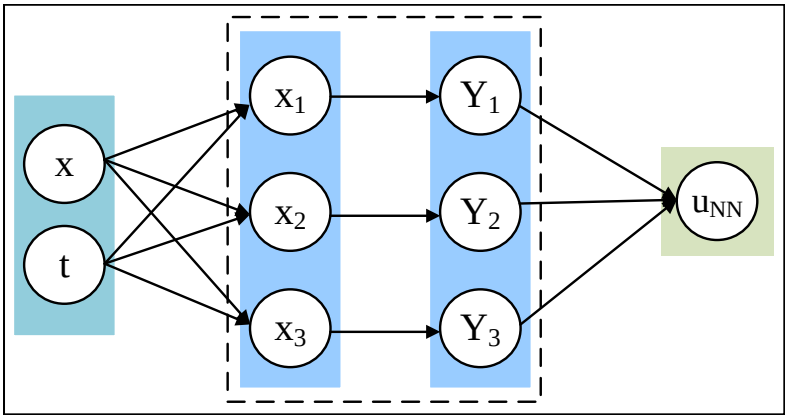


Figure 1. A simple NN.

The loss function of fPINNs for the forward problem with the approximate solution is defined as the mean-squared error of the equation residual

$$L_{FW} = \frac{1}{|S_F|} \sum_{(x,t)\in S} \left[\mathscr{L}_{FDM}\{\tilde{u}(x,t)\} + \mathscr{L}_{AD}\{\tilde{u}(x,t)\} - f(x,t)\right]^2, \tag{3.5}$$

where $S_F \subset \Omega \times [0,T]$ and $|S_F|$ represents the number of training points. Then, we train the NNs to optimize the loss function of the forward problem with respect to the NNs parameters μ, thus obtaining the optimal parameters μ_{best}. Finally, we specify a set of arbitrary test points to test the trained NNs and observe the training performance.

The codes for solving the forward and inverse problems of the equation using NNs is similar. We only need to incorporate the parameters to be identified in the inverse problem into the loss function to be optimized in the forward problem, and no other changes are necessary. Next, we consider the following form of the inverse problem:

$$\begin{cases} \mathscr{L}^{\xi=\{\alpha,\gamma,v\}}\{u(x,t)\} = f(x,t), & (x,t) \in \Omega \times [0,T], \\ u(x,t) = 0, & x \in \partial\Omega, \\ u(x,0) = g(x), & x \in \Omega, \\ u(x,t) = h(x,t), & (x,t) \in \Omega \times [0,T], \end{cases} \tag{3.6}$$

where ξ is the parameter of the equation, so the loss function L_{IV} for the inverse problem under consideration is

$$\begin{aligned} L_{IV}\{\mu, \xi = \{\alpha,\gamma,v\}\} =& W_{I_1} \frac{1}{\left|S_{I_1}\right|} \sum_{(x,t)\in S_{I_1}} [\mathscr{L}_{FDM}^{\{\alpha,\gamma\}}\{\tilde{u}(x,t)\} + \mathscr{L}_{AD}^{v}\{\tilde{u}(x,t)\} - f(x,t)]^2 \\ &+ W_{I_2} \frac{1}{\left|S_{I_2}\right|} \sum_{(x,t)\in S_{I_2}} [\tilde{u}(x,t) - h(x,t)]^2, \end{aligned} \tag{3.7}$$

where $\alpha \in (0,1)$ and $\gamma \in (1,2)$, $S_{I_1} \subset \Omega \times [0,T]$ and $S_{I_2} \subset \Omega \times [0,T]$ are two sets of different training points, and W_{I_1} and W_{I_2} are preselected weight coefficients. We train the NNs to minimize the loss function, thereby obtaining α_{best} and γ_{best}, the flow velocity v_{best}, and the optimal parameters μ_{best} of the NNs.

3.2. gfPINNs

We incorporate the residual information of the equation into the loss function of NNs and train the NNs to minimize this loss function, thus obtaining the optimal parameters of NNs. If the residuals in the PDEs are zero, then the gradient of the residuals in the PDEs should also be zero. Therefore, adding gradient information to the loss function is a necessary condition for training NNs. One motivation behind gfPINNs is that the residual in the loss function often fluctuates near zero. Penalizing the slope of the residual can reduce these fluctuations, making the residual closer to zero. In this section, we continue to consider the formulation of the forward and inverse problems of the equation discussed in the previous section.

We first consider the forward problem in the form of (3.1) and provide the loss function of gfPINNs for this form:

$$L_{gFW} = W_F L_{FW} + W_{g_1F} L_{g_1FW} + W_{g_2F} L_{g_2FW}, \tag{3.8}$$

where

$$L_{g_1FW} = \frac{1}{\left|S_{g_1F}\right|} \sum_{(x,t)\in S_{g_1F}} \left[\frac{\partial \mathscr{L}_{FDM}\{\tilde{u}(x,t)\}}{\partial x} + \frac{\partial \mathscr{L}_{AD}\{\tilde{u}(x,t)\}}{\partial x} - \frac{\partial f(x,t)}{\partial x}\right]^2, \tag{3.9}$$

$$L_{g_2FW} = \frac{1}{\left|S_{g_2F}\right|} \sum_{(x,t)\in S_{g_2F}} \left[\frac{\partial \mathscr{L}_{FDM}\{\tilde{u}(x,t)\}}{\partial t} + \frac{\partial \mathscr{L}_{AD}\{\tilde{u}(x,t)\}}{\partial t} - \frac{\partial f(x,t)}{\partial t}\right]^2, \tag{3.10}$$

and the approximate solution of the equation is the same as Eq (3.4): $\tilde{u}(x,t) = \rho(x)u_{NN}(x,t) + g(x)$. The expression L_{FW} as shown in Eq (3.5), where W_F, W_{g_1F}, and W_{g_2F} are preselected weighting coefficients, $S_{g_1F} \subset \Omega \times [0,T]$ and $S_{g_2F} \subset \Omega \times [0,T]$ are two sets of different training points.

Next, we consider the inverse problem in the form of (3.6) and provide the loss function of gfPINNs for this form. The approach for the inverse problem of gfPINNs is similar to that of fPINNs. We provide the loss function for the inverse problem of gfPINNs.

$$L_{gIV} = W_I L_{IV}\{\mu, \xi = \{\alpha, \gamma, v\}\} + W_{g_1I} L_{g_1IV} + W_{g_2I} L_{g_2IV}, \tag{3.11}$$

where

$$\begin{aligned} L_{g_1IV} =& W_{g_1I_1} \frac{1}{\left|S_{g_1I_1}\right|} \sum_{(x,t)\in S_{g_1I_1}} \left[\frac{\partial \mathscr{L}^{\{\alpha,\gamma\}}_{FDM}\{\tilde{u}(x,t)\}}{\partial x} + \frac{\partial \mathscr{L}^{v}_{AD}\{\tilde{u}(x,t)\}}{\partial x} - \frac{\partial f(x,t)}{\partial x}\right]^2 \\ &+ W_{g_1I_2} \frac{1}{\left|S_{g_1I_2}\right|} \sum_{(x,t)\in S_{g_1I_2}} \left[\frac{\partial \tilde{u}(x,t)}{\partial x} - \frac{\partial h(x,t)}{\partial x}\right]^2, \end{aligned} \tag{3.12}$$

$$\begin{aligned} L_{g_2IV} =& W_{g_2I_1} \frac{1}{\left|S_{g_2I_1}\right|} \sum_{(x,t)\in S_{g_2I_1}} \left[\frac{\partial \mathscr{L}^{\{\alpha,\gamma\}}_{FDM}\{\tilde{u}(x,t)\}}{\partial t} + \frac{\partial \mathscr{L}^{v}_{AD}\{\tilde{u}(x,t)\}}{\partial t} - \frac{\partial f(x,t)}{\partial t}\right]^2 \\ &+ W_{g_2I_2} \frac{1}{\left|S_{g_2I_2}\right|} \sum_{(x,t)\in S_{g_2I_2}} \left[\frac{\partial \tilde{u}(x,t)}{\partial t} - \frac{\partial h(x,t)}{\partial t}\right]^2, \end{aligned} \tag{3.13}$$

and the expression $L_{IV}\{\mu, \xi = \{\alpha, \gamma, \upsilon\}\}$ as shown in Eq (3.7), where W_I, $W_{g_1 I}$, $W_{g_2 I}$, $W_{g_1 I_1}$, $W_{g_1 I_2}$, $W_{g_2 I_1}$, and $W_{g_2 I_2}$ are preselected weighting coefficients, $S_{g_1 I_1}, S_{g_2 I_1} \subset \Omega \times [0, T]$, $S_{g_1 I_2}, and S_{g_2 I_2} \subset \Omega \times [0, T]$ are four sets of different training points.

