Networks and Heterogeneous Media

http://www.aimspress.com/journal/nhm

NHM, 18(3): 1083–1104.
DOI: 10.3934/nhm.2023047
Received: 13 October 2022
Revised: 12 March 2023
Accepted: 13 March 2023
Published: 30 March 2023

Research article

An artificial neural network approach for a class of time-fractional diffusion and diffusion-wave equations

Yinlin Ye, Hongtao Fan, Yajing Li*, Ao Huang and Weiheng He

College of Science, Northwest A&F University, Yangling, Shanxi 712100, China

* **Correspondence:** E-mail: hliyajing@163.com, fanht17@nwafu.edu.cn.

Abstract: In this paper, the artificial neural network method is applied to solve the time-fractional diffusion and diffusion-wave equations. This method combines Taylor series and neural network method, and uses the terms of different power terms of Taylor series as neurons. An error function is given to update the weights of the proposed neural network. In addition, in order to balance the contributions of different error terms in the error function, we propose an adaptive weight adjustment method. In the end, four numerical examples are given to demonstrate the effectiveness of proposed method in solving the time-fractional diffusion and diffusion-wave equations.

Keywords: fractional diffusion equation; fractional diffusion-wave equation; artificial neural network method; Taylor series; adaptive weight adjustment

1. Introduction

Fractional calculus is an important branch of mathematics. As its theory has been successfully applied to various fields in recent years, people have gradually found that fractional calculus can describe some “abnormal” phenomena in natural science and engineering applications. For example, mechanical behavior of viscoelastic materials [1], plasma motion under high temperature and high pressure [2], abnormal diffusion of extremely cold atoms or cell protoplasm [3], etc. Time fractional partial differential equation is an important model. When its fractional order $\alpha \in (0, 1)$, it is called time-fractional diffusion equation; when $\alpha \in (1, 2)$, it is called time-fractional diffusion-wave equation. The time-fractional diffusion equation can describe the abnormal slow diffusion phenomenon in porous media with fractal structure [4], while the time-fractional diffusion-wave equation can describe the propagation of mechanical scattered waves in viscoelastic media with power-law creep characteristics [5].

It is well known that the analytical solution of time-fractional diffusion and diffusion-wave equations is notoriously difficult to obtain. In order to promote the development and progress of science, a great quantity of researchers and scholars have studied the numerical approximation method of time-fractional diffusion and diffusion-wave equations. In recent years, Gu and his collaborators have proposed many effective methods to solve the time-fractional diffusion equations, such as quadratic spline collocation method [6], Lagrange interpolation method [7], parallel-in-time iterative algorithm [8] and the graded L1 scheme method [9]. For the time-fractional diffusion-wave equations, many classical numerical methods have been proposed, including finite difference method [10, 11], element-free Galerkin method [12], wavelets method [13, 14], meshless local collocation method [15], etc.

Moreover, artificial neural network method is also proposed to solve fractional differential equations. With the rapid development of machine learning technology, the demand for solving fractional differential equations by using artificial neural network method has increased by leaps and bounds, and some excellent results have been achieved. In 2017, Qu [16] applied cosine radial basis function neural network to solve fractional differential equations with initial value or boundary value problems for the first time. In 2018, Rostami et al. [17] adopted an iterative algorithm combining power series and neural network method to approximately seek the solution of high-order linear and ordinary differential equations. Rizaner et al. [18] proposed a numerical approch for approximately solving the first-order initial value problem using unsupervised radial basis function network, and verified the effectiveness with some numerical examples. In 2020, Hadian-Rasanan et al. [20] solved different types of Lane-Emden equation by constructing a single-layer orthogonal network with a fractional Legendre function as the activation function of the hidden layer, and used the Levenberg-Marquardt algorithm to train them. The above literature is to solve fractional ordinary differential equations by making use of artificial neural network method. However, there are few literatures on solving time-fractional partial equations by using artificial neural networks. Jafarian et al. regarded the artificial neural network method with a generalized sigmoid function as the cost function as the iterative scheme for solving linear fractional ordinary differential equations in [19]. Recently, Qu et al. [21] proposed a neural network method based on cosine basis and sine basis to solve a class of time-fractional diffusion and diffusion-wave equation with zero boundary conditions. Deep neural network methods for solving forward and inverse problems of time-fractional diffusion equations with conformable derivatives were introduced by Ye et al. [22]. In this paper, a new numerical method is proposed for a class of time-fractional diffusion and diffusion-wave equations with Dirichlet boundary conditions by combining Taylor series method with artificial neural network.

In this paper, We consider the time-fractional diffusion equation with the following form:

$$\begin{cases} {}_cD_{0,t}^{\alpha}u(x,t) + c^2u_{xx}(x,t) = f(x,t), \\ u(0,t) = \lambda(t), u(L,t) = \mu(t), \\ u(x,0) = \varphi(x), \\ 0 < \alpha < 1, 0 < x \leqslant P, 0 < t \leqslant Q, \end{cases} \tag{1.1}$$

and the time-fractional diffusion-wave equation with the form as follows:

$$\begin{cases} {}_cD_{0,t}^{\alpha}u(x,t) + c^2u_{xx}(x,t) = f(x,t), \\ u(0,t) = \lambda(t), u(L,t) = \mu(t), \\ u(x,0) = \varphi(x), u_t(x,0) = \psi(x) = 0, \\ 1 < \alpha < 2, 0 < x \leqslant P, 0 < t \leqslant Q. \end{cases} \tag{1.2}$$

For the time-fractional diffusion equation (1.1) and the time-fractional diffusion-wave equation (1.2), we establish a single-layer neural network model with Taylor series of different power terms as neurons, and give the error iteration formula and neural network weight update formula. It is worth mentioning that we have adaptively adjusted the weights of different error terms, which will be shown in Section 4.

The main contributions of this paper are as below:

- The single layer functional link artificial neural network based on Taylor series is established for the first time to solve the time-fractional diffusion and diffusion-wave equation with more general form.
- The error function composed of initial error term, boundary error term and internal error term is constructed to update the unknown parameters in the neural network. In addition, the adaptive weight adjustment method is proposed to balance the different error terms.
- Compared with the traditional iterative numerical method, the proposed neural network method can calculate the solution at any position in the domain without repeating the whole iterative process.

The rest of this paper is arranged as follows. Section 2 briefly introduces the related definition and important properties of fractional calculus. In Section 3, we present the main idea of our neural network methods for solving the time-fractional diffusion and diffusion-wave equations (1.1), (1.2). In Section 4, we propose an adaptive weight adjustment method to adjust the weights of different error terms in the iteration. After that, four numerical examples are given to illustrate the effectiveness of the method in Section 5. Finally, we summarize the main work of our paper in Section 6.

2. Preliminaries

This section gives some basic definition and related properties about fractional calculus; for more details see [23].

Definition 2.1. The Riemann-Liouville fractional integral of order $\alpha > 0$ for a function $f(x) \in L^1((0, Q])$ is defined as

$$_{RL}I^{\alpha}_{0+}(f(x)) = \frac{1}{\Gamma(\alpha)} \int_0^x (x-t)^{\alpha-1} f(t)\,\mathrm{d}t,$$

where $\Gamma(\cdot)$ denotes the Gamma function.

Definition 2.2. The Caputo fractional derivative of order $\alpha > 0$ for a function $f(x) \in AC((0, Q])$ with $n \in \mathbb{N}^+$ is defined as

$$_{C}D^{\alpha}_{0,x}f(x) = \begin{cases} {}_{RL}I^{n-\alpha}_{0+}\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^n (f(x)), & n-1 < \alpha < n, \\ \left(\frac{\mathrm{d}}{\mathrm{d}x}\right)^n f(x), & \alpha = n. \end{cases}$$

Since the derivative of a constant is 0, the following useful property holds in the Caputo sense:

$$_{C}D^{\alpha}_{0,x}(x^k) = \begin{cases} 0, & k \in Z^+, k < \lceil \alpha \rceil, \\ \frac{\Gamma(k+1)}{\Gamma(k+1-\alpha)} x^{k-\alpha}, & k \in Z^+, k \geq \lceil \alpha \rceil, x > 0. \end{cases}$$

Definition 2.3. The Mittag-Leffler function is defined by

$$E_{\alpha,\beta}(z)=\sum_{k=0}^{\infty}\frac{z^k}{\Gamma(\alpha k+\beta)},$$

where $z,\beta\in\mathbb{C}$, $\alpha>0$, and $\mathbb{C}$ is the usual set of complex numbers. Particularly, $E_{1,1}(z)=exp(z)$, that is, the common exponential function.

In order to facilitate the calculation of the Mittag-Leffler function later, the definition of the extended sine and cos functions and related properties are given [21, 23]. The calculation code of the Mittag-Leffler function can be found in [24].

Definition 2.4. The extended sine function $Sin_{\alpha,\beta}(x)$ and cos function $Cos_{\alpha,\beta}(x)$ are defined as

$$Sin_{\alpha,\beta}(x)=\sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^{2k-1}}{\Gamma(\alpha(2k-1)+\beta)},\quad Cos_{\alpha,\beta}(x)=\sum_{k=0}^{\infty}(-1)^{k}\frac{x^{2k}}{\Gamma(\alpha(2k)+\beta)},$$

where $x,\beta\in\mathbb{C}$, $\alpha>0$. According to the Euler formulas, we have

$$Sin_{\alpha,\beta}(x)=\frac{E_{\alpha,\beta}(ix)-E_{\alpha,\beta}(-ix)}{2i},\quad Cos_{\alpha,\beta}(x)=\frac{E_{\alpha,\beta}(ix)+E_{\alpha,\beta}(-ix)}{2}.$$

In particular, we have $Sin_{1,1}(x)=sin(x)$, $Cos_{1,1}(x)=cos(x)$.

