

Deep neural network methods for solving forward and inverse problems of time fractional diffusion equations with conformable derivative

Yinlin Ye^a, Yajing Li^{a,*}, Hongtao Fan^a, Xinyi Liu^a, Hongbing Zhang^a

^aCollege of Science, Northwest A&F University, Yangling, Shaanxi 712100, China

Abstract

Physics-informed neural networks (PINNs) show great advantages in solving partial differential equations. In this paper, we for the first time propose to study conformable time fractional diffusion equations by using PINNs. By solving the supervise learning task, we design a new spatio-temporal function approximator with high data efficiency. L-BFGS algorithm is used to optimize our loss function, and back propagation algorithm is used to update our parameters to give our numerical solutions. For the forward problem, we can take IC/BCs as the data, and use PINN to solve the corresponding partial differential equation. Three numerical examples are carried out to demonstrate the effectiveness of our methods. In particular, when the order of the conformable fractional derivative α tends to 1, a class of weighted PINNs is introduced to overcome the accuracy degradation caused by the singularity of solutions. For the inverse problem, we use the data obtained to train the neural network, and the estimation of parameter λ in the equation is elaborated. Similarly, we give three numerical examples to show that our method can accurately identify the parameters, even if the training data is corrupted with 1% uncorrelated noise.

Keywords: conformable time fractional derivative; fractional diffusion; PINNs; weighted PINNs

1. Introduction

Fractional diffusion equations are often used to simulate complex phenomena in many fields, such as mechanics of wake-up or creep in polymer systems [1], kinematics in viscoelastic media [2], solute transport in porous media with fractured geometry [3], etc. The superior modeling ability of fractional diffusion equation has aroused great interest in numerically solving such problems. Many excellent numerical methods are produced. Such methods include finite difference method[5], finite element method[4], spectral method[6] and so on. Although these numerical algorithms have high accuracy, most of them are time-consuming in grid generation, and can not perfectly integrate the actual data into the existing algorithms.

*Corresponding author
Email address: hliyajing@163.com (Yajing Li)

In the past few years, machine learning algorithms have developed rapidly, especially neural network methods. The results of using neural networks to solve partial differential equations have sprung up. For example, Deep Ritz method [7] uses an energy minimization formulation as the loss function of artificial neural networks to solve Partial differential equations. Raissi et al. [8] introduced a physical information neural network (PINN) that is trained to solve supervised learning tasks about any given physical laws described by general nonlinear partial differential equations. When solving the forward and inverse problems of partial differential equations, they put forward two methods of continuous form and discrete form, which obtained good solving accuracy, and the speed of this method was the same when solving the forward and inverse problems of partial differential equations. Subsequently, in [9], they adopted PINNs to directly encode the governing equations into the deep neural network via automatic differentiation, so as to overcome some limitations for simulating incompressible laminar and turbulent flows. Sheng and Yang in [10] proposed a penalty-free neural network (PFNN) method, which can effectively solve a class of second-order boundary-value problems on complex geometries. Numerical experiments showed that PFNN was superior to several existing methods in accuracy, flexibility and robustness. Recently, more and more scholars began to pay attention to using neural network methods to solve fractional partial differential equations, and relevant literatures have emerged one after another. Pang et al. [11] extended PINNs to fractional PINNs (fPINNs) to solve space-time fractional advection-diffusion equations, and systematically studied their convergence. A novel element of the fPINNs was the hybrid approach that they introduced for constructing the residual in the loss function using both automatic differentiation for the integer-order operators and numerical discretization for the fractional operators. Then Pang et al. in [12] extended PINNs to parameter and function inference for integral equations such as nonlocal Poisson and nonlocal turbulence models, and referred to them as nonlocal PINNs (nPINNs). Qu et al. [13] proposed a neural network method based on Legendre polynomials to solve space and time fractional diffusion equations.