This defines the loss function of gfPINNs, which is exactly the same as discussed above for fPINNs. We train the NNs to obtain the optimal parameters of the NNs.

3.3. Finite difference discretization for time-fractional derivatives

In the x direction $[0, M]$, we take the mesh points $x_p = ih_x, i = 0, 1, 2, ..., M_1$, and in the t direction $[0, T]$, we take the mesh points $t_n = n\tau, n = 0, 1, ..., N$, where $h_x = \frac{M}{M_1}$ and $\tau = \frac{T}{N}$ are the uniform spatial step size and temporal step size, respectively. Denote $\Omega_h \equiv \left\{0 \le i \le M_1\right\}, \Omega_\tau \equiv \{0 \le n \le N\}$. Suppose $u_i^n = u(x_i, t_n)$ is a grid function on $\Omega_h \times \Omega_\tau$.

We approximate the fractional derivatives of the equation using the finite difference discretization [33, 34].

For $\alpha \in (0, 1)$, we have ${}_0^C D_t^\alpha u(x,t) \mid_{(x_i,t_n)} = D_\tau^\alpha \tilde{u}_i^n + R_1(\tilde{u}_i^n)$,

$$D_\tau^\alpha \tilde{u}_i^n := \frac{\tau^{-\alpha}}{\Gamma(2-\alpha)}[a_0^\alpha \tilde{u}_i^n + \sum_{k=1}^{n-1}(a_{n-k}^\alpha - a_{n-k-1}^\alpha)\tilde{u}_i^k - a_{n-1}^\alpha \tilde{u}_i^0], \tag{3.14}$$

where $\tilde{u}_i^n = \tilde{u}(x_i, t_n)$, $R_1 \le C(\tau^{2-\alpha})$, and $a_k^\alpha = (k+1)^{1-\alpha} - k^{1-\alpha}$.

Lemma 3.1. *[33]* $\alpha \in (0, 1)$, $a_l^\alpha = (l+1)^{1-\alpha} - l^{1-\alpha}$, $l = 0, 1, 2, \ldots,$

(1) $1 = a_0^\alpha > a_1^\alpha > a_2^\alpha > \cdots > a_l^\alpha > 0$, $\lim_{l\to\infty} a_l^\alpha \to 0$,

(2) $(1-\alpha)^{l^{-\alpha}} < a_{l-1}^{(\alpha)} < (1-\alpha)(l-1)^{-\alpha}$, $l \ge 1$.

For $\gamma \in (1, 2)$, ${}_0^C D_t^\gamma u(x,t) \mid_{(x_i,t_n)} = D_\tau^\gamma \tilde{u}_i^n + R_2(\tilde{u}_i^n)$,

$$D_\tau^\gamma \tilde{u}_i^n := \frac{\tau^{1-\gamma}}{\Gamma(3-\gamma)}[b_0^\gamma \delta_t \tilde{u}_i^n + \sum_{k=1}^{n-1}(b_{n-k}^\gamma - b_{n-k-1}^\gamma)\delta_t \tilde{u}_i^k - b_{n-1}^\gamma \delta_t \tilde{u}_i^0], \tag{3.15}$$

where $\delta_t u(x,t) = \frac{\partial u(x,t)}{\partial t}$, $R_2 \le C(\tau^{3-\gamma})$, and $b_k^\gamma = (k+1)^{2-\gamma} - k^{2-\gamma}$.

Given the spatial position x, it can be seen from the finite difference discretization that the time-fractional derivative of $\tilde{u}(x,t)$ evaluated at time t depends on the value of $\tilde{u}(x,t)$ calculated at all previous times $0, \tau, 2\tau, \cdots, t$. We call the current time and the previous time the training points and the auxiliary points, respectively.

4. Numerical examples

In this section, we demonstrate the effectiveness of gfPINNs in solving forward and inverse problems of the multiterm time-fractional Burger-type equation and we compared fPINNs with gfPINNs. We solve the forward problems of the equation and present the experimental results in Section 4.1. We solve the inverse problems and present the experimental results in Section 4.2.

We give a fabricated solution to the problem $u(x,t) = t^p \sin(\pi x)$. In the given approximate solution (3.4), the auxiliary function $\rho(\cdot)$ is defined as $\rho(\cdot) = 1 - \|x\|_2^2$. We use the following form of L^2 relative error:

$$\frac{\left\{\sum_k [u(x_{test,k}, t_{test,k}) - \tilde{u}(x_{test,k}, t_{test,k})]^2\right\}^{\frac{1}{2}}}{\left\{\sum_k [u(x_{test,k}, t_{test,k})]^2\right\}^{\frac{1}{2}}} \tag{4.1}$$

to measure the performance of the NNs, where $\tilde{u}$ denotes the approximated solution, u is the exact solution, and (x_k, t_k) denotes the k-th test point.

We wrote the code in Python and took advantage of the automatic differentiation capability of TensorFlow [37]. The stochastic gradient descent Adam algorithm [38] was used to optimize the loss function. We initialized the NNs parameters using normalized Glorot initialization [39]. Otherwise, when training a neural network, we set the learning rate, the number of neurons, the number of hidden layers, and the activation function as 1×10^{-3}, 20, 4, and $\tanh(x)$, respectively.

4.1. Forward problems

In this section, we consider the the multiterm time-fractional Burger-type equation of the form (2.1) with initial and boundary conditions (2.2). We let $\upsilon = 1$, $(x,t) \in [0,1]\times[0,1]$, and $g(x) = 0$, considering the smooth fabricated solution $u(x,t) = t^p \sin(\pi x)$ and the forcing term

$$\begin{aligned} f(x,t) =& \frac{\Gamma(p+1)}{\Gamma(p+1-\alpha)} t^{(p-\alpha)}(p-\alpha)\sin(\pi x) + \frac{\Gamma(p+1)}{\Gamma(p+1-\gamma)} t^{(p-\gamma)}(p-\gamma)\sin(\pi x) \\ &+ t^{2p}\sin(\pi x)\cos(\pi x) + \pi^2 t^p \sin(\pi x). \end{aligned} \tag{4.2}$$

Case 1: We choose $c_1 = 1$, $c_2 = 0$, and $\alpha = 0.5$, considering the smooth fabricated solution $u(x,t) = t^4 \sin(\pi x)$ and the forcing term $f(x,t) = 3.5\frac{\Gamma(5)}{\Gamma(4.5)} t^{3.5}\sin(\pi x) + t^8 \sin(\pi x)\cos(\pi x) + \pi^2 t^4 \sin(\pi x)$. We consider $M_1 - 1$ training points of the spatial domain: $x_i = i h_x$ for $i = 1, 2, \cdots, M_1 - 1$ and N training points of the time domain: $t_n = n\tau$ for $n = 1, 2, \cdots, N$. We do not need to place training points on the initial and boundary since the approximate solution $\tilde{u}(x,t) = tx(1-x)u_{NN}(x,t;\mu)$ satisfies the initial and boundary conditions automatically. For fPINNs, the loss function can be written as

$$\begin{aligned} L_{FW} = \frac{1}{(M_1-1)N} \sum_{i=1}^{M-1}\sum_{n=1}^{N} \Bigg\{ & \frac{\tau^{-0.5}}{\Gamma(1.5)} \left[a_0^{0.5}\tilde{u}(x_i,t_n) + \sum_{k=1}^{n-1}(a_{n-k}^{0.5} - a_{n-k-1}^{0.5})\tilde{u}(x_i,t_k) \right] \\ & + \tilde{u}(x_i,t_n)\frac{\partial \tilde{u}(x_i,t_n)}{\partial x_i} - \frac{\partial^2 \tilde{u}(x_i,t_n)}{\partial x_i^2} - f(x_i,t_n) \Bigg\}^2. \end{aligned} \tag{4.3}$$