Lemma 2.1. For the Caputo fractional derivative of some special functions, we have three useful results as follows:

$$\begin{aligned}{}_cD^{\alpha}_{0,t}(\exp(t))&={}_cD^{\alpha}_{0,t}(E_{1,1}(t))=t^{1-\alpha}E_{1,2-\alpha}(t),\\ {}_cD^{\alpha}_{0,t}(\sin(t))&={}_cD^{\alpha}_{0,t}(Sin_{1,1}(t))=t^{-\alpha}Sin_{1,1-\alpha}(t),\end{aligned}$$

and

$${}_cD^{\alpha}_{0,t}(\cos(t))={}_cD^{\alpha}_{0,t}(Cos_{1,1}(t))=t^{-\alpha}Cos_{1,1-\alpha}(t).$$

3. Methodology

With regard to the time-fractional diffusion equation (1.1) and the time-fractional diffusion-wave equation (1.2), the proposed method is described in detail in this section. In order to solve the above two problems, we built a single layer neural network with a series of rapidly convergent components of Taylor series as neurons, as shown in Figure 1.

In simple terms, we construct the k-th test solution in the following form:

$$u^k(x,t)=\sum_{i=0}^{N}\sum_{j=0}^{i}\theta^k_{\frac{i(i+1)}{2}+j+1}T_{ij},\tag{3.1}$$

with the first few Taylor components:

$$\begin{aligned}T_{00}(u)&=1,\\ T_{10}(u)&=t,\\ T_{01}(u)&=x,\end{aligned}$$

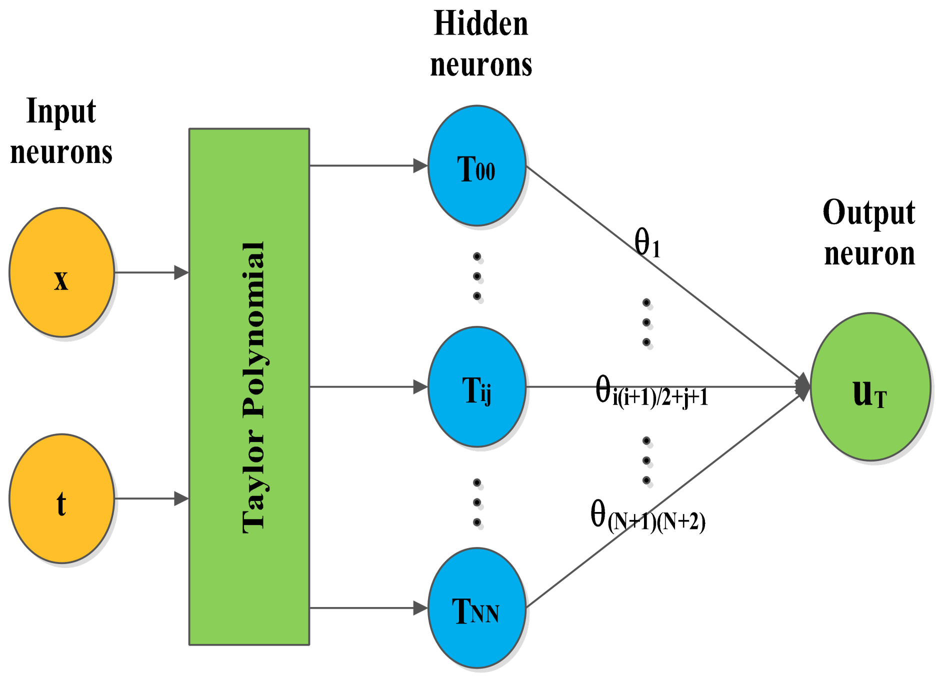


Figure 1. Neural network architecture for solving the problem (1.1) and (1.2).

and the higher order components of Taylor series may be expressed as the modalities as below:

$$T_{ij} = x^j t^{i-j}.$$

Here, $N \in \mathbb{N}_+$, $\frac{(N+1)(N+2)}{2}$ and $\theta^k_{\frac{i(i+1)}{2}+j+1}$ represent the highest power of the series, the number of neurons and the unknown weights of the neural network in the k-th iteration, respectively.

Since the weights of the neural network are unknown, in the first iteration, the weights are randomly preseted by a Gaussian distribution. In the subsequent iterations, we renew the weights through the error function. For the initial boundary value problems like Eqs (1.1) and (1.2), the error function is composed of the following three terms:

$$E^k = \frac{1}{2}\left\|E^k_{in}\right\|^2_F + \frac{1}{2}\left\|E^k_{ini}\right\|^2_2 + \frac{1}{2}\left\|E^k_{bou}\right\|^2_F, \tag{3.2}$$

where $\|\cdot\|_F$ is Frobenius matrix norm and $\|\cdot\|_2$ represents vector 2-norm. Let $x_m = \frac{(m-1)P}{N_m-1}$, $t_n = \frac{(n-1)Q}{N_n-1}$, $m = 1, 2, \cdots, N_m$, $n = 1, 2, \cdots, N_n$. The matrix E^k_{in} is

$$\begin{pmatrix} e^k_{2,2} & e^k_{2,3} & \cdots & e^k_{2,N_n} \\ e^k_{3,2} & e^k_{3,3} & \cdots & e^k_{3,N_n} \\ \vdots & \vdots & \ddots & \vdots \\ e^k_{N_m-1,2} & e^k_{N_m-1,3} & \cdots & e^k_{N_m-1,N_n} \end{pmatrix}, \tag{3.3}$$

with

$$e^k_{m,n} = {}_cD_{0,t}{}^{\alpha} u^k(x_m, t_n) + a^2 u^k_{xx}(x_m, t_n) - f(x_m, t_n), \tag{3.4}$$

for $2 \leq m \leq N_m - 1$, $2 \leq n \leq N_n$. The vector E^k_{ini} is

$$\begin{pmatrix} e^k_{1,1} \\ e^k_{2,1} \\ \vdots \\ e^k_{N_m,1} \end{pmatrix}, \tag{3.5}$$

with

$$e^k_{m,1} = u^k(x_m, 0) - \varphi(x_m). \tag{3.6}$$

The matrix E^k_{bou} is

$$\begin{pmatrix} e^k_{1,1} & e^k_{1,2} & \cdots & e^k_{1,N_n} \\ e^k_{N_m,1} & e^k_{N_m,2} & \cdots & e_{N_m,N^k_n} \end{pmatrix}, \tag{3.7}$$

with

$$e^k_{1,n} = u^k(0, t_n) - \lambda(t_n), \tag{3.8}$$

and

$$e^k_{N_m,n} = u^k(P, t_n) - \mu(t_n). \tag{3.9}$$

As for the internal points, by substituting the k-th test solution (3.1) into Eq (3.4) and making use of Definition 2.2, we further obtain that

$$\begin{aligned} e^k_{m,n} &= {}_cD_{0,t}{}^{\alpha} \sum_{i=0}^{N} \sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} x^j_m t^{i-j}_n + c^2 j(j-1) \sum_{i=0}^{N} \sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} x^{j-2}_m t^{i-j}_n - f(x_m, t_n) \\ &= \frac{i\Gamma(i-j+1)}{\Gamma(i-j+1-\alpha)} \sum_{i=0}^{N} \sum_{j=0}^{i-1} \theta^k_{\frac{i(i+1)}{2}+j+1} x^j_m t^{i-j-\alpha}_n \\ &\quad + c^2 j(j-1) \sum_{i=0}^{N} \sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} x^{j-2}_m t^{i-j}_n - f(x_m, t_n). \end{aligned} \tag{3.10}$$

Concerning the initial points, we gain the initial error by substituting the k-th test solution (3.1) into Eq (3.6)

$$e^k_{m,1} = \sum_{i=0}^{N} \theta^k_{\frac{(i+1)(i+2)}{2}} x^i_m - \varphi(x_m). \tag{3.11}$$

With regard to the boundary points, substituting the k-th test solution (3.1) into Eqs (3.8) and (3.9) we acquire the boundary errors

$$e^k_{1,n} = \sum_{i=0}^{N} \theta^k_{\frac{i(i+1)}{2}+1} t^i_n - \lambda(t_n), \tag{3.12}$$

and

$$e^k_{N_m,n} = \sum_{i=0}^{N} \sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} P^j t^{i-j}_n - \mu(t_n). \tag{3.13}$$

Here, we use the gradient descent method to update the k-th unknown weights $\theta^k_{\frac{i(i+1)}{2}+j+1}$. For more details, please refer to [25].

The weight update formula is as follow:

$$\theta^{k+1}_{\frac{i(i+1)}{2}+j+1} = \theta^{k}_{\frac{i(i+1)}{2}+j+1} + \Delta\theta^{k}_{\frac{i(i+1)}{2}+j+1},\ k = 0, 1, 2, \cdots, K, \tag{3.14}$$