Over the past few decades, various definitions of fractional derivative including Grunwald-Letnikov fractional derivatives, Riemann-Liouville fractional derivatives, Caputo fractional derivatives, Riesz fractional derivatives, etc, have been reported in many literatures. These fractional derivatives do not satisfy the chain rule. In 2014, Khalil et al.[14] introduced a new fractional derivative, which performed well and followed the Leibniz rule and the chain rule, called the conformable derivative. Due to its effectiveness and applicability, conformable derivatives have been applied to Newtonian mechanics [15], quantum mechanics [16], arbitrary time scale problems [17], diffusion transport [18], neutron dynamics[19] and other fields. Because the conformable derivatives have good properties, the research on the theory and algorithm of partial differential equations with conformable derivatives has also attracted extensive interest. [20] studied the stochastic solution of equations with conformable time derivative where the space operators may correspond to fractional Brownian motion, or a Lévy

process, or a general semigroup in a Banach space, or a process killed upon exiting a bounded domain in \mathbb{R}^d . [21] introduced a modern approach for solving the nonlinear evolution equations in the frame of a recent generalized conformable derivative. A conformable delay perturbation of matrix exponential function was offered to give the representation of solutions for linear nonhomogeneous conformable fractional delay differential equations, and the existence and uniqueness of solutions and Ulam-Hyers stability of the equations were proved in [22]. The delayed exponential matrix function in conformable version was established and used to derive the expression of the solution for homogeneous and nonhomogeneous equations respectively in [23]. [25] introduced novel approximate numerical approach, so called "extended reduced conformable differential transform method (ERC-DTM)" which was the implementation of DTM for conformable time fractional partial differential equations with proportional delay. In [24], the truncated solution of space-time fractional differential equations, including conformable derivative was constructed by the help of residual power series method. However, there are few literatures on using neural network methods to numerically solve the conformable time fractional diffusion equations.

In this paper, we consider using PINNs to solve the conformable time fractional diffusion equation in the following form,

$$\begin{cases} T^\alpha u(t, x) - \lambda u_{xx}(t, x) = 0, & 0 < \alpha < 1, x \in \Omega, t \in (0, T], \\ u(0, x) = g(x), \\ u(t, 0) = \varphi(t), \end{cases} \quad (1)$$

where $u(t, x)$ represents the solution of the equation, $\lambda \in \mathbb{R}$ is a parameter in the equation, Ω is a subset of \mathbb{R} , T^α is the conformable derivative. Our aim is to give the numerical solution of the conformable time fractional diffusion equation based on the parameter λ at a given time.

We summarize the main contributions and findings as follows:

(1) The common fractional derivatives, such as Riemann Liouville derivative and Caputo derivative, do not meet the chain rule and can not directly encode the deep neural network through the chain rule, so that the corresponding fractional differential equations can not be solved by PINNs. However, we found a fractional derivative with good properties, that is, the conformable derivative, which satisfies Leibniz's rule and chain rule. Therefore, in this paper, the PINN method is used to solve the fractional diffusion equation with integrated derivative for the first time, and experiments show that our method has good simulation effect. According to this, we have filled in the defect that neural network can not be used to solve fractional differential equations in some previous articles directly.

(2) As $\alpha \rightarrow 1$, due to the influence of the conformable derivative, the singularity of the solution of the equation (1) increases, resulting in the decline of the accuracy of our method. Therefore, we propose a weighted PINN method to deal with the influence of the singular solution by constraining

the solution of the equation (1).

(3) We use the PINN method to solve the inverse problem of the equation (1) and accurately identify the parameters of the equation (1). Even if the training data is corrupted with 1% uncorrelated Gaussian noise, our prediction is still robust.

The rest of this paper is arranged as follows. Section 2 briefly introduces the definition and related properties of the conformable derivative. In section 3, we present the main idea of our neural network method for solving the conformable time-fractional diffusion equation. In section 4, we solve the forward problem of the equation, i.e., taking IC/BCs as data to train our predictive solutions by minimizing the loss function. We solved the inverse problem, that is, using the data we obtained, to predict the parameter λ of the equation by using the neural network method in section 5. Finally, we summarize the work of this paper and look forward to the future work in section 6.