The loss function of gfPINNs can be given as

$$\begin{aligned} L_{g_2 FW} = \frac{1}{(M_1-1)N} \sum_{i=1}^{M-1}\sum_{n=1}^{N} \Bigg\{ & \frac{\tau^{-0.5}}{\Gamma(1.5)} \left[a_0^{0.5}\frac{\partial \tilde{u}(x_i,t_n)}{\partial x_i} + \sum_{k=1}^{n-1}(a_{n-k}^{0.5} - a_{n-k-1}^{0.5})\frac{\partial \tilde{u}(x_i,t_k)}{\partial x_i} \right] \\ & + \tilde{u}(x_i,t_n)\frac{\partial^2 \tilde{u}(x_i,t_n)}{\partial x_i^2} + \left(\frac{\partial \tilde{u}(x_i,t_n)}{\partial x_i}\right)^2 - \frac{\partial^3 \tilde{u}(x_i,t_n)}{\partial x_i^3} - \frac{\partial f(x_i,t_n)}{\partial x_i} \Bigg\}^2, \end{aligned} \tag{4.4}$$

$$L_{g_2FW} = \frac{1}{(M_1-1)N}\sum_{i=1}^{M-1}\sum_{n=1}^{N}\left\{\frac{\tau^{-0.5}}{\Gamma(1.5)}\left[a_0^{0.5}\frac{\partial\tilde{u}(x_i,t_n)}{\partial t_n}+\sum_{k=1}^{n-1}(a_{n-k}^{0.5}-a_{n-k-1}^{0.5})\frac{\partial\tilde{u}(x_i,t_k)}{\partial t_k}\right]\right.$$
$$\left.+\,\tilde{u}(x_i,t_n)\frac{\partial^2\tilde{u}(x_i,t_n)}{\partial x_i\partial t_n}+\frac{\partial\tilde{u}(x_i,t_n)}{\partial x_i}\frac{\partial\tilde{u}(x_i,t_n)}{\partial t_n}-\frac{\partial^3\tilde{u}(x_i,t_n)}{\partial x_i^2\partial t_n}-\frac{\partial f(x_i,t_n)}{\partial t_n}\right\}^2. \tag{4.5}$$

By substituting Eqs (4.3)–(4.5) into Eq (3.8), we get the gfPINNs loss function L_{gFW} with $W_F = 1$, $W_{g_1F} = 1$, and $W_{g_2F} = 1$. Next, we selected 2000 training points to train fPINNs and gfPINNs and other parameters of the NNs are set to those described at the beginning of this section. Figures 2–4 present a comparison between the predicted solutions from the fPINNs and gfPINNs models and the exact solution of the equation, demonstrating that gfPINNs can effectively solve the equation. Figure 5 shows the absolute errors between the exact solution and the solutions predicted by fPINNs and gfPINNs, and it can be seen that the prediction performance of gfPINNs is better than that of fPINNs. Figure 6 illustrates the L^2 relative errors of both fPINNs and gfPINNs models for a single experiment as the iteration count varies, showing that while both can achieve errors as low as 10^{-4}, gfPINNs exhibits comparatively lower error and reduced oscillation.

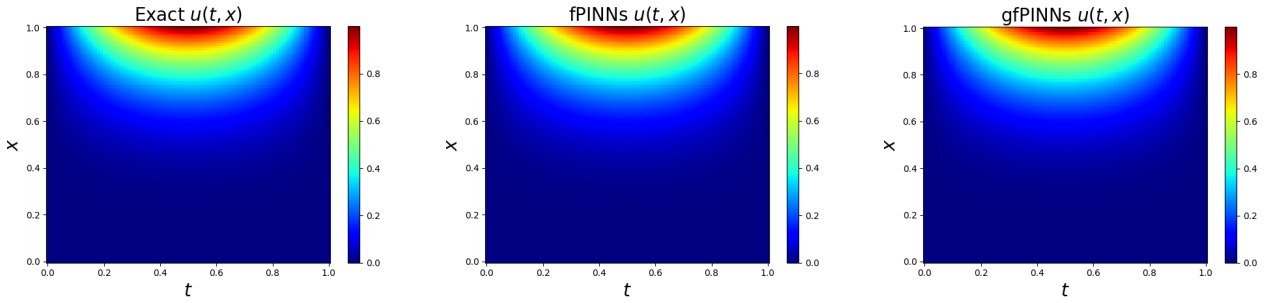


Figure 2. The exact solution and predicted solutions of the equation.

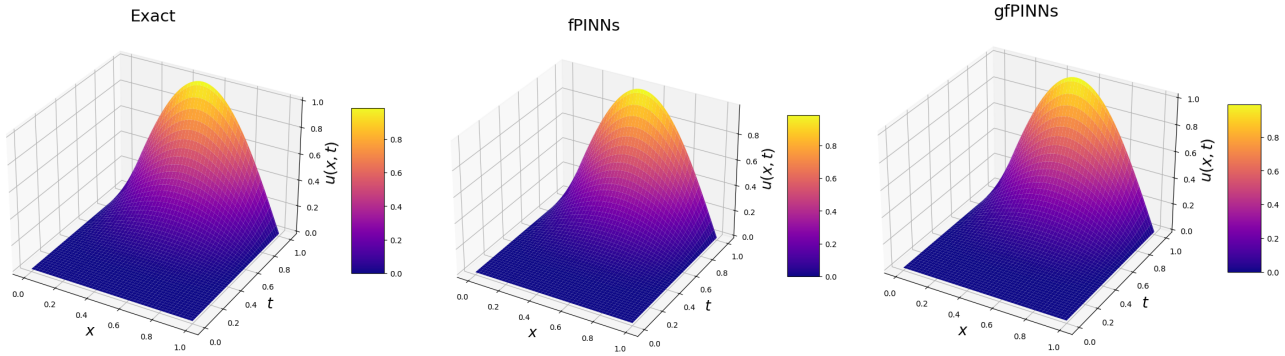


Figure 3. The exact solution and numerical solutions' profiles of velocity $u(x,t)$ with $\alpha = 0.5$.

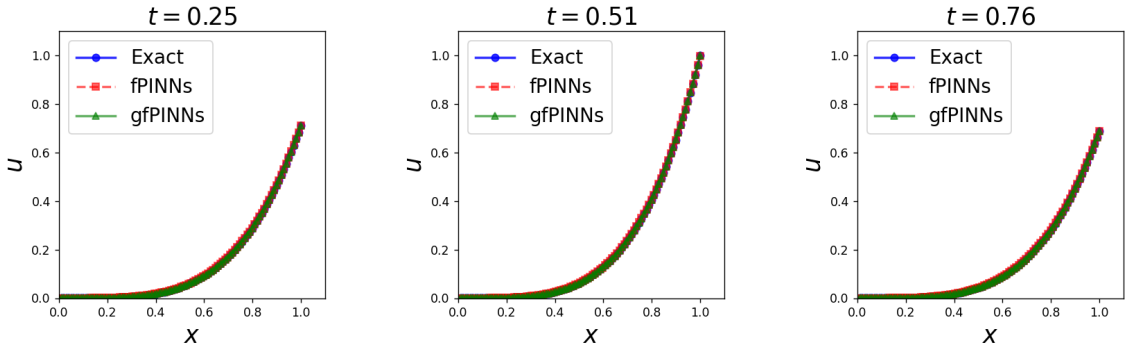
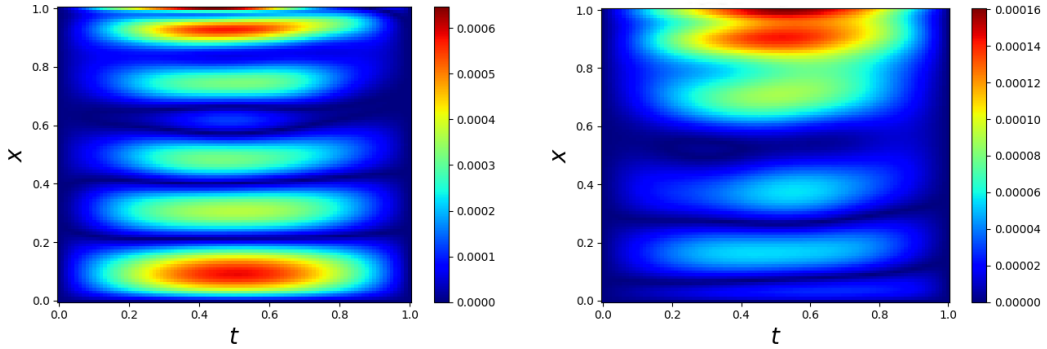


Figure 4. Predicted cross-sectional views of the equation using fPINNs and gfPINNs.



(a) Absolute error for fPINNs **(b)** Absolute error for gfPINNs

Figure 5. The absolute errors for solutions predicted by fPINNs and gfPINNs.