with

$$\begin{aligned}
\Delta\theta^{k}_{\frac{i(i+1)}{2}+j+1} &= -\eta\frac{\partial E^k}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&= -\eta\sum_{m=1,n=1}^{N_m,N_n}\frac{\partial Ee^k}{\partial e^k_{m,n}}\frac{\partial e^k_{m,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&= -\eta\sum_{m=2,n=2}^{N_m-1,N_n}\frac{\frac{1}{2}\partial\left\{\sum_{m=2,n=2}^{N_m-1,N_n}\left(e^k_{m,n}\right)^2\right\}}{\partial e^k_{m,n}}\frac{\partial e^k_{m,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&\quad -\eta\sum_{m=1}^{N_m}\frac{\frac{1}{2}\partial\left\{\sum_{m=1}^{N_m}\left(e^k_{m,1}\right)^2\right\}}{\partial e^k_{m,1}}\frac{\partial e^k_{m,1}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&\quad -\eta\sum_{n=1}^{N_n}\frac{\frac{1}{2}\partial\left\{\sum_{n=1}^{N_n}\left(e^k_{1,n}\right)^2\right\}}{\partial e^k_{1,n}}\frac{\partial e^k_{1,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&\quad -\eta\sum_{n=1}^{N_n}\frac{\frac{1}{2}\partial\left\{\sum_{n=1}^{N_n}\left(e^k_{N_m,n}\right)^2\right\}}{\partial e^k_{N_m,n}}\frac{\partial e^k_{N_m,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&= -\eta\sum_{m=2,n=2}^{N_m-1,N_n}(e^k_{m,n})^2\frac{\partial e^k_{m,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} - \eta\sum_{m=1}^{N_m}(e^k_{m,1})^2\frac{\partial e^k_{m,1}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&\quad -\eta\sum_{n=1}^{N_n}(e^k_{1,n})^2\frac{\partial e^k_{1,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} \\
&\quad -\eta\sum_{n=1}^{N_n}(e^k_{N_m,n})^2\frac{\partial e^k_{N_m,n}}{\partial\theta^{k}_{\frac{i(i+1)}{2}+j+1}} - \eta\sum_{m=2,n=2}^{N_m-1,N_n}(e^k_{m,n}) \\
&\quad \times\left\{\frac{i\Gamma(i-j+1)}{\Gamma(i-j+1-\alpha)}\sum_{i=0}^{N}\sum_{j=0}^{i-1}x_m^j t_n^{i-j-\alpha} + a^2 j(j-1)\sum_{i=0}^{N}\sum_{j=0}^{i}x_m^{j-2}t_n^{i-j}\right\} \\
&\quad -\eta\sum_{m=1}^{N_m}(e^k_{m,1})\sum_{i=0}^{N}x_m^i - \eta\sum_{n=1}^{N_n}(e^k_{1,n})\sum_{i=0}^{N}t_n^i - \eta\sum_{n=1}^{N_n}(e^k_{N_m,n})\sum_{i=0}^{N}\sum_{j=0}^{i}P^j t_n^{i-j}.
\end{aligned}$$

Here, $\theta^0_{\frac{i(i+1)}{2}+j+1}$ are some numbers randomly generated by Gaussian distribution, K is an integer representing the maximum number of iterations, and η is the learning rate of neural network.

3.1. The time-fractional diffusion-wave equation (1.2)

For the time-fractional wave equation (1.2), we just need to rewrite Eqs (3.10) and (3.14) into

$$\begin{aligned} e^k_{m,n} &= {}_cD_{0,t}{}^{\alpha} \sum_{i=0}^{N}\sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} x^j_m t^{i-j}_n + a^2 j(j-1) \sum_{i=0}^{N}\sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} x^{j-2}_m t^{i-j}_n - f(x_m, t_n) \\ &= \frac{i(i-1)\Gamma(i-j+1)}{\Gamma(i-j+1-\alpha)} \sum_{i=0}^{N}\sum_{j=0}^{i-2} \theta^k_{\frac{i(i+1)}{2}+j+1} x^j_m t^{i-j-\alpha}_n \\ &\quad + a^2 j(j-1) \sum_{i=0}^{N}\sum_{j=0}^{i} \theta^k_{\frac{i(i+1)}{2}+j+1} x^{j-2}_m t^{i-j}_n - f(x_m, t_n), \end{aligned} \tag{3.15}$$

and

$$\theta^{k+1}_{\frac{i(i+1)}{2}+j+1} = \theta^k_{\frac{i(i+1)}{2}+j+1} + \Delta\theta^k_{\frac{i(i+1)}{2}+j+1},\ k = 0, 1, 2, \cdots, K, \tag{3.16}$$

with

$$\begin{aligned} \Delta\theta^k_{\frac{i(i+1)}{2}+j+1} &= -\eta \frac{\partial E^k}{\partial \theta^k_{\frac{i(i+1)}{2}+j+1}} \\ &= -\eta \sum_{m=2,n=2}^{N_m-1,N_n} (e^k_{m,n}) \left\{ \frac{i(i-1)\Gamma(i-j+1)}{\Gamma(i-j+1-\alpha)} \sum_{i=0}^{N}\sum_{j=0}^{i-2} x^j_m t^{i-j-\alpha}_n \right. \\ &\quad \left. + a^2 j(j-1) \sum_{i=0}^{N}\sum_{j=0}^{i} x^{j-2}_m t^{i-j}_n \right\} - \eta \sum_{m=1}^{N_m} (e^k_{m,1}) \sum_{i=0}^{N} x^i_m \\ &\quad - \eta \sum_{n=1}^{N_n} (e^k_{1,n}) \sum_{i=0}^{N} t^i_n - \eta \sum_{n=1}^{N_n} (e^k_{N_m,n}) \sum_{i=0}^{N}\sum_{j=0}^{i} P^j t^{i-j}_n, \end{aligned}$$

respectively.

4. Adaptive weight adjustment

It is well known that a system of partial differential equations can have infinite solutions without specifying proper bounds or initial conditions [26]. So is the time-fractional diffusion and diffusion-wave equation. Therefore, we believe that the gradient of internal error ($\nabla_{\bar{\theta}} E_{in}$), the gradient of initial error ($\nabla_{\bar{\theta}} E_{ini}$) and the gradient of boundary error ($\nabla_{\bar{\theta}} E_{bou}$) should match each other in magnitude, otherwise the network will eventually tend to learn any solution satisfying the equation due to the dominant effect of the gradient of internal error, and then, it is easy to fall back to the wrong prediction. In this section, let us talk over how to balance the interaction between internal error term (E_{in}), initial error term (E_{ini}) and boundary error term (E_{bou}). For simplicity, $\bar{\theta}$ denotes $\theta_{\frac{i(i+1)}{2}+j+1}$ in this section.

Recall the error function Eq (3.2) and weight update formula Eq (3.14) proposed in Section 3, we get that

$$\bar{\theta}^{k+1} = \bar{\theta}^k - \eta \nabla_{\bar{\theta}^k} E^k$$

$$= \bar{\theta}^k - \eta \left[\frac{1}{2} \nabla_{\bar{\theta}^k} \left\| E_{in}^k \right\|_F^2 + \frac{1}{2} \nabla_{\bar{\theta}^k} \left\| E_{ini}^k \right\|_2^2 + \frac{1}{2} \nabla_{\bar{\theta}^k} \left\| E_{bou}^k \right\|_F^2 \right]. \tag{4.1}$$

Usually, the internal error term dominates the overall. To balance the interaction between different terms in this error, a simple method is to weight the initial error term and the boundary error term. Thus, Eq (3.2) and Eq (3.14) can be rewritten as

$$E^k = \frac{1}{2} \left\| E_{in}^k \right\|_F^2 + \frac{1}{2} \omega_1^k \left\| E_{ini}^k \right\|_2^2 + \frac{1}{2} \omega_2^k \left\| E_{bou}^k \right\|_F^2, \tag{4.2}$$

and

$$\begin{aligned}
\bar{\theta}^{k+1} &= \bar{\theta}^k - \eta \nabla_{\bar{\theta}^k} E^k \\
&= \bar{\theta}^k - \eta [\frac{1}{2} \nabla_{\bar{\theta}^k} \left\| E_{in}^k \right\|_F^2 + \frac{1}{2} \omega_1^k \nabla_{\bar{\theta}^k} \left\| E_{ini}^k \right\|_2^2 + \frac{1}{2} \omega_2^k \nabla_{\bar{\theta}^k} \left\| E_{bou}^k \right\|_F^2] \\
&= -\eta \sum_{m=2,n=2}^{N_m-1,N_n} (e_{m,n}^k) \left\{ \frac{i\Gamma(i-j+1)}{\Gamma(i-j+1-\alpha)} \sum_{i=0}^{N} \sum_{j=0}^{i-1} x_m^j t_n^{i-j-\alpha} \right. \\
&\left. + c^2 j(j-1) \sum_{i=0}^{N} \sum_{j=0}^{i} x_m^{j-2} t_n^{i-j} \right\} - \eta \omega_1^k \sum_{m=1}^{N_m} (e_{m,1}^k) \sum_{i=0}^{N} x_m^i - \eta \omega_2^k \sum_{n=1}^{N_n} (e_{1,n}^k) \sum_{i=0}^{N} t_n^i \\
&- \eta \omega_2^k \sum_{n=1}^{N_n} (e_{N_m,n}^k) \sum_{i=0}^{N} \sum_{j=0}^{i} P^j t_n^{i-j},
\end{aligned} \tag{4.3}$$

respectively, ω_1^k, ω_2^k are the weighted coefficients. This method is similar to the weighted optimization method, which improves the optimization effect by penalizing the initial error term and boundary error term.

For the time-fractional diffusion-wave equation (1.2), Eq (3.16) can be rewritten as

$$\begin{aligned}
\bar{\theta}^{k+1} &= \bar{\theta}^k - \eta \nabla_{\bar{\theta}^k} E^k \\
&= \bar{\theta}^k - \eta [\frac{1}{2} \nabla_{\bar{\theta}^k} \left\| E_{in}^k \right\|_F^2 + \frac{1}{2} \omega_1^k \nabla_{\bar{\theta}^k} \left\| E_{ini}^k \right\|_2^2 + \frac{1}{2} \omega_2^k \nabla_{\bar{\theta}^k} \left\| E_{bou}^k \right\|_F^2] \\
&= -\eta \sum_{m=2,n=2}^{N_m-1,N_n} (e_{m,n}^k) \left\{ \frac{i(i-1)\Gamma(i-j+1)}{\Gamma(i-j+1-\alpha)} \sum_{i=0}^{N} \sum_{j=0}^{i-2} x_m^j t_n^{i-j-\alpha} \right. \\
&\left. + c^2 j(j-1) \sum_{i=0}^{N} \sum_{j=0}^{i} x_m^{j-2} t_n^{i-j} \right\} - \eta \omega_1^k \sum_{m=1}^{N_m} (e_{m,1}^k) \sum_{i=0}^{N} x_m^i \\
&- \eta \omega_2^k \sum_{n=1}^{N_n} (e_{1,n}^k) \sum_{i=0}^{N} t_n^i - \eta \omega_2^k \sum_{n=1}^{N_n} (e_{N_m,n}^k) \sum_{i=0}^{N} \sum_{j=0}^{i} P^j t_n^{i-j}.
\end{aligned} \tag{4.4}$$

Next, we will discuss how to choose the weighted coefficients ω_1^k, ω_2^k. For different models, the optimal constant can be very different, which means that we cannot find a fixed empirical formula that can be transferred between different time-fractional diffusion and diffusion-wave equations. Also, the error function always consists of different parts that are used to constrain the equation. It is impractical to manually assign different weights to different terms of the error function. The traditional parameter adjustment method can neither guarantee the efficiency nor ensure the optimization to the global optimal value. Therefore, inspired by the learning rate annealing algorithm in [27], we propose an adaptive weight coefficients adjustment method.