2. Preliminaries

This section gives some basic definitions and properties regarding the conformable derivative.

Definition 1. Given a function $f: [0, \infty) \rightarrow \mathbb{R}$ and $t > 0$, for all $t > 0$, $\alpha \in (0, 1)$, then the conformable fractional derivative of f of order α is defined by

$$T_\alpha(f)(t) = \lim_{\varepsilon \rightarrow 0} \frac{f(t + \varepsilon t^{1-\alpha}) - f(t)}{\varepsilon}. \quad (2)$$

If f is α -differentiable in some $(0, a)$, $a > 0$, and $\lim_{t \rightarrow 0+} f^{(\alpha)}(t)$ exists, then define,

$$f^{(\alpha)}(0) = \lim_{t \rightarrow 0+} f^{(\alpha)}(t).$$

In addition, if the conformable fractional derivative of f of order α exists, then we simply say f is α -differentiable.

Noted that $T_\alpha(t^p) = pt^{p-\alpha}$ is established. Further, the definition of conformable derivative is consistent with the classical Riemann-Liouville fractional derivative and the Caputo fractional derivative of polynomials, up to a constant multiple.

Definition 2. If a function $f: [0, \infty) \rightarrow \mathbb{R}$ is α -differentiable at $t_0 > 0$, $\alpha \in (0, 1]$, then f is continuous at t_0 .

Definition 3. If $\alpha \in (0, 1]$ and f, g is α -differentiable at $t > 0$, then,

- (1) $T_\alpha(af + bg) = aT_\alpha(f) + bT_\alpha(g)$, for all $a, b \in \mathbb{R}$.
- (2) $T_\alpha(t^p) = pt^{p-\alpha}$ for all $p \in \mathbb{R}$.
- (3) $T_\alpha(\lambda) = 0$, for all constant functions $f(t) = \lambda$.
- (4) $T_\alpha(fg) = fT_\alpha(g) + gT_\alpha(f)$.
- (5) $T_\alpha\left(\frac{f}{g}\right) = \frac{gT_\alpha(f) - fT_\alpha(g)}{g^2}$.
- (6) In addition, if f is differentiable, then $T_\alpha(f)(t) = t^{1-\alpha} \frac{df}{dt}(t)$.

3. Methodology

For the conformable time-fractional diffusion equations(1), we use $f(t, x)$ to determine the left part, i.e.,

$$f(t, x) = T^\alpha u(t, x) - \lambda u_{xx}(t, x), \quad (3)$$

where $u(t, x)$ can be approximated by using a deep neural networks. Different from other definitions of fractional derivatives, conformable derivatives satisfy the chain rule, so we can approach it by automatic differentiation. To highlight the simplicity of implementing this idea, we use a Python snippet containing TensorFlow. TensorFlow is a symbolic mathematics system based on data flow programming, which is widely applied in the programming implementation of various machine learning algorithms. Thus, $u(t, x)$ can be simply defined as:

```
def u(t, x) :
    u = neural_net(tf.concat([t, x], 1), weights, biases)
return u
```

Similarly, $f(t, x)$ can be simply defined as:

```
def f(t, x) :
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t - lambda * u_xx
return f
```

During the training of PINN, we approximate the solution of the equation by minimizing the following loss function.

$$Loss = MSE_u + MSE_f, \quad (4)$$

where MSE represents the mean square error and the parameters of neural networks $u(t, x)$ and $f(t, x)$ are shared. The loss function consists of two parts, which are specifically expressed as follows:

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

where $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}, N_u$ represent the initial and the boundary training data of the equation, the number of training points for ICs and BCs, respectively,

$$MSE_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

where $\{t_f^i, x_f^i\}$ represent the collocation of neural network $f(t, x)$. The error MSE_u corresponds to the initial values and boundary values of the equation, while the error MSE_f is used to reinforce the structure imposed by equation (1) within a finite set of collocation points.