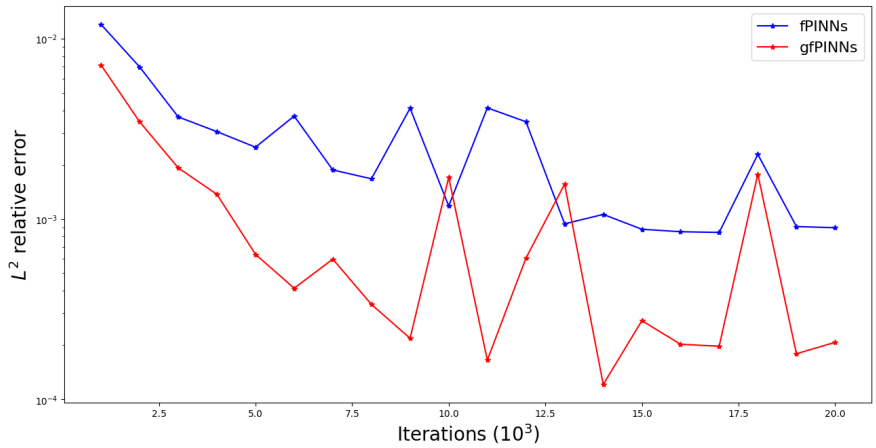


Figure 6. The L^2 relative error of the problem with the number of iterations.

Case 2: We choose $c_1 = 0$, $c_2 = 1$, and $\gamma = 1.5$, considering the smooth fabricated solution $u(x,t) = t^4 \sin(\pi x)$ and the forcing term $f(x,t) = 2.5\frac{\Gamma(5)}{\Gamma(3.5)}t^{2.5}\sin(\pi x) + t^8 \sin(\pi x)\cos(\pi x) + \pi^2 t^4 \sin(\pi x)$.

Similarly, we give the loss function of fPINNs as

$$L_{FW} = \frac{1}{(M_1-1)N}\sum_{i=1}^{M-1}\sum_{n=1}^{N}\left\{\frac{\tau^{-0.5}}{\Gamma(1.5)}\left[b_0^{1.5}\frac{\partial\tilde{u}(x_i,t_n)}{\partial t_n}+\sum_{k=1}^{n-1}(b_{n-k}^{1.5}-b_{n-k-1}^{1.5})\frac{\partial\tilde{u}(x_i,t_n)}{\partial t_k}\right]\right.$$
$$\left.+\tilde{u}(x_i,t_n)\frac{\partial\tilde{u}(x_i,t_n)}{\partial x_i}-\frac{\partial^2\tilde{u}(x_i,t_n)}{\partial x_i^2}-f(x_i,t_n)\right\}^2. \tag{4.6}$$

For gfPINNs, the loss function can be written as

$$L_{g_1FW} = \frac{1}{(M_1-1)N}\sum_{i=1}^{M-1}\sum_{n=1}^{N}\left\{\frac{\tau^{-0.5}}{\Gamma(1.5)}\left[b_0^{1.5}\frac{\partial^2\tilde{u}(x_i,t_n)}{\partial t_n\partial x_i}+\sum_{k=1}^{n-1}(b_{n-k}^{1.5}-b_{n-k-1}^{1.5})\frac{\partial^2\tilde{u}(x_i,t_k)}{\partial t_k\partial x_i}\right]\right.$$
$$\left.+\tilde{u}(x_i,t_n)\frac{\partial^2\tilde{u}(x_i,t_n)}{\partial x_i^2}+\left(\frac{\partial\tilde{u}(x_i,t_n)}{\partial x_i}\right)^2-\frac{\partial^3\tilde{u}(x_i,t_n)}{\partial x_i^3}-\frac{\partial f(x_i,t_n)}{\partial x_i}\right\}^2, \tag{4.7}$$

$$L_{g_2FW} = \frac{1}{(M_1-1)N}\sum_{i=1}^{M-1}\sum_{n=1}^{N}\left\{\frac{\tau^{-0.5}}{\Gamma(1.5)}\left[b_0^{1.5}\frac{\partial^2\tilde{u}(x_i,t_n)}{\partial t_n^2}+\sum_{k=1}^{n-1}(b_{n-k}^{1.5}-b_{n-k-1}^{1.5})\frac{\partial^2\tilde{u}(x_i,t_k)}{\partial t_k^2}\right]\right.$$
$$\left.+\tilde{u}(x_i,t_n)\frac{\partial^2\tilde{u}(x_i,t_n)}{\partial x_i\partial t_n}+\frac{\partial\tilde{u}(x_i,t_n)}{\partial x_i}\frac{\partial\tilde{u}(x_i,t_n)}{\partial t_n}-\frac{\partial^3\tilde{u}(x_i,t_n)}{\partial x_i^2\partial t_n}-\frac{\partial f(x_i,t_n)}{\partial t_n}\right\}^2. \tag{4.8}$$

By substituting Eqs (4.6)–(4.8) into Eq (3.8), we get the gfPINNs loss function L_{gFW} with $W_F = 1$, $W_{g_1F} = 0.16$, and $W_{g_2F} = 0.16$. Next, we selected 2000 training points to train fPINNs and gfPINNs and other parameters of the NNs are set to those described at the beginning of this section. Figures 7–9 present a comparison between the predicted solutions from the fPINNs and gfPINNs models and the exact solution of the equation, demonstrating that gfPINNs can effectively solve the equation. Figure 10 illustrates the absolute errors between the exact solution and the solutions predicted by both fPINNs and gfPINNs, revealing that the gfPINNs exhibit a relatively smaller absolute error. Figure 11 presents the iteration convergence curves for both the fPINNs and gfPINNs models for a single experiment, revealing that while both can achieve L^2 relative errors of 10^{-4} with increasing iterations, the prediction errors of gfPINNs are relatively low and more stable, resulting in superior prediction performance compared to fPINNs.

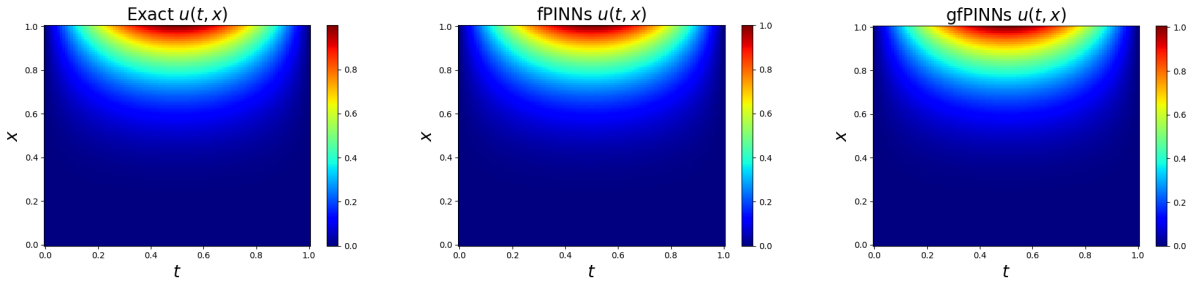


Figure 7. The exact solution and predicted solutions of the equation.

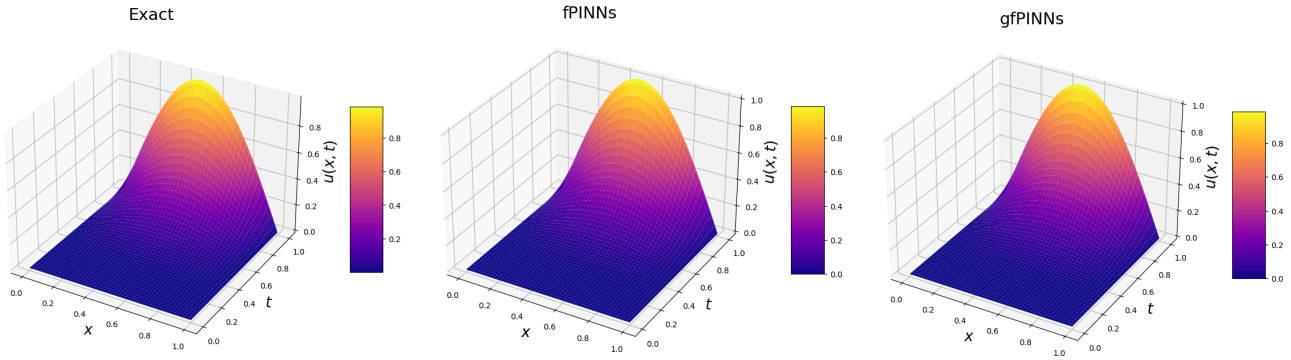


Figure 8. The exact solution and numerical solutions' profiles of velocity $u(x, t)$ with $\gamma = 1.5$.