At the $(k+1)$th iterations, we calculate the estimated values of ω_1^k and ω_2^k by the formula as below:

$$\hat{\omega_1}^{k+1} = \frac{\overline{|\nabla_{\bar{\theta}} E_{in}|}}{\overline{|\nabla_{\bar{\theta}} E_{ini}|}}, \quad \hat{\omega_2}^{k+1} = \frac{\overline{|\nabla_{\bar{\theta}} E_{in}|}}{\overline{|\nabla_{\bar{\theta}} E_{bou}|}}, \tag{4.5}$$

where $\overline{|\nabla_{\bar{\theta}} E_{in}|}$, $\overline{|\nabla_{\bar{\theta}} E_{ini}|}$ and $\overline{|\nabla_{\bar{\theta}} E_{bou}|}$ denote the means of $|\nabla_{\bar{\theta}} E_{in}|$, $|\nabla_{\bar{\theta}} E_{ini}|$ and $|\nabla_{\bar{\theta}} E_{bou}|$.

Then, the weighted coefficients of $(k+1)$ iterations are updated by the weighted moving average method:

$$\omega_1^{k+1} = (1-\xi)\omega_1^k + \xi\hat{\omega_1}^{k+1}, \quad \omega_2^{k+1} = (1-\xi)\omega_2^k + \xi\hat{\omega_2}^{k+1} \tag{4.6}$$

with ξ is given as 0.1.

Finally, the algorithm of this paper is summarized as Algorithm 1.

Algorithm 1 Neural network method based on adaptive weight adjustment

```
1: Initialization: Generate grid points $(x_m, t_n)(m = 1, 2, \cdots, N_m, n = 1, 2, \cdots, N_n)$ on the rectangular area $[0, P] \times [0, Q]$. The weights $\theta^0_{\frac{i(i+1)}{2}+j+1}$ $(i = 0, 1, \cdots, N, j = 0, 1, \cdots, i)$ are randomly generated by Gaussian distribution.
2: while the error function $E_k < \epsilon$ ($\epsilon$ is a small parameter, which is given as $10^{-7}$) or the iteration time $k < K$ do
3:     Compute the error function $E_k, k > K$ ($K$ is maximum number of iterations) and weight variation $\Delta\theta^k_{\frac{i(i+1)}{2}+j+1}$ $(i = 0, 1, \cdots, N, j = 0, 1, \cdots, i)$.
4:     Adjust the weights $\theta^k_{\frac{i(i+1)}{2}+j+1}$ according to the Eq (4.3).
5:     Calculate the weighted coefficients $\omega_1^k, \omega_2^k$ by the Eqs (4.5) and (4.6).
6: end while
7: Save network weights $\theta^k_{\frac{i(i+1)}{2}+j+1}$, calculate neural network solution $u^k(x, t)$ and test other points on the solution area $[0, P] \times [0, Q]$.
```

5. Numerical illustrations

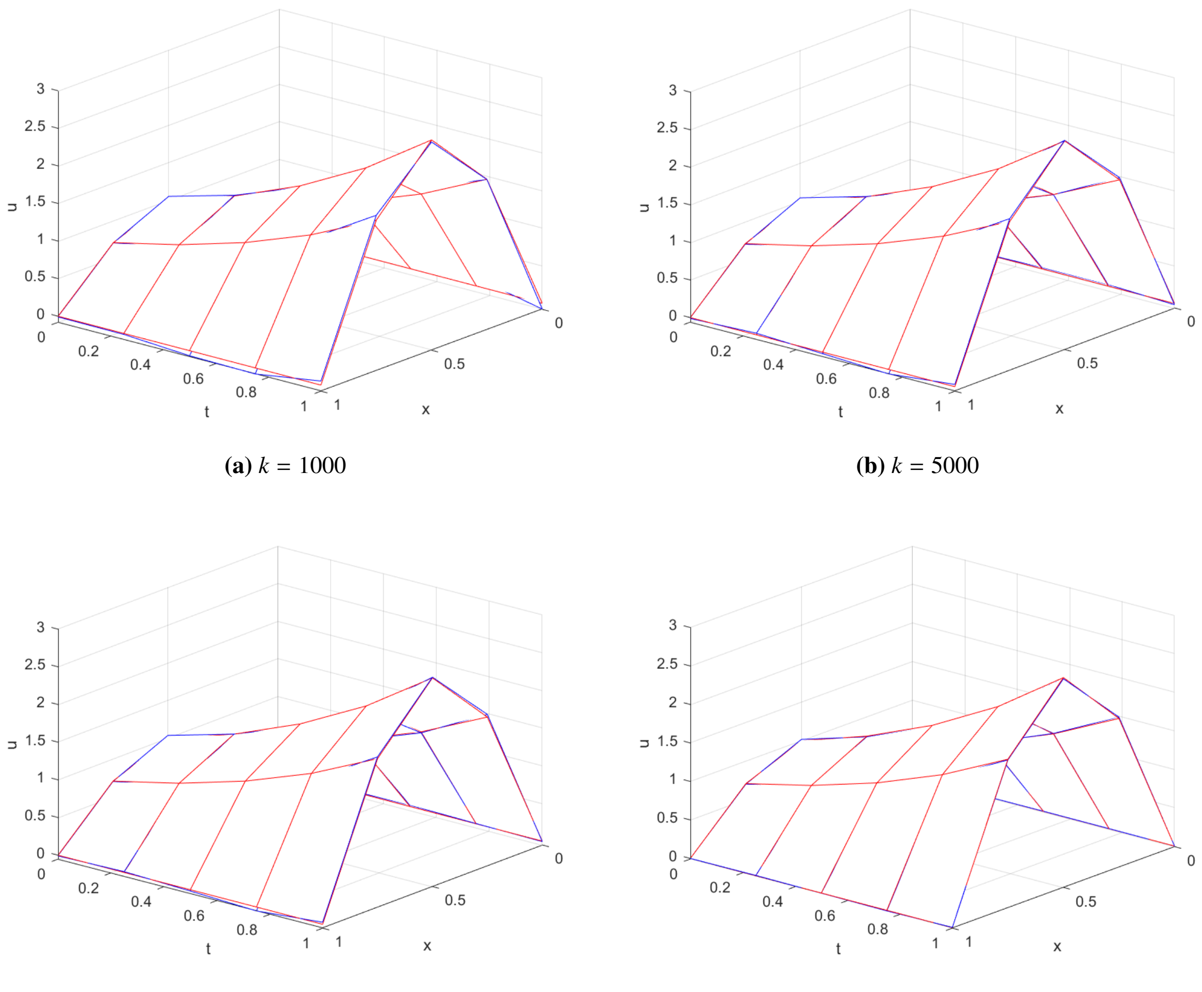
In this section, we give four examples to illustrate the applicability of our proposed neural network method in solving time-fractional diffusion equation (1.1) and time-fractional diffusion-wave equation (1.2). All experiments were performed on a computer with the following configurations: Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz 1.80 GHz using MATLAB R2019a.

5.1. *The time-fractional diffusion equation* (1.1)

Example 1. We consider Eq (1.1) with the conditions: $\lambda(x) = \mu(x) = 0$, $\varphi(x) = sin(\frac{\pi x}{P})$ and $f(x,t) = t^{1-\alpha} sin(\frac{\pi x}{P}) E_{1,2-\alpha}(t) - (\frac{c\pi}{P})^2 exp(t) sin(\frac{\pi x}{P})$. The exact solution of the equation may be calculated by means of Definition 2.4 and Lemma 2.1.

$$u(x,t) = sin(\frac{\pi x}{P}) exp(t).$$

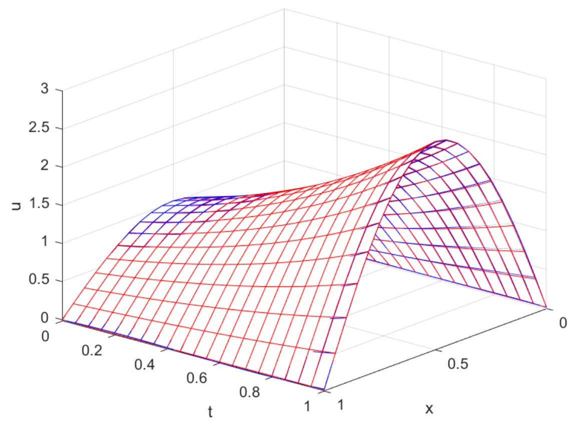
At first, the parameters of the model are set to the following: $P = 1$, $Q = 1$, $c = 1$ and $0 < \alpha < 1$. Then, the parameters of the proposed method we use are as follows: $N = 5$, $m = 5$, $n = 5$, learning rate $\mu = 4 \times 10^{-4}$, and accuracy threshold $\epsilon = 1 \times 10^{-7}$. When $\alpha = 0.9$, Figure 2 represent the comparisons between the exact solution and the neural network solution, and the number of iterations is 2000, 5000, 10000, 20000, respectively. We observe that the approximate solution gets better and better as the number of iterations increases. Then, we check the approximate solution on the grid of $N_m \times N_n = 21 \times 21$ in Figure 3a. The variation of the error value with the number of iterations is shown in Figure 4. Also, in Figure 3, a comparison of the exact and the approximate solution is given under $\alpha = 0.3, 0.5$ and 0.7. In addition, we list the numerical solutions and absolute errors of some specific test points for $\alpha = 0.3, 0.5, 0.7$ and 0.9 in Table 1. We can see that we get similar results for different fractional order α. Eventually, the weights of the neural network are listed in Table 2.