The neural network $u(t, x)$ trained by IC/BCs and the neural network $f(t, x)$ containing physical information both contribute to the loss function and share the hyper-parameters, which together constitute the PINN architecture for conformable time-fractional diffusion equations as displayed in the following figure 1.

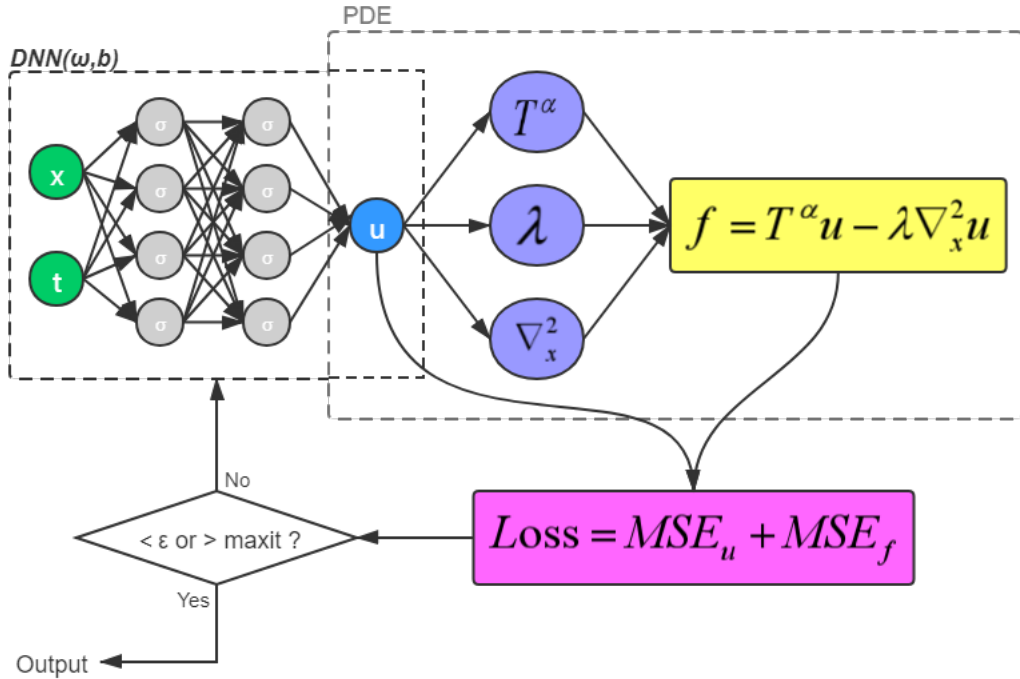


Figure 1: Schematic of PINN for the conformable time-fractional diffusion equations.

By the employment of PINN, we can take IC/BCs as the data to solve the corresponding partial differential equation since it is obvious that the conformable time-fractional diffusion equation is wellposed, with suitable IC/BCs. This is referred to as the forward problem, and we shall report our results in section 4; On the other hand, assuming that we have some of the training data, which may contain unrelated noise, the parameterized equations are learned by making use of it. Likewise, it is

referred to as the inverse problem and in section 5.

4. Forward problems

In this section, we aim to solve the forward problem for the conformable time-fractional diffusion equations, that is, to obtain the neural network approximation of the solution of the equation using IC/BCs and prior physical information. It is worth mentioning that for wellposed forward problems, fractional partial differential equations can usually be uniquely solved by using IC/BCs.