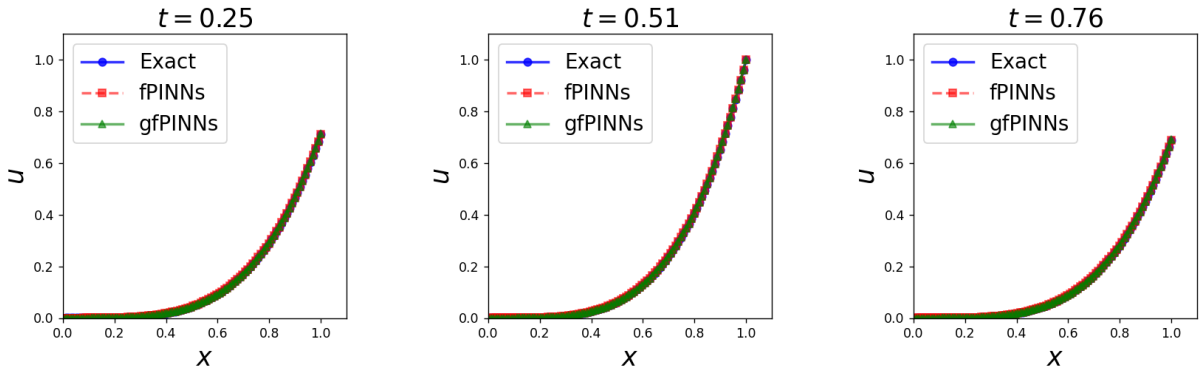
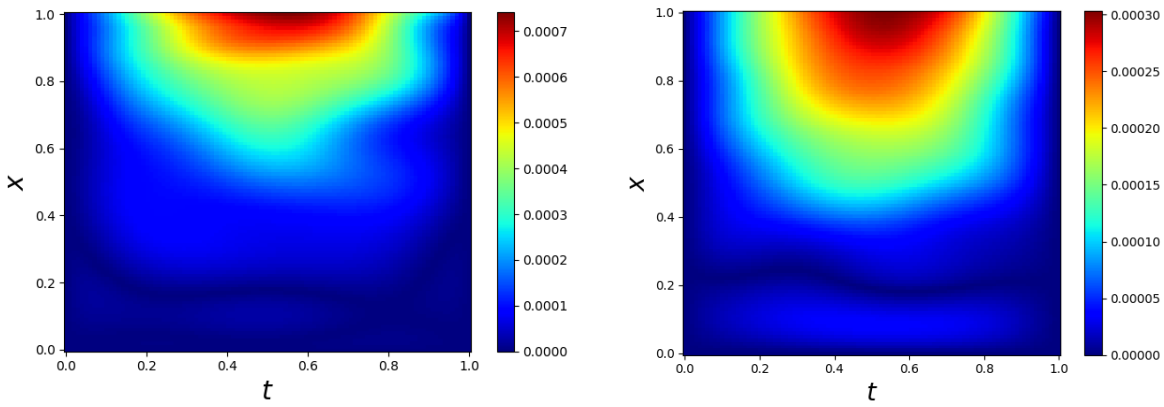


Figure 9. Predicted cross-sectional views of the equation using fPINNs and gfPINNs.



(a) Absolute error for fPINNs **(b)** Absolute error for gfPINNs

Figure 10. The absolute errors for solutions predicted by fPINNs and gfPINNs.

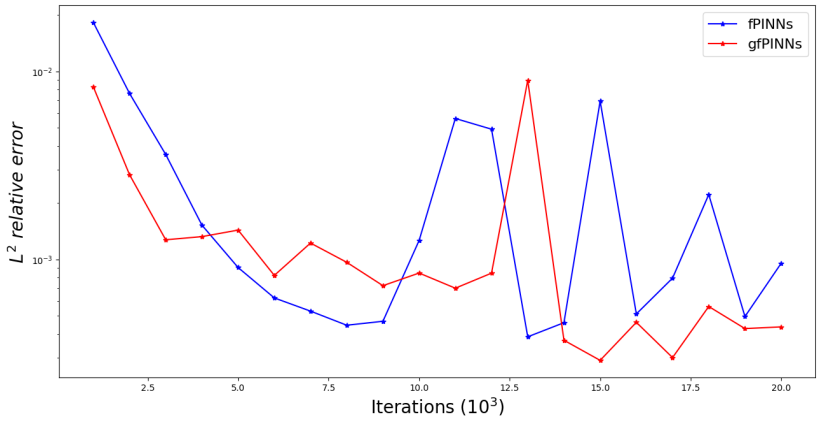


Figure 11. The L^2 relative error of the problem with the number of iterations.

4.2. Inverse problems

We use the code that solves the forward problem to solve the inverse problem. We simply add the parameters to be identified in the inverse problem to the list of parameters to be optimized in the forward problem, without changing anything else. In this section, gfPINNs are applied to solve the inverse problems of the multiterm time-fractional Burger-type equation of the form (3.6). We let $\upsilon = 1$, $(x, t) \in [0, 1] \times [0, 1]$, $g(x) = 0$, and considering additional concentration measurements at the final time $u(x, 1) = h(x, 1)$. Here, we still consider the smooth fabricated solution $u(x, t) = t^p \sin(\pi x)$ and the forcing term of formula (4.2).

Case 1: We choose $c_1 = 1$ and $c_2 = 0$. Similarly, we get the gfPINNs loss function L_{gFW} with $W_I = 1$, $W_{g_1 I} = 0.25$, and $W_{g_2 I} = 0.25$. We set the fractional derivative to be 0.6. We selected 470 training points to train fPINNs and gfPINNs and other parameters of the NNs are set to those described at the beginning of this section. Figures 12–14 display a comparison between the predicted solutions from the fPINNs and gfPINNs models and the exact solution of the equation, demonstrating that gfPINNs can effectively solve the problem. Figure 15 illustrates the absolute errors between the exact solution and the solutions predicted by both fPINNs and gfPINNs, revealing that the gfPINNs exhibit a relatively smaller and more stable absolute error. Figure 16 illustrates the iteration convergence curves for the fPINNs and gfPINNs for a single experiment, indicating that although gfPINNs incur a higher computational cost for solving the inverse problem due to an additional loss term, both models can achieve L^2 relative errors of 10^{-4} as iterations progress, with gfPINNs showing a lower and more stable error curve compared to fPINNs.

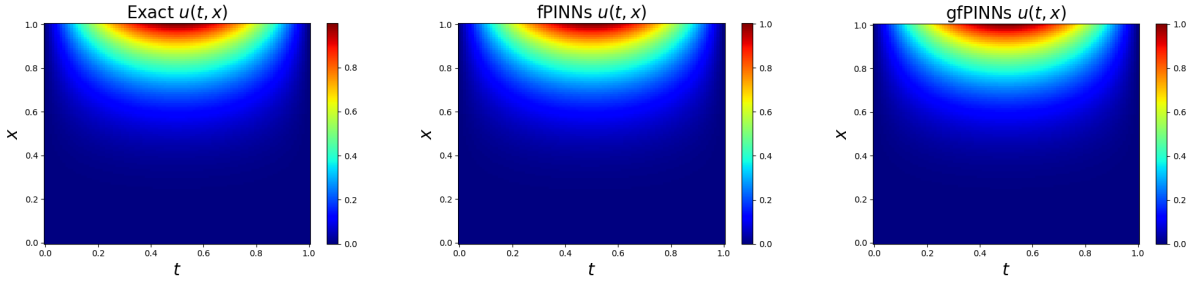


Figure 12. The exact solution and predicted solutions of the equation.