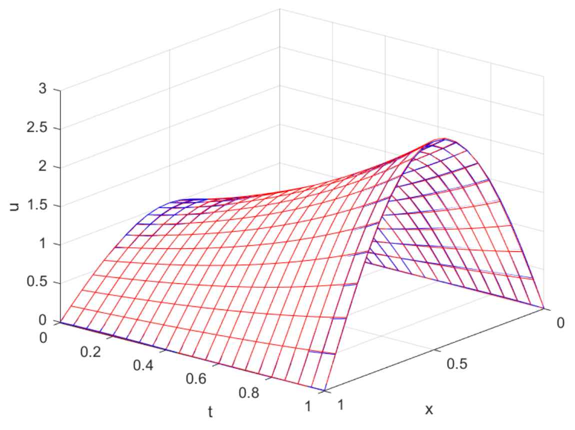
(a) $k = 1000$ **(b)** $k = 5000$

(c) $k = 10000$ **(d)** $k = 20000$

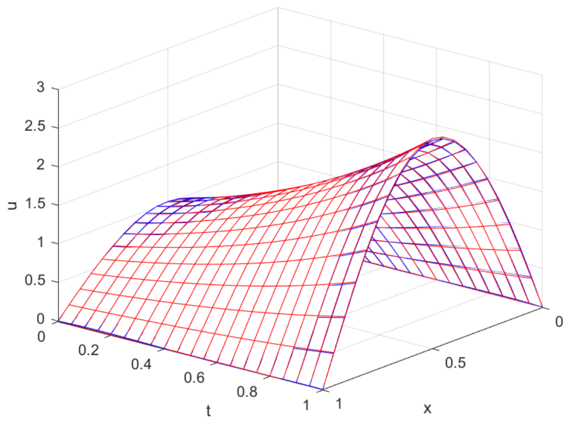
Figure 2. The exact solution and the neural network solution for Example 1 at $\alpha = 0.9$. Red: exact solution, blue: neural network solution.



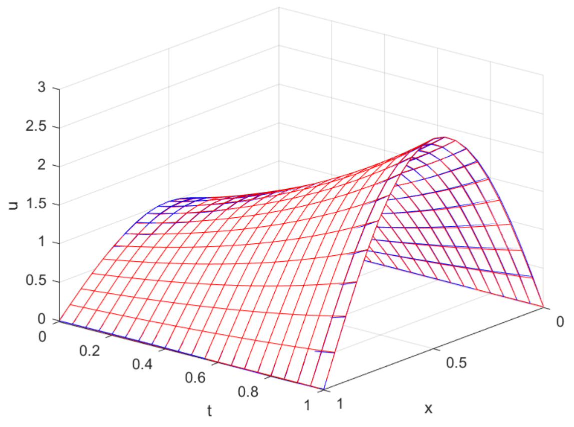
(a) $\alpha = 0.9$



(b) $\alpha = 0.3$



(c) $\alpha = 0.5$



(d) $\alpha = 0.7$

Figure 3. The exact solution and the neural network solution on the grid of $N_m \times N_n = 21 \times 21$ for Example 1 at $\alpha = 0.9$ with $k = 20000$. Red: exact solution, blue: neural network solution.

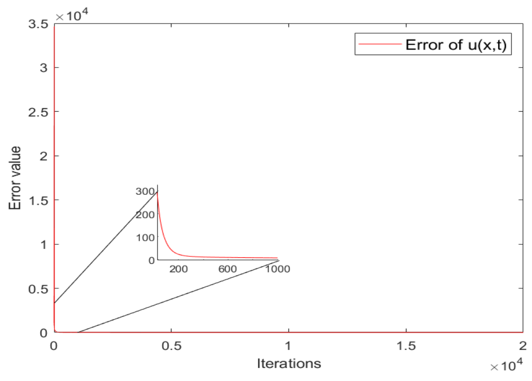


Figure 4. Error decreasing trend with regard to iterations for fractional order $\alpha = 0.9$ and maximum training time $K = 20000$.

Table 1. The numerical solutions and absolute errors for Example 1 at $\alpha = 0.3, 0.5, 0.7$ and 0.9 when $k = 20000$.

α / (x,t)	0.3		0.5		0.7		0.9	
	solution	error	solution	error	solution	error	solution	error
$(0,0)$	0.0029	0.0029	0.0046	0.0046	0.0031	0.0031	0.0030	0.0030
$(0.2,0.2)$	0.6996	0.0184	0.7029	0.0150	0.7034	0.0145	0.7003	0.0176
$(0.4,0.4)$	1.4004	0.0184	1.4026	0.0162	1.4084	0.0104	1.4042	0.0147
$(0.6,0.6)$	1.7059	0.0270	1.7076	0.0254	1.7087	0.0243	1.7054	0.0275
$(0.8,0.8)$	1.2730	0.0351	1.2781	0.0301	1.2695	0.0386	1.2694	0.0387
$(1,1)$	0.0025	0.0025	0.0014	0.0014	0.0024	0.0024	0.0032	0.0032

Table 2. The weights of the proposed neural network method for Example 1 at $\alpha = 0.3, 0.5, 0.7$ and 0.9 when $k = 20000$.

Fractional order	$\theta^k_{\frac{i(i+1)}{2}+j+1}$, $k = 20000$						
0.3	0.0029	–0.0344	2.9583	–0.0048	3.4271	0.3197	0.0755
	2.1478	–1.7237	–3.7582	0.1448	0.4967	–1.4288	–4.5874
	–2.9112	–0.1865	0.1164	–0.4064	–1.0538	3.0193	3.3892
0.5	0.0046	0.0168	2.9111	–0.2540	3.5760	0.5170	0.3759
	2.0230	–2.0247	–4.1744	0.1251	0.5603	–1.1722	–4.4873
	–2.4350	–0.2712	0.0068	–0.2784	–1.2850	3.0901	3.1768
0.7	0.0031	–0.0509	2.9372	0.2153	3.5563	0.4696	–0.2334
	1.6944	–1.8766	–4.1846	–0.0421	0.8784	–0.7943	–4.6371
	–2.4028	0.1078	0.0842	–0.8617	–1.0688	3.0298	3.1788
0.9	0.0030	–0.0381	2.9011	0.1165	3.5459	0.6432	–0.0392
	1.8245	–2.1044	–4.4339	–0.1421	0.8739	–0.9409	–4.2895
	–2.2707	0.0989	–0.0227	–0.6420	–1.1731	2.9353	3.1575

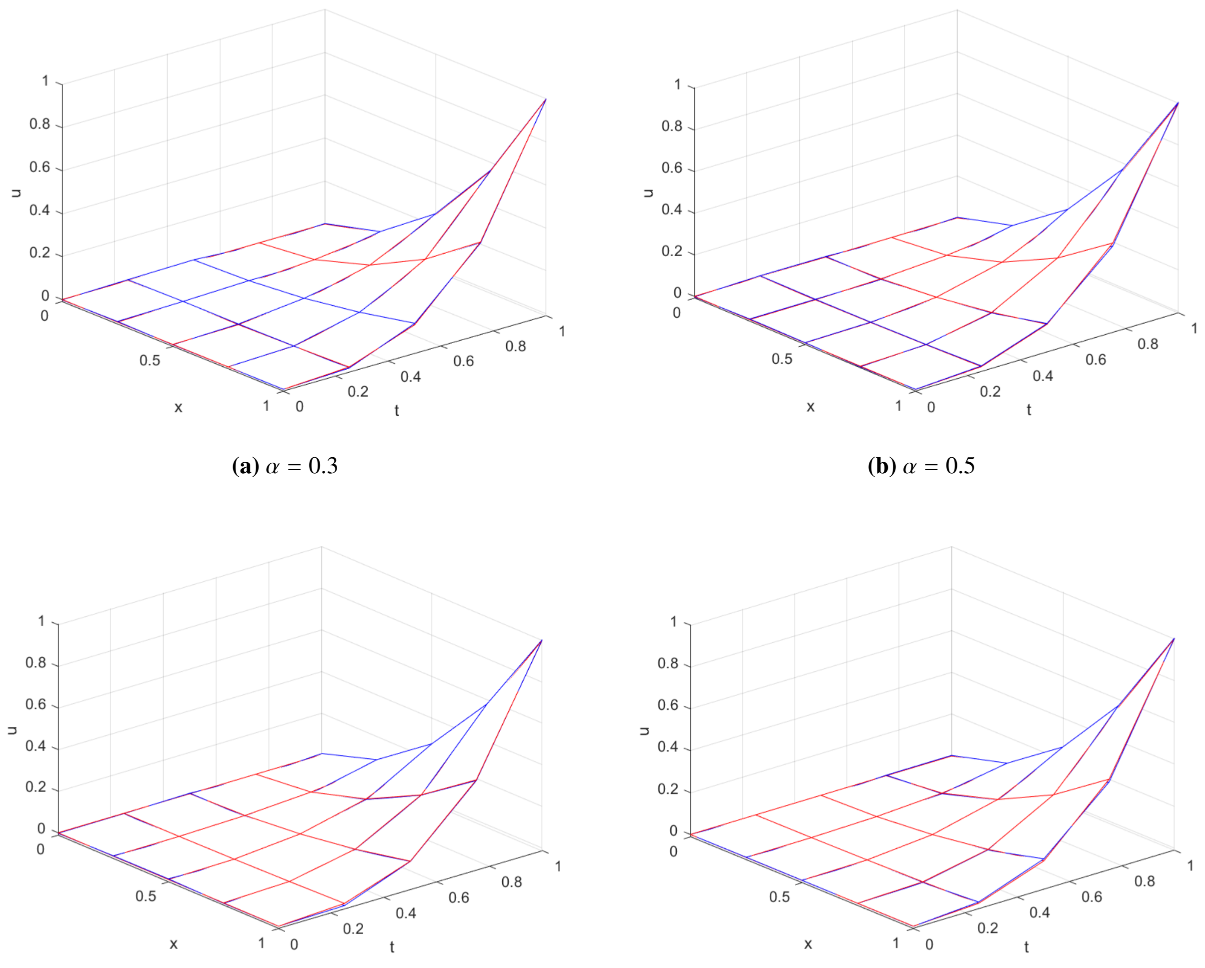
Example 2. In this example, we consider Eq (1.1) with the conditions as follows:

$$\begin{cases} \lambda(t) = 0, \mu(t) = P^2 t^3, \\ \varphi(x) = 0, \\ f(x,t) = \frac{\Gamma(4)}{\Gamma(4-\alpha)} t^{3-\alpha} x^2 + 2c^2 t^3, \end{cases}$$

in this situation, the exact solution is

$$u(x,t) = x^2 t^3.$$

Let $P = 1$, $Q = 1$, $c = 1$ and $0 < \alpha < 1$. In our proposed neural network method, the parameters are as follows: $N = 7$, $N_m = 5$, $N_n = 5$. We set the learning rate η and accuracy threshold ϵ to 1×10^{-4} and 1×10^{-7}, respectively. Through 20000 iterations, we obtain the neural network solution of Example 2 for different fractional orders $\alpha = 0.3, 0.5, 0.7$ and 0.9. Figure 5 give a comparison of neural network solutions and exact solutions for different fractional orders. In Table 3, we list the numerical solutions and absolute errors of several test points under different α. Next, we subdivided the computational grid into $N_m \times N_n = 21 \times 21$ and tested our neural network solution on Figure 6. In Figure 7, we plot the variation of the error for different fractional order. Finally, the weights of neurons after iteration are listed in Table 4.



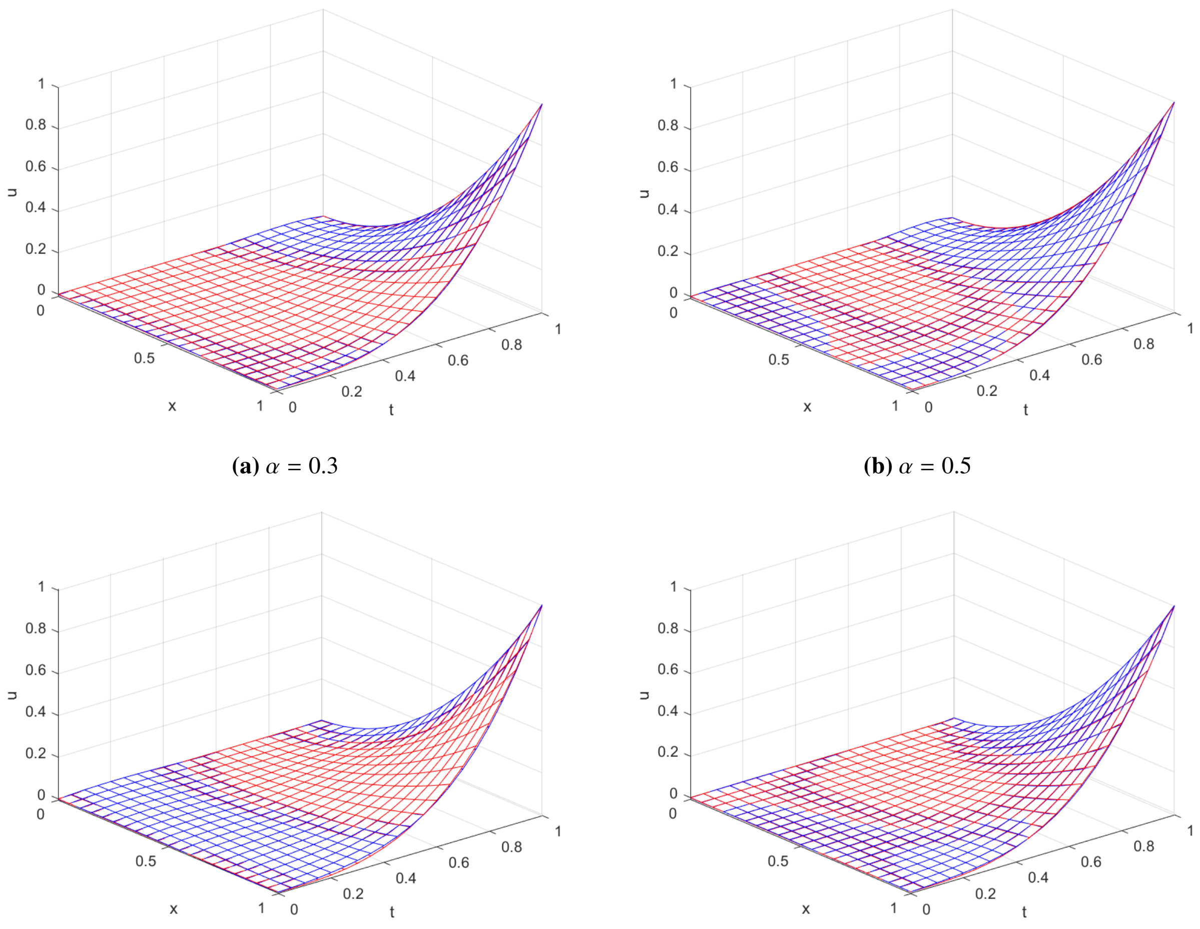
(a) $\alpha = 0.3$

(b) $\alpha = 0.5$

(c) $\alpha = 0.7$

(d) $\alpha = 0.9$

Figure 5. The exact solution and the approximate solution for Example 2 with $k = 20000$. Red: exact solution, blue: neural network solution.



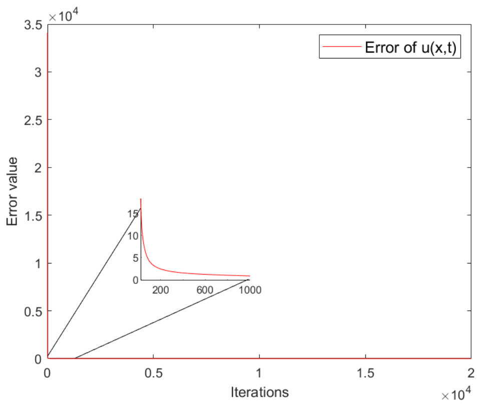
(a) $\alpha = 0.3$ **(b)** $\alpha = 0.5$

(c) $\alpha = 0.7$ **(d)** $\alpha = 0.9$

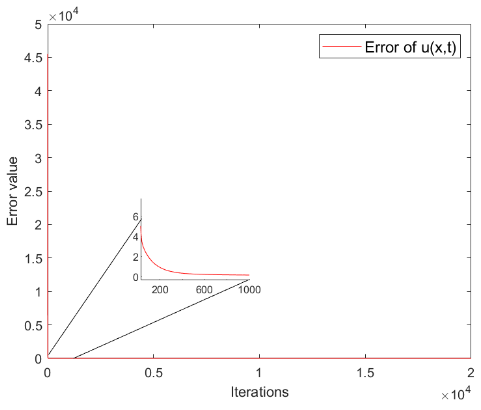
Figure 6. The exact solution and the neural network solution on the gird of $N_m \times N_n = 21 \times 21$ for Example 2 with $k = 20000$. Red: exact solution, blue: neural network solution.

Table 3. The numerical solutions and absolute errors for Example 2 at $\alpha = 0.3, 0.5, 0.7$ and 0.9 when $k = 20000$.

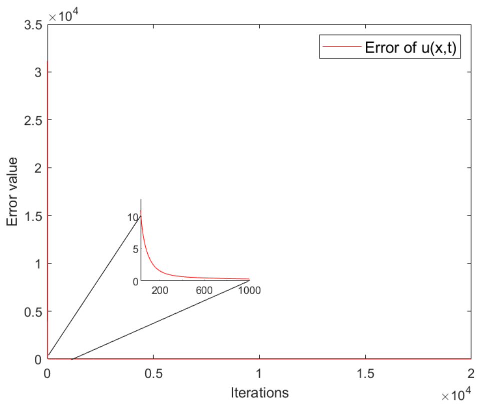
α	0.3		0.5		0.7		0.9	
(x, t)	solution	error	solution	error	solution	error	solution	error
$(0, 0)$	–0.0016	0.0016	–0.0025	0.0025	–0.0023	0.0023	–0.0001	0.0001
$(0.2, 0.2)$	0.0061	0.0058	0.0035	0.0032	–0.0064	0.0067	–0.0076	0.0080
$(0.4, 0.4)$	0.0188	0.0085	0.0088	0.0014	0.0014	0.0088	0.0001	0.0101
$(0.6, 0.6)$	0.0816	0.0039	0.0656	0.0122	0.0740	0.0037	0.0705	0.0073
$(0.8, 0.8)$	0.3148	0.0128	0.3071	0.0206	0.3260	0.0017	0.3144	0.0133
$(1, 1)$	1.0011	0.0011	1.0055	0.0055	1.0020	0.0020	1.0026	0.0026



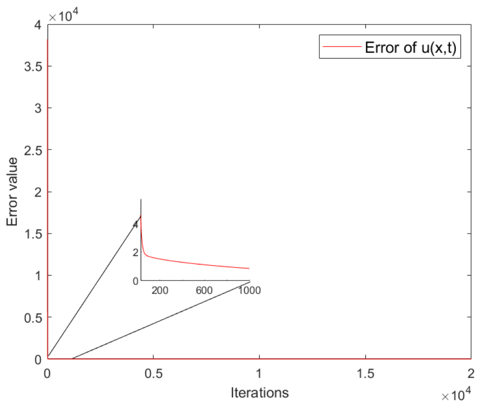
(a) $\alpha = 0.3$



(b) $\alpha = 0.5$



(c) $\alpha = 0.7$



(d) $\alpha = 0.9$

Figure 7. Error decreasing trend with regard to iterations where maximum training time $K = 20000$.