For this forward problem, by taking IC/BCs as data the loss function (4) can be established and designed as

$$Loss = MSE_u + MSE_f = MSE_{IC} + MSE_{BC} + MSE_f \quad (5)$$

where

$$MSE_{IC} = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} |u(0, x_{IC}^i) - g(x_{IC}^i)|^2,$$

and

$$MSE_{BC} = \frac{1}{N_{BC}} \sum_{j=1}^{N_{BC}} |u(t_{BC}^j, 0) - \phi(t_{BC}^j)|^2,$$

here $\{x_{IC}^i, g(x_{IC}^i)\}_{i=1}^{N_{IC}}, \{t_{BC}^j, \phi(t_{BC}^j)\}_{j=1}^{N_{BC}}, N_{IC}, N_{BC}$ denote the initial value of the equation, the boundary training data of the equation, the number of training points for ICs and the number of training points for BCs, respectively. Besides, MSE_f applied to represent prior physical information can be rewritten as

$$\begin{aligned} MSE_f &= \frac{1}{N_f} \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2 \\ &= \frac{1}{N_f} \sum_{i=1}^{N_f} |T^\alpha u(t_f^i, x_f^i) - \lambda u_{xx}(t_f^i, x_f^i)|^2 \end{aligned}$$

where $\{t_f^i, x_f^i\}$ indicate the collocation of neural network $f(t, x)$ and N_f represents the number of collocation points. The error MSE_f in the neural network $f(t, x)$ is applied to reinforce the structure imposed by equation (1) within a finite set of collocation points.

Next, we will carry out three examples to demonstrate the applicability of our neural network method in solving conformable time-fractional diffusion equations. All experiments are implemented on a computer, which is configured as follows: Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz 1.80 GHz, using Python3.6 + Tensorflow1.15. The same configuration is applied in the inverse problem in Section 5.

Example 1. We consider equation (1) with $\alpha = 0.5$, $\lambda = 0.5073$, which has an analytical solution of the following form,

$$u(t, x) = \sqrt{\frac{\alpha}{4\pi\lambda t^\alpha}} \exp\left\{-\frac{\alpha}{4\lambda t^\alpha} x^2\right\},$$

the image of the analytical solution is exhibited in figure 2.

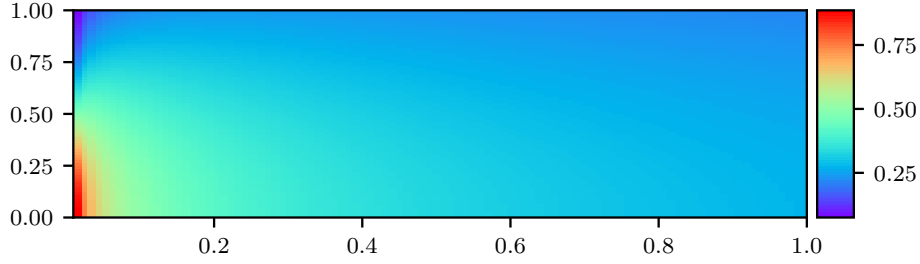


Figure 2: Analytical solution of the equation when $\alpha = 0.5$.

Figures 3-4 reveal the predicted results of our neural network method for data-driven solutions of the conformable time-fractional diffusion equation. Specifically, given a set of initial and boundary data with $N_u = N_{IC} + N_{BC} = 100$ and a set of collocation points with $N_f = 10000$, both of them are randomly distributed. Next, we employ the mean square error loss function defined in equation (4) to train 3021 parameters of the deep neural network that has 9 hidden layers with 20 neurons at each layer, and learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation. The hyperbolic tangent function serve as our activation function over here. At the top pannel in figures 3-4, we display the space-time solution $u(t, x)$ predicted by our neural network and the location of the initial and boundary training data. What here must stress specially is that, all classical numerical methods for solving partial differential equations require discretization of the equations, but our method does not require any discretization in time or space. In the case, the exact solution given is analytically available and relative \mathbb{L}_2 error of the predicted solution is measured as $2 \cdot 10^{-3}$. At the bottom pannel of of figures 3-4, a more detailed evaluation of the predicted solution is offered. To be specific, figure 3 is presented with the comparison between the exact solution and the predicted solution at three different moments of $t = 0.01, 0.05, 0.10$. In addition, the comparison between the exact solution and the predicted solution $t = 0.25, 0.50, 0.75$ is displayed in figure 4.