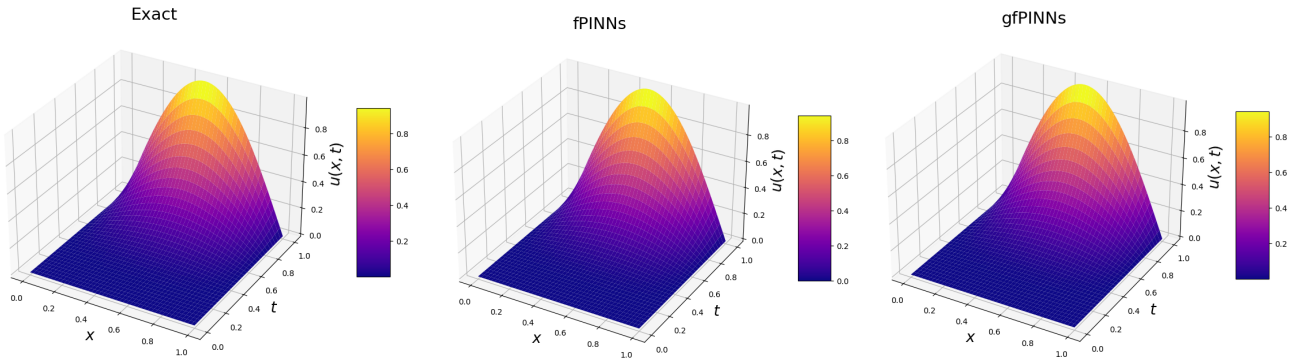


Figure 13. The exact solution and numerical solutions' profiles of velocity $u(x, t)$.

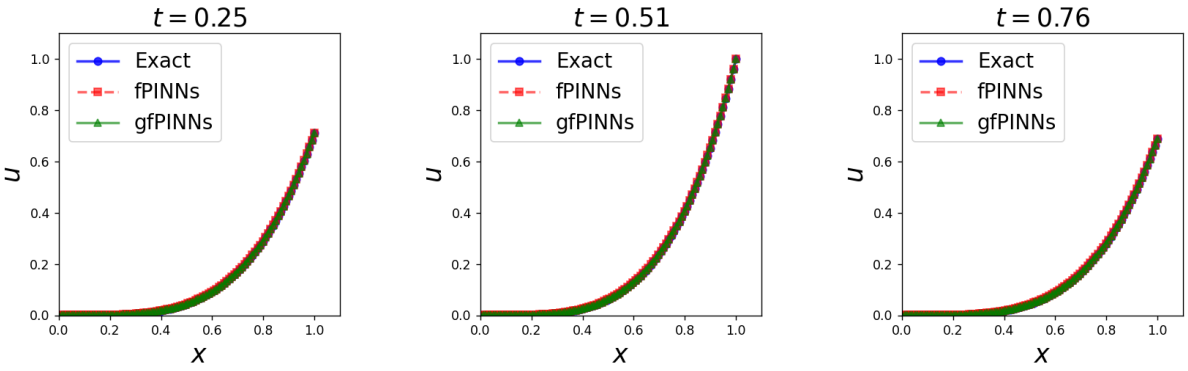
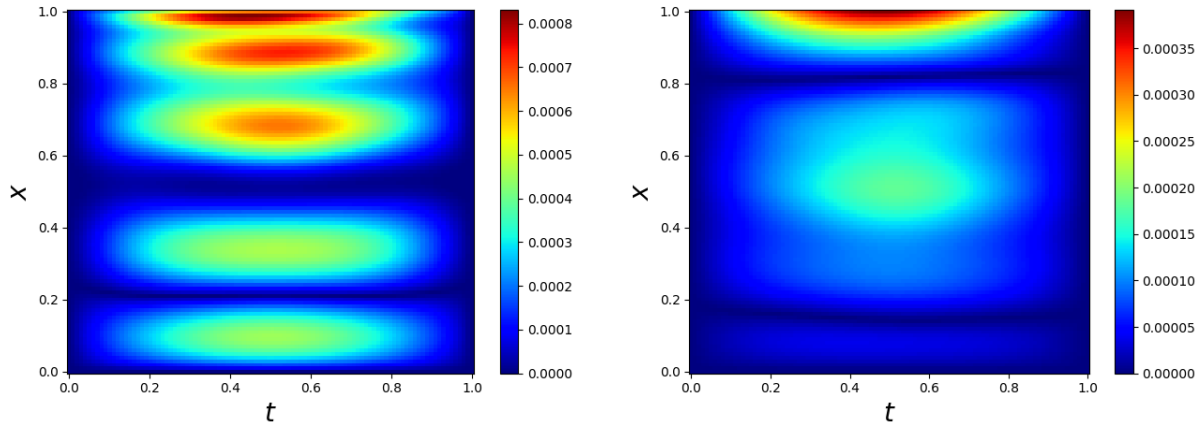


Figure 14. Predicted cross-sectional views of the equation using fPINNs and gfPINNs.



(a) Absolute error for fPINNs **(b)** Absolute error for gfPINNs

Figure 15. The absolute errors for solutions predicted by fPINNs and gfPINNs.

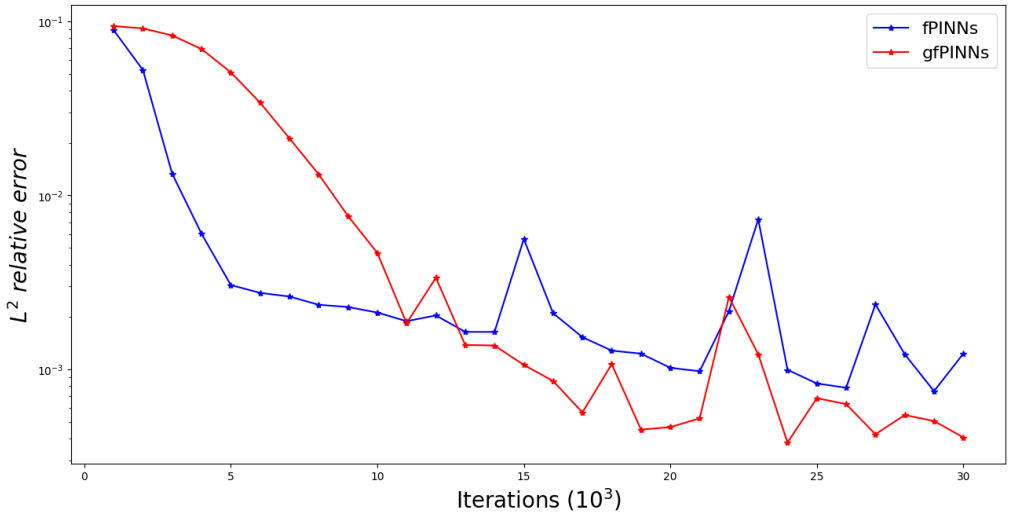


Figure 16. The L^2 relative error of the problem with the number of iterations.

Case 2: We choose $c_1 = 0$ and $c_2 = 1$. Similarly, we get the gfPINNs loss function L_{gFW} with $W_I = 1$, $W_{g_1 I} = 0.16$, and $W_{g_2 I} = 0.0001$. We set the fractional derivative to be 1.6. We selected 400 training points to train fPINNs and gfPINNs and other parameters of the NNs are set to those described at the beginning of this section. For fPINNs and gfPINNs, we get the similar conclusion as Case 1 by training the NNs and observing the experimental results. Figures 17–19 display a comparison between the predicted solutions from the fPINNs and gfPINNs models and the exact solution of the equation, demonstrating that gfPINNs can effectively solve the problem. Figure 20 illustrates the absolute errors between the exact solution and the solutions predicted by both fPINNs and gfPINNs, revealing that the gfPINNs exhibit a relatively smaller absolute error. Figure 21 compares the L^2 relative errors of fPINNs and gfPINNs for a single experiment as iterations progress, revealing that while gfPINNs incurs a higher computational cost due to an additional loss term, both models can achieve an L^2 relative error of 10^{-3}, with gfPINNs demonstrating a lower and more stable error curve than fPINNs.

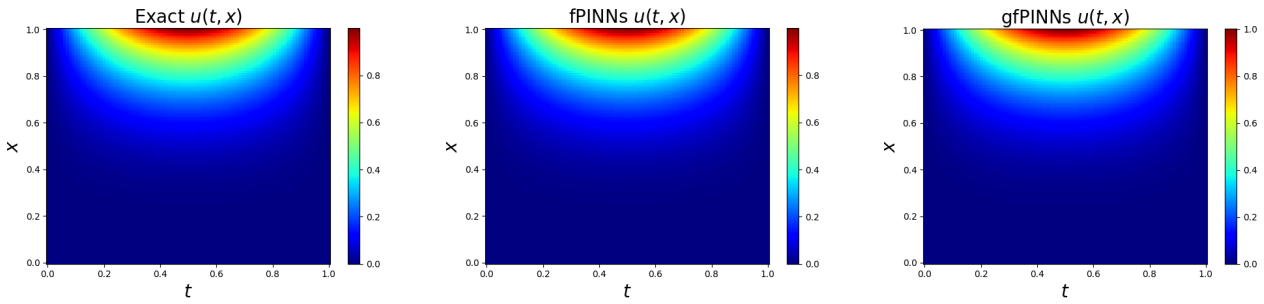


Figure 17. The exact solution and predicted solutions of the equation.