Table 4. The weights of the proposed neural network method for Example 2 at $\alpha = 0.3, 0.5, 0.7$ and 0.9 when $k = 20000$.

Fractional order	$\theta^k_{\frac{i(i+1)}{2}+j+1}$, $k = 20000$											
	–0.0016	–0.0002	0.0571	0.1500	–0.0567	–0.2120	–0.2402	0.0174	0.0459	0.1907	0.2427	–0.2256
0.3	0.4118	0.1506	0.1091	–0.4905	0.1521	0.3623	0.3238	–0.5528	–0.3150	0.0692	0.2436	0.3069
	–0.0435	–0.1723	0.0865	0.3350	0.2728	–0.1388	–0.1348	–0.1450	0.2798	–0.1127	0.1988	–0.1630
	–0.0025	0.0238	0.0540	0.0662	–0.2532	–0.1620	–0.1016	0.1590	0.4392	0.1513	–0.3655	–0.0698
0.5	–0.1684	–0.0452	0.0068	0.3707	0.2070	0.1868	0.5117	–0.4333	0.0094	0.1612	0.1791	0.2668
	0.1049	–0.2033	0.3859	–0.2370	–0.1491	–0.2049	0.2384	–0.3002	–0.0007	0.1888	–0.1895	0.1808
	–0.0023	–0.0265	0.0230	0.1398	–0.2252	–0.0691	–0.0352	0.0130	–0.0024	0.2792	–0.1484	0.3733
0.7	0.4558	0.2513	–0.4968	–0.1691	0.1646	0.0865	–0.1089	0.0741	0.1482	0.0129	0.1104	–0.1779
	0.0332	0.0814	–0.3102	0.2354	0.2346	–0.3240	0.4362	–0.2617	0.4132	–0.2044	0.1158	–0.1176
	–0.0001	–0.0464	0.0058	0.0056	–0.0356	0.0137	0.0867	–0.1687	0.2562	–0.1507	–0.0945	0.3658
0.9	0.4054	–0.3842	0.3073	0.2090	–0.2674	–0.0495	0.5397	0.0868	–0.1517	–0.0252	–0.2444	0.1457
	–0.1503	–0.0782	–0.1682	–0.1482	–0.1317	0.4388	–0.0699	0.4540	–0.1167	–0.2030	0.2436	0.1234

Example 3. Let $P = 1$, $Q = 1$, $c = 1$ and $1 < \alpha < 2$, we consider Eq (1.2) with $\lambda(t) = 0$, $\mu(t) = P^2t^3$, $\varphi(x) = 0$, $\psi(x) = 0$ and $f(x,t) = \frac{\Gamma(4)}{\Gamma(4-\alpha)}t^{3-\alpha}x^2 + 2c^2t^3$. Through calculation, we can gain the exact solution $u(x,t) = x^2t^3$.

In our proposrd method, we set $N = 7$ and sample gird $N_m \times N_n = 5 \times 5$. The learning rate η and accuracy threshold ϵ are set to 1×10^{-4} and 1×10^{-7}, respectively. With 20000 iterations, we acquire a good neural network solution. Figure 8 show a comparison of the neural network solution and the exact solution on sampling grid, their comparison on denser grid $N_m \times N_n = 21 \times 21$ and the change of the error value with the number of iterations, respectively. In Table 5, we list the numerical solutions and absolute errors of several test points for $\alpha = 1.9$. Also, Table 6 lists the weights of the proposed neural network in this example. The results illustrate that the proposed neural network method is also very effective for seeking the solution of the time-fractional diffusion-wave equations. (On account of too many graphs, we only list the situation of $\alpha = 1.9$ here.)

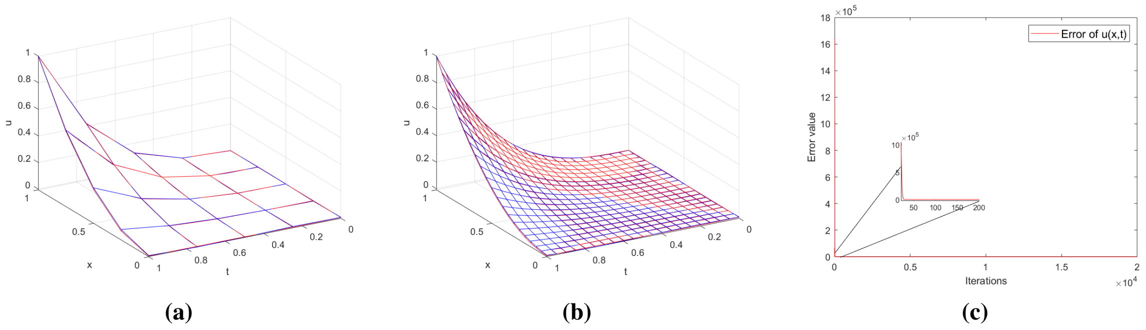


Figure 8. The exact solution and the neural network solution for Example 3 at $\alpha = 1.9$ with $k = 20000$. (a) Comparison of the neural network solution and the exact solution on sampling grid. (b) comparison on denser grid $N_m \times N_n = 21 \times 21$. (c) the change of the error value with the number of iterations. Red: exact solution, blue: neural network solution.

Table 5. The numerical solutions and absolute errors for Example 3 at $\alpha = 1.9$ when $k = 20000$.

Test point	Numerical solution	Absolute error	Test point	Numerical solution	Absolute error
$(0, 0)$	0.0015	0.0015	$(0.5, 0.5)$	0.0317	0.0004
$(0.1, 0.1)$	0.0013	0.0013	$(0.6, 0.6)$	0.0773	0.0004
$(0.2, 0.2)$	0.0011	0.0008	$(0.7, 0.7)$	0.1651	0.0030
$(0.3, 0.3)$	0.0028	0.0004	$(0.8, 0.8)$	0.3209	0.0068
$(0.4, 0.4)$	0.0106	0.0004	$(1, 1)$	1.0015	0.0015

Table 6. The weights of the proposed neural network method for Example 3 at $\alpha = 1.9$ with $k = 20000$.

Fractional order	$\theta^k_{\frac{i(i+1)}{2}+j+1}$, $k = 20000$					
	0.0015	0.0231	–0.0217	–0.0077	0.01778	–0.0464
	–0.2435	0.4708	–0.4991	0.2667	0.3001	–0.3739
1.9	0.0398	0.5381	–0.0875	0.0560	0.1424	0.2924
	0.3189	–0.3403	–0.1458	–0.2037	–0.0313	0.4367
	0.2250	0.0753	–0.1526	–0.2626	0.07791	–0.0908
	0.2382	–0.4721	0.3075	–0.5056	0.3633	0.2947

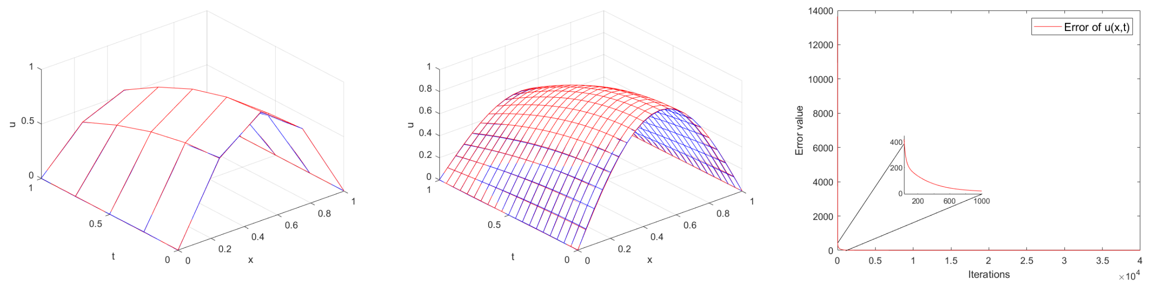
Example 4. In this example, we consider Eq (1.2) with the conditions as follows:

$$\begin{cases} \lambda(t) = \mu(t) = 0, \\ \varphi(x) = sin(\frac{\pi x}{P}), \\ f(x,t) = \sin\left(\frac{\pi x}{P}\right)\cos_{1,1-\alpha}(t) - \left(\frac{c\pi}{P}\right)^2 \sin\left(\frac{\pi x}{P}\right)\cos(t), \end{cases}$$

in this situation, the exact solution is

$$u(x,t) = sin(\frac{\pi x}{P})cos(t).$$

Let $P = 1$, $Q = 1$, $c = 1$ and $1 < \alpha < 2$, other parameters are listed as follows: $N = 5$, $N_m = N_n = 5$, $\eta = 6 \times 10^{-5}$ and $\epsilon = 1 \times 10^{-7}$. Similar to Example 3, we only show the case of $\alpha = 1.5$ by 40000 iterations. In Figure 9, we show the comparison of the neural network solution and the exact solution on sampling grid $N_m \times N_n = 5 \times 5$, their comparison on denser grid $N_m \times N_n = 21 \times 21$ and the change of error value with the number of iterations, respectively. In addition, we list the numerical solutions and absolute errors of several test points in Table 7. The weights of the neural network in this example are listed in Table 8.



(a) (b) (c)

Figure 9. The exact solution and the neural network solution for Example 4 at $\alpha = 1.5$ with $k = 40000$. (a)Comparison of the neural network solution and the exact solution on sampling grid. (b)comparison on denser grid $N_m \times N_n = 21 \times 21$. (c) the change of the error value with the number of iterations. Red: exact solution, blue: neural network solution.

Table 7. The numerical solutions and absolute errors for Example 4 at $\alpha = 1.5$ when $k = 40000$.