The error between predicted solution and exact solutions when $\alpha = 0.5$ is drawn in figure 5. We can observe that the error is particularly close to 0, while the average error is about $2.0 \cdot 10^{-3}$. In figure 6, we exhibit the variances between predicted solution and exact solutions for $\alpha = 0.5$. It is worth mentioning that the mean square error is about $4.2 \cdot 10^{-7}$. From figures 5-6, we can find that the predicted solution is a good approximation of the exact solutions for the conformable time-fractional diffusion equation at $\alpha = 0.5$.

Example 2. In this example, the analytical solution of equation (1) with $\alpha = 0.3$, $\lambda = 0.5073$ is given by

$$u(t, x) = \sqrt{\frac{\alpha}{4\pi\lambda t^\alpha}} \exp\left\{-\frac{\alpha}{4\lambda t^\alpha} x^2\right\},$$

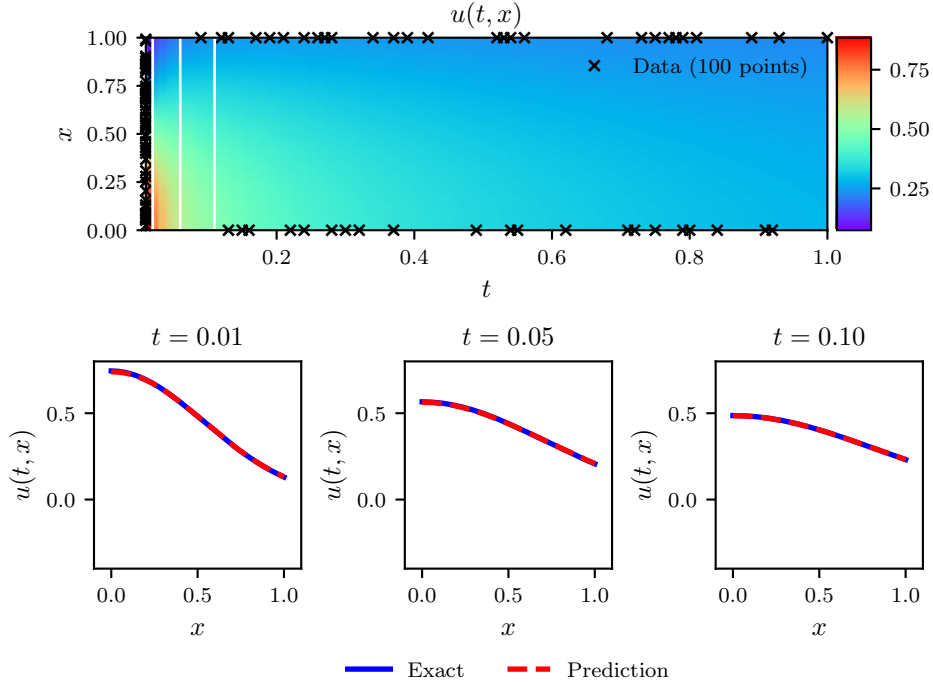


Figure 3: Comparison between the exact solution of the equation at $\alpha = 0.5$ and the exact solution and the predicted solution at three different times of $t = 0.01, 0.05, 0.10$.

whose image is described in 7.

For the conformable time-fractional diffusion equation, the results predicted by neural network method in are listed in figures 8-9. Concretely, both a set of initial and boundary data with $N_u = N_{IC} + N_{BC} = 100$ and a set of collocation points with $N_f = 10000$ are randomly distributed. Then we make use of the mean square error loss function defined in equation (4) to learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation by training 3021 parameters while the deep neural network has 9 hidden layers with 20 neurons per layer. Here, we pick the hyperbolic tangent function as activation function. Figures 8-9 manifest the space-time solution $u(t, x)$ predicted by our neural network and the location of the initial and boundary training data at the top panel. For this problem, the relative error of the predicted solution is measured as $1.8 \cdot 10^{-3}$ in the \mathbb{L}_2 norm. At the bottom panel of of figures 8-9, we reveal the comparison between the exact solution and the predicted solution at $t = 0.01, 0.05, 0.10$ and $t = 0.25, 0.50, 0.75$, respectively.