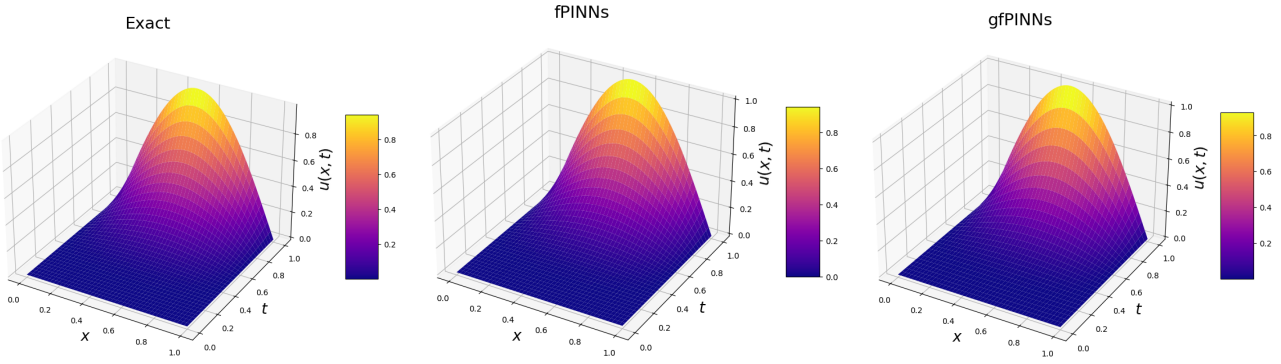


Figure 18. The exact solution and numerical solutions' profiles of velocity $u(x,t)$.

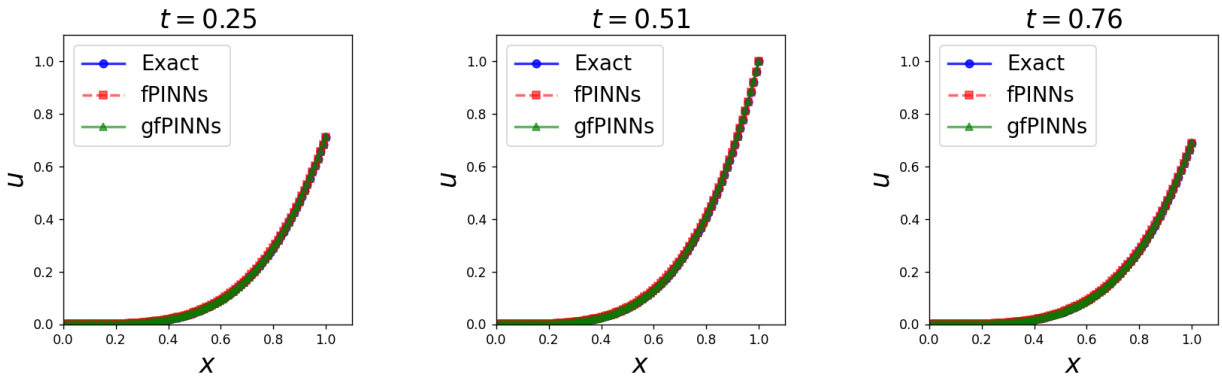
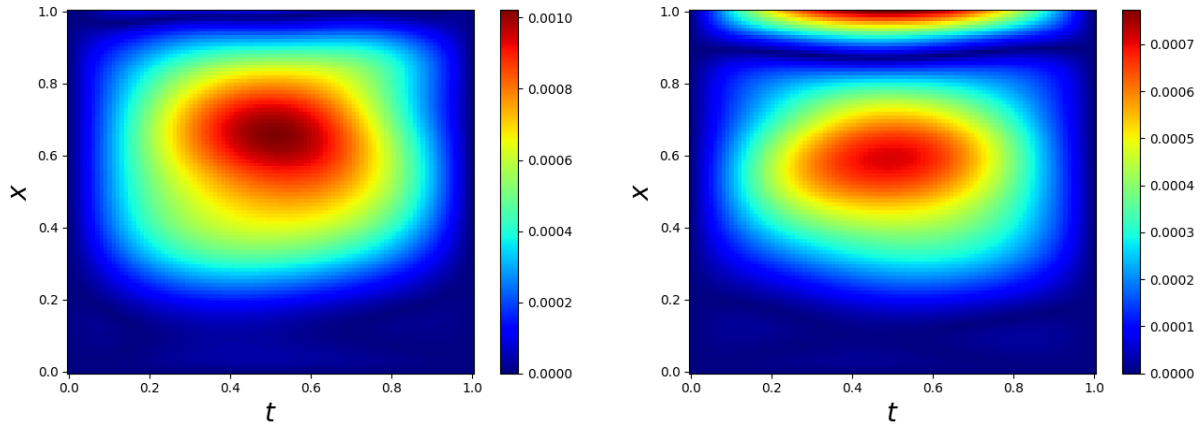


Figure 19. Predicted cross-sectional views of the equation using fPINNs and gfPINNs.



(a) Absolute error for fPINNs **(b)** Absolute error for gfPINNs

Figure 20. The absolute errors for solutions predicted by fPINNs and gfPINNs.

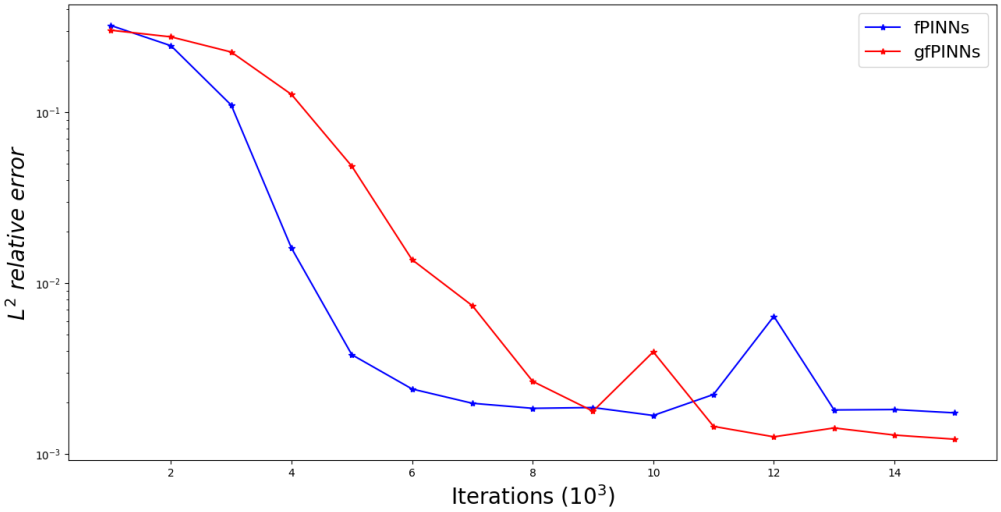


Figure 21. The L^2 relative error of the problem with the number of iterations.

5. Conclusions

In this paper, the effectiveness of gfPINNs in solving the forward and inverse problems of the multiterm time-fractional Burger-type equation is verified through numerical examples. The L^2 relative errors for solutions predicted by both fPINNs and gfPINNs can achieve 10^{-4} for forward problems and 10^{-3} or even 10^{-4} for inverse problems. The experimental results indicate that gfPINNs demonstrate relatively lower and more stable errors with the increase of training iterations, thereby enhancing prediction performance. Nonetheless, the inclusion of an additional loss term in gfPINNs may result in a higher computational cost, such as when solving inverse problems, fPINNs exhibit faster convergence compared to gfPINNs.

Author contributions

Shanhao Yuan, Yanqin Liu, Qiuping Li and Chao Guo: Conceptualization, Methodology; Yibin Xu, Shanhao Yuan and Yanfeng Shen: Software, Visualization, Validation; Shanhao Yuan: Writing–Original draft preparation; Yanqin Liu: Writing–Reviewing & editing. All authors have read and approved the final version of the manuscript for publication.

Acknowledgments

We appreciated the support by the Natural Science Foundation of Shandong Province (ZR2023MA062), the National Science Foundation of China (62103079), the Belt and Road Special Foundation of The National Key Laboratory of Water Disaster Prevention (2023491911), and the Open Research Fund Program of the Data Recovery Key Laboratory of Sichuan Province (DRN19020).

Conflict of interest

The authors declare that they have no conflicts of interest.