Test point	Numerical solution	Absolute error	Test point	Numerical solution	Absolute error
$(0, 0)$	-0.0028	0.0028	$(0.5, 0.5)$	0.8497	0.0279
$(0.1, 0.1)$	0.3185	0.0110	$(0.6, 0.6)$	0.7633	0.0216
$(0.2, 0.2)$	0.5758	0.0002	$(0.7, 0.7)$	0.6026	0.0161
$(0.3, 0.3)$	0.7558	0.0171	$(0.8, 0.8)$	0.3929	0.0166
$(0.4, 0.4)$	0.8484	0.0276	$(1, 1)$	-0.0011	0.0011

Table 8. The weights of the proposed neural network method for Example 4 at $\alpha = 1.5$ with $k = 40000$.

Fractional order	$\theta^k_{\frac{i(i+1)}{2}+j+1}$, $k = 40000$						
	-0.0028	0.0963	3.3865	-0.2123	-0.8028	-1.4247	0.0071
1.5	-0.2903	0.3029	-2.5919	0.1321	-0.3393	0.4284	0.6957
	-1.0843	-0.0163	-0.1139	0.2444	0.2087	-0.3435	1.7188

6. Conclusion

In this paper, different power terms of Taylor polynomial are used as neurons of the proposed artificial neural network to solve the time-fractional diffusion and diffusion-wave equations. The initial weight of the neural network is given randomly according to the Gaussian distribution, and what is more the weight in each iteration is updated by the error function. In order to balance the contribution of initial error term, boundary error term and internal error term in the error function, we weight different error terms, and give an adaptive weight adjustment algorithm. The trained network is found to be effective by testing other points within the solution region. In addition, the value of the error function decreases as the number of iterations increases. This illustrates that our method is very effective for solving time-fractional diffusion and diffusion-wave equations. In the future work, we will consider to extend our methods for solving other time-fractional partial differential equations, such as the time-fractional mixed diffusion and diffusion-wave equations [28], time-fractional diffusion equations with a time-invariant type variable order [29] and the generalized time space fractional diffusion equations with variable coefficients [30].

Acknowledgments

The authors express their sincere thanks to the anonymous reviewer for his/her careful reading of the paper, giving valuable comments and suggestions. It is their contributions that greatly improve the paper. The authors also thank the editors for their kind help.

This work was supported by the National Natural Science Foundation of China (Nos. 11801452, 11701456); Shaanxi Fundamental Science Research Project for Mathematics and Physics (Grant No. 22JSQ020); Fundamental Research Project of Natural Science in Shaanxi Province General Project (Youth) (Nos.2019JQ-415, 2019JQ-196); Chinese Universities Scientific Funds (Nos. 2452022104, 2452022105, 2452022372); and Innovation and Entrepreneurship Training Program for College Students of Northwest A&F University (202110712186, 202210712048, S202210712377).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. Y. A. Rossikhin, M. V. Shitikova, Applications of fractional calculus to dynamic problems of linear and nonlinear hereditary mechanics of solids, *Appl. Mech. Rev.*, **50** (1997), 15–67. https://doi.org/10.1115/1.3101682
2. D. del-Castillo-Negrete, B. A. Carreras, V. E. Lynch, Front dynamics in reaction-diffusion systems with Lévy flights: a fractional diffusion approach, *Phys. Rev. Lett.*, **91** (2003), 018302. https://doi.org/10.1103/PhysRevLett.91.018302
3. A. Dechant, E. Lutz, Anomalous spatial diffusion and multifractality in optical lattices, *Phys. Rev. Lett.*, **108** (2012), 230601. https://doi.org/10.1103/PhysRevLett.108.230601
4. M. Giona, H. E. Roman, Fractional diffusion equation for transport phenomena in random media, *Phys. A*, **185** (1992), 87–97. https://doi.org/10.1016/0378-4371(92)90441-R
5. F. Mainardi, Fractional diffusive waves in viscoelastic solids, In: J. I. Wegner, F. R. Norwood, eds.,*Nonlinear Waves in Solids.*, Fairfield: ASME/AMR, 1995, 93–97.
6. W. H. Luo, T. Z. Huang, G. C. Wu, X. M. Gu, Quadratic spline collocation method for the time fractional subdiffusion equation, *Appl. Math. Comput.*, **276** (2016), 252–265. https://doi.org/10.1016/j.amc.2015.12.020
7. W. H. Luo, C. Li, T. Z. Huang, X. M. Gu, G. C. Wu, A high-order accurate numerical scheme for the Caputo derivative with applications to fractional diffusion problems, *Numer. Funct. Anal. Optim.*, **39** (2018), 600–622. https://doi.org/10.1080/01630563.2017.1402346
8. X. M. Gu, S. L. Wu, A parallel-in-time iterative algorithm for Volterra partial integro-differential problems with weakly singular kernel, *J. Comput. Phys.*, **417** (2020), 109576. https://doi.org/10.1016/j.jcp.2020.109576
9. Y. L. Zhao, X. M. Gu, A. Ostermann, A preconditioning technique for an all-at-once system from Volterra subdiffusion equations with graded time steps, *J. Sci. Comput.*, **88** (2021), 11. https://doi.org/10.1007/s10915-021-01527-7
10. Z. Liu, A. Cheng, X. Li, A novel finite difference discrete scheme for the time fractional diffusion-wave equation, *Appl. Numer. Math.*, **134** (2018), 17–30. https://doi.org/10.1016/j.apnum.2018.07.001

11. R. Du, Y. Yan, Z. Liang, A high-order scheme to approximate the Caputo fractional derivative and its application to solve the fractional diffusion wave equation, *J. Comput. Phys.*, **376** (2019), 1312–1330. https://doi.org/10.1016/j.jcp.2018.10.011

12. X. Li, S. Li, A fast element-free Galerkin method for the fractional diffusion-wave equation, *Appl. Math. Lett.*, **122** (2021), 107529. https://doi.org/10.1016/j.aml.2021.107529

13. M. H. Heydari, M. R. Hooshmandasl, F. M. M. Ghaini, C. Cattani, Wavelets method for the time fractional diffusion-wave equation, *Phys. Lett. A*, **379** (2015), 71–76. https://doi.org/10.1016/j.physleta.2014.11.012

14. M. H. Heydari, Z. Avazzadeh, M. F. Haromi, A wavelet approach for solving multi-term variable-order time fractional diffusion-wave equation, *Appl. Math. Comput.*, **341** (2019), 215–228. https://doi.org/10.1016/j.amc.2018.08.034

15. A. Kumar, A. Bhardwaj, B. V. R. Kumar, A meshless local collocation method for time fractional diffusion wave equation, *Comput. Math. Appl.*, **78** (2019), 1851–1861. https://doi.org/10.1016/j.camwa.2019.03.027

16. H. Qu, Cosine radial basis function neural networks for solving fractional differential equations, *Adv. Appl. Math. Mech.*, **9** (2017), 667–679. https://doi.org/10.4208/aamm.2015.m909

17. F. Rostami, A. Jafarian, A new artificial neural network structure for solving high-order linear fractional differential equations, *Int. J. Comput. Math.*, **95** (2018), 528–539. https://doi.org/10.1080/00207160.2017.1291932

18. F. B. Rizaner, A. Rizaner, Approximate solutions of initial value problems for ordinary differential equations using radial basis function networks, *Neural Process. Lett.*, **48** (2018), 1063–1071. https://doi.org/10.1007/s11063-017-9761-9

19. A. Jafarian, S. M. Nia, A. K. Golmankhaneh, B. Baleanu, On artificial neural networks approach with new cost functions, *Appl. Math. Comput.*, **339** (2018), 546–555. https://doi.org/10.1016/j.amc.2018.07.053

20. A. H. Hadian-Rasanan, D. Rahmati, S. Gorgin, K. Parand, A single layer fractional orthogonal neural network for solving various types of Lane-Emden equation, *New Astron.*, **75** (2020), 101307. https://doi.org/10.1016/j.newast.2019.101307

21. H. Qu, X. Liu, Z. She, Neural network method for fractional-order partial differential equations, *Neurocomputing*, **414** (2020), 225–237. https://doi.org/10.1016/j.neucom.2020.07.063

22. Y. Ye, H. Fan, Y. Li, X. Liu, H. Zhang, Deep neural network methods for solving forward and inverse problems of time fractional diffusion equations with conformable derivative, *Neurocomputing*, **509** (2022), 177–192. https://doi.org/10.1016/j.neucom.2022.08.030

23. A. A. Kilbas, H. M. Srivastava, J. J. Trujillo, *Theory and Applications of Fractional Differential Equations*, New York: Elsevier, 2006.

24. R. Garrappa, The Mittag-Leffler function, MATLAB Central File Exchange. Available from: https://www.mathworks.com/matlabcentral/fileexchange/48154-the-mittag-leffler-function

25. M. H. Hassoun, *Fundamentals of artificial neural networks*, Cambridge: MIT Press, 1995.

26. L. C. Evans, *Partial Differential Equations*, 2^{nd} edition, Providence: American Mathematical Society, 2010.

27. S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.*, **43** (2021), A3055–A3081. https://doi.org/10.1137/20M1318043

28. J. Shen, X. M. Gu, Two finite difference methods based on an H2N2 interpolation for two-dimensional time fractional mixed diffusion and diffusion-wave equations, *Discrete Contin. Dyn. Syst. Ser. B*, **27** (2022), 1179–1207. https://doi.org/10.3934/dcdsb.2021086

29. X. M. Gu, H. W. Sun, Y. L. Zhao, X. Zheng, An implicit difference scheme for time-fractional diffusion equations with a time-invariant type variable order, *Appl. Math. Lett.*, **120** (2021), 107270. https://doi.org/10.1016/j.aml.2021.107270

30. X. M. Gu, T. Z. Huang, Y. L. Zhao, P. Lyu, B. Carpentieri, A fast implicit difference scheme for solving the generalized time-space fractional diffusion equations with variable coefficients, *Numer Methods Partial Differ Equ*, **37** (2021), 1136–1162. https://doi.org/10.1002/num.22571