In figure 10, we exhibit the error of predicted solution and exact solutions with $\alpha = 0.3$. What we can see is that the error is close to 0 particularly and the average error is about $1.8 \cdot 10^{-3}$. Besides, the variances between predicted solution and exact solutions with $\alpha = 0.3$ are portrayed in figure 11, while the mean square error is about $2.6 \cdot 10^{-7}$. In addition, for the conformable time-fractional diffusion equation at $\alpha = 0.3$, we can discover that the predicted solution approximate the exact solutions pretty

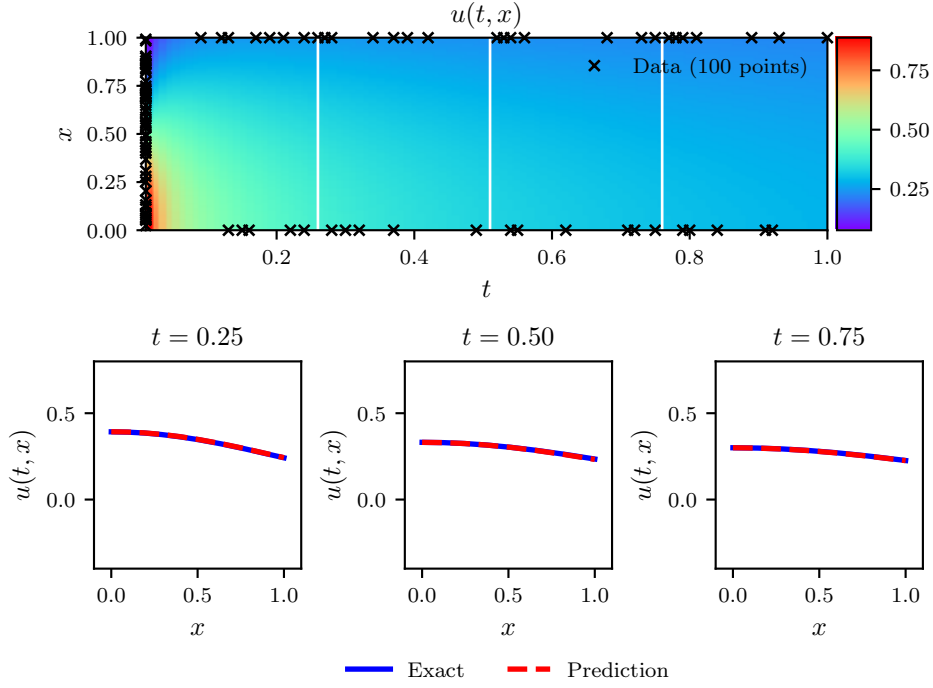


Figure 4: Comparison between the exact solution of the equation at $\alpha = 0.5$ and the exact solution and the predicted solution at three different times of $t = 0.25, 0.50, 0.75$.

well from figures 10-11.

Example 3. Figure 12 reveals the analytical solution of equation (1), which can be represented as the following form,

$$u(t, x) = \sqrt{\frac{\alpha}{4\pi\lambda t^\alpha}} \exp\left\{-\frac{\alpha}{4\lambda t^\alpha} x^2\right\},$$

where $\alpha = 0.8$, $\lambda = 0.5073$.

Figures 13-14 provide the predicted results by making use of our neural network method for the conformable time-fractional diffusion equation. Specifically, both the initial and boundary data with $N_u = N_{IC} + N_{BC} = 100$ and the collocation points with $N_f = 10000$ are selected randomly. And

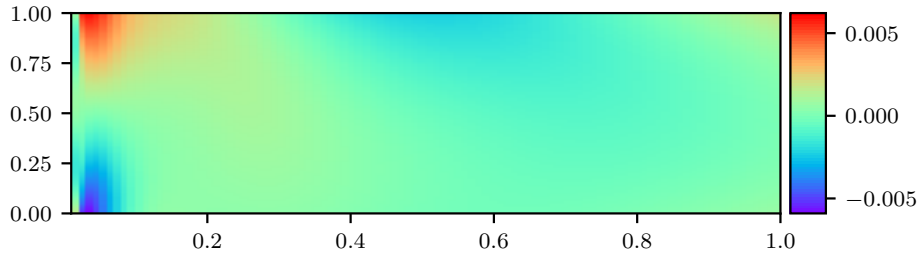


Figure 5: Error between the predicted solutions and exact solutions of the equation when $\alpha = 0.5$.