References

1. L. Cristofaro, R. Garra, E. Scalas, I. Spassiani, A fractional approach to study the pure-temporal epidemic type aftershock sequence (ETAS) process for earthquakes modeling, *Fract. Calc. Appl. Anal.*, **26** (2023), 461–479. https://doi.org/10.1007/s13540-023-00144-5
2. Y. Zhang, H. G. Sun, H. H. Stowell, M. Zayernouri, S. E. Hansen, A review of applications of fractional calculus in earth system dynamics, *Chaos Solitons Fract.*, **102** (2017), 29–46. https://doi.org/10.1016/j.chaos.2017.03.051
3. M. I. Molina, Fractional electrical impurity, *New J. Phys.*, **26** (2024), 013020. https://doi.org/10.1088/1367-2630/ad19f8
4. Y. Q. Yang, Q. W. Qi, J. Y. Hu, J. S. Dai, C. D. Yang, Adaptive fault-tolerant control for consensus of nonlinear fractional-order multi-agent systems with diffusion, *Fractal Fract.*, **7** (2023), 1–20. https://doi.org/10.3390/fractalfract7100760
5. P. Baliarsingh, L. Nayak, Fractional derivatives with variable memory, *Iran. J. Sci. Technol. Trans. A Sci.*, **46** (2022), 849–857. https://doi.org/10.1007/s40995-022-01296-4
6. J. B. Hu, Studying the memory property and event-triggered control of fractional systems, *Inform. Sci.*, **662** (2024), 120218. https://doi.org/10.1016/j.ins.2024.120218
7. J. Guo, D. Xu, W. L. Qiu, A finite difference scheme for the nonlinear time-fractional partial integro-differential equation, *Math. Methods Appl. Sci.*, **43** (2020), 3392–3412. https://doi.org/10.1002/mma.6128
8. H. Z. Hu, Y. P. Chen, J. W. Zhou, Two-grid finite element method for time-fractional nonlinear schrodinger equation, *J. Comp. Math.*, **42** (2024), 1124–1144. https://doi.org/10.4208/jcm.2302-m2022-0033
9. W. Zhang, C. X. Wu, Z. S. Ruan, S. F. Qiu, A Jacobi spectral method for calculating fractional derivative based on mollification regularization, *Asymptot. Anal.*, **136** (2024), 61–77. https://doi.org/10.3233/ASY-231869
10. Q. L. Gu, Y. P. Chen, J. W. Zhou, J. Huang, A fast linearized virtual element method on graded meshes for nonlinear time-fractional diffusion equations, *Numer. Algorithms*, 2024. https://doi.org/10.1007/s11075-023-01744-1
11. S. S. Yu, M. Guo, X. Y. Chen, J. L. Qiu, J. Q. Sun, Personalized movie recommendations based on a multi-feature attention mechanism with neural networks, *Mathematics*, **11** (2023), 1–22. https://doi.org/10.3390/math11061355
12. X. Y. Ding, J. Q. Lu, X. Y. Chen, Lyapunov-based stability of time-triggered impulsive logical dynamic networks, *Nonlinear Analy. Hybrid Syst.*, **51** (2024), 101417. https://doi.org/10.1016/j.nahs.2023.101417
13. T. G. Yang, G. C. Li, T. Y. Wang, S. Y. Yuan, X. Y. Yang, X. G. Yu, et al., A novel 1D-convolutional spatial-time fusion strategy for data-driven fault diagnosis of aero-hydraulic pipeline systems, *Mathematics*, **11** (2023), 1–21. https://doi.org/10.3390/math11143113
14. L. Lu, Y. H. Su, G. E. Karniadakis, Collapse of deep and narrow neural nets, 2018, arXiv: 1808.04947.

15. Y. Q. Liu, T. Mao, D. X. Zhou, Approximation of functions from Korobov spaces by shallow neural networks, *Inform. Sci.*, **670** (2024), 120573. https://doi.org/10.1016/j.ins.2024.120573

16. G. A. Anastassiou, D. Kouloumpou, Neural network approximation for time splitting random functions, *Mathematics*, **11** (2023), 1–25. https://doi.org/10.3390/math11092183

17. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed 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

18. Q. Z. Hou, Y. X. Li, V. P. Singh, Z. W. Sun, Physics-informed neural network for diffusive wave model, *J. Hydrology*, **637** (2024), 131261. https://doi.org/10.1016/j.jhydrol.2024.131261

19. S. M. Sivalingam, P. Kumar, V. Govindaraj, A neural networks-based numerical method for the generalized Caputo-type fractional differential equations, *Math. Comput. Simul.*, **213** (2023), 302–323. https://doi.org/10.1016/j.matcom.2023.06.012

20. Q. Z. Hou, Y. X. Li, V. P. Singh, Z. W. Sun, J. G. Wei, Physics-informed neural network for solution of forward and inverse kinematic wave problems, *J. Hydrology*, **633** (2024), 130934. https://doi.org/10.1016/j.jhydrol.2024.130934

21. H. Bararnia, M. Esmaeilpour, On the application of physics informed neural networks (PINN) to solve boundary layer thermal-fluid problems, *Int. Commun. Heat Mass Transfer*, **132** (2022), 105890. https://doi.org/10.1016/j.icheatmasstransfer.2022.105890

22. X. P. Zhang, Y. Zhu, J. Wang, L. L. Ju, Y. Z. Qian, M. Ye, et al., GW-PINN: a deep learning algorithm for solving groundwater flow equations, *Adv. Water Resour.*, **165** (2022), 104243. https://doi.org/10.1016/j.advwatres.2022.104243

23. S. P. Zheng, Y. Y. Lin, J. H. Feng, F. Jin, Viscous regularization PINN algorithm for shallow water equations (Chinese), *Chinese J. Comput. Phys.*, **40** (2023), 314–324.

24. G. F. Pang, L. Lu, G. E. Karniadakis, fPINNs: fractional physics-informed neural networks, *SIAM J. Sci. Comput*, **41** (2019), A2603–A2626. https://doi.org/10.1137/18M1229845

25. 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

26. S. P. Wang, H. Zhang, X. Y. Jiang, Fractional physics-informed neural networks for time-fractional phase field models, *Nonlinear Dyn.*, **110** (2022), 2715–2739. https://doi.org/10.1007/s11071-022-07746-3

27. 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

28. 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

29. 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

30. J. Yu, L. Lu, X. H. Meng, G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, *Comput. Methods Appl. Mech. Eng.*, **393** (2022), 114823. https://doi.org/10.1016/j.cma.2022.114823

31. G. Z. Xie, B. B. Fu, H. Li, W. L. Du, Y. D. Zhong, L. W. Wang, et al., A gradient-enhanced physics-informed neural networks method for the wave equation, *Eng. Anal. Bound. Elem.*, **166** (2024), 105802. https://doi.org/10.1016/j.enganabound.2024.105802

32. Y. X. Deng, G. Lin, X. Yang, Multifidelity data fusion via gradient-enhanced Gaussian process regression, *Commun. Comput. Phys.*, **28** (2020), 1812–1837. https://doi.org/10.4208/cicp.OA-2020-0151

33. Z. Z. Sun, X. N. Wu, A fully discrete difference scheme for a diffusion-wave system, *Appl. Numer. Math.*, **56** (2006), 193–209. https://doi.org/10.1016/j.apnum.2005.03.003

34. B. T. Jin, R. Lazarovl, Z. Zhou, An analysis of the L1 scheme for the subdiffusion equation with nonsmooth data, *IMA J. Numer. Anal.*, **36** (2016), 197–221. https://doi.org/10.1093/imanum/dru063

35. A. A. Kilbas, H. M. Srivastava, J. J. Trujillo, *Theory and applications of fractional differential equations*, Amsterdam: Elsevier, 2006.

36. I. Podlubny, *Fractional differential equations*, Academic Press, 1999.

37. M. Ramchandani, H. Khandare, P. Singh, P. Rajak, N. Suryawanshi, A. S. Jangde, et al., Survey: Tensorflow in machine learning, *J. Phys. Conf. Ser.*, **2273** (2022), 012008. https://doi.org/10.1088/1742-6596/2273/1/012008

38. D. P. Kingma, J. Ba, Adam: a method for stochastic optimization, In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, 2015.

39. M. V. Narkhede, P. P. Bartakke, M. S. Sutaone, A review on weight initialization strategies for neural networks, *Artif. Intell. Rev.*, **55** (2022), 291–322. https://doi.org/10.1007/s10462-021-10033-z

AIMS Press