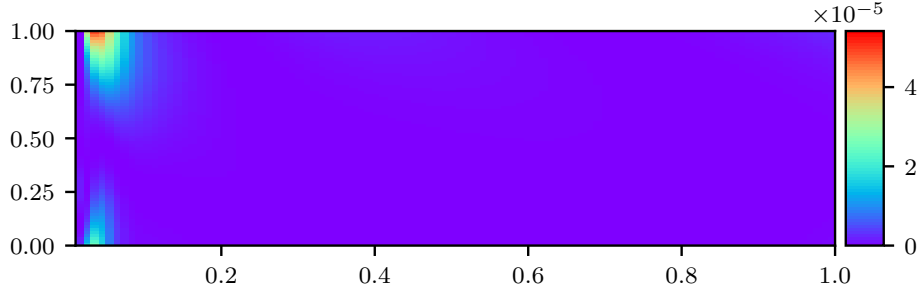


Figure 6: Variance of the predicted solutions and exact solutions of the equation when $\alpha = 0.5$.

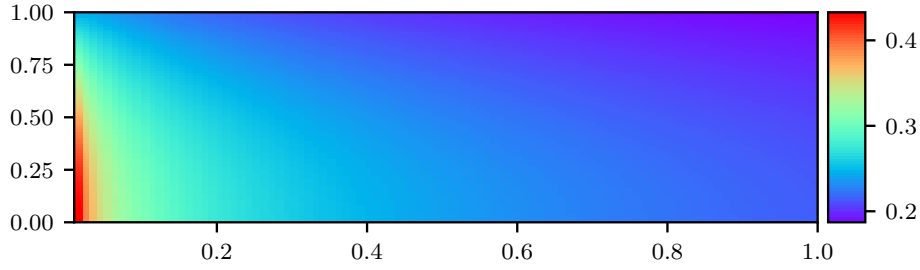


Figure 7: Analytical solution of the equation when $\alpha = 0.3$.

then the mean square error loss function defined in equation (4) is employed to train 3021 parameters of the deep neural network which has 9 hidden layers with 20 neurons at each layer and learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation. In the same way, the hyperbolic tangent function is chosen as our activation function. At the top pannel of figures 13-14, we exhibit the space-time solution $u(t, x)$ predicted by our neural network and the location of the initial and boundary training data. In addition, the exact solution given is truly available and the relative \mathbb{L}_2 error between predicted solution and exact solution is measured as $1.4 \cdot 10^{-2}$. In addition, more detailed evaluation of the predicted solution is given at the bottom panel of figures 13-14. In detail, we give the comparison between the exact solution and the predicted solution at three different moments of $t = 0.01, 0.05, 0.10$ in figure 13. Unfortunately, we can see that the predicted solution has some error at $t = 0.01$ and some flaws at both $t = 0.05$ and $t = 0.10$ in this figure. We will discuss why this is the case and how to solve it later in this section. On the contrary, in figure 14, the predicted solution we got at three different moments of $t = 0.25, 0.50, 0.75$ is performing well.

In figure 15, the error between predicted solution and exact solutions with $\alpha = 0.8$ is described. What we show is that the error is extremely close to 0, and the average error is about $1.4 \cdot 10^{-2}$. In addition, we exhibit the variances of predicted solution and exact solutions at $\alpha = 0.8$ in figure 16 and the mean square error is about $3.7 \cdot 10^{-5}$. However, variances we got are relatively large in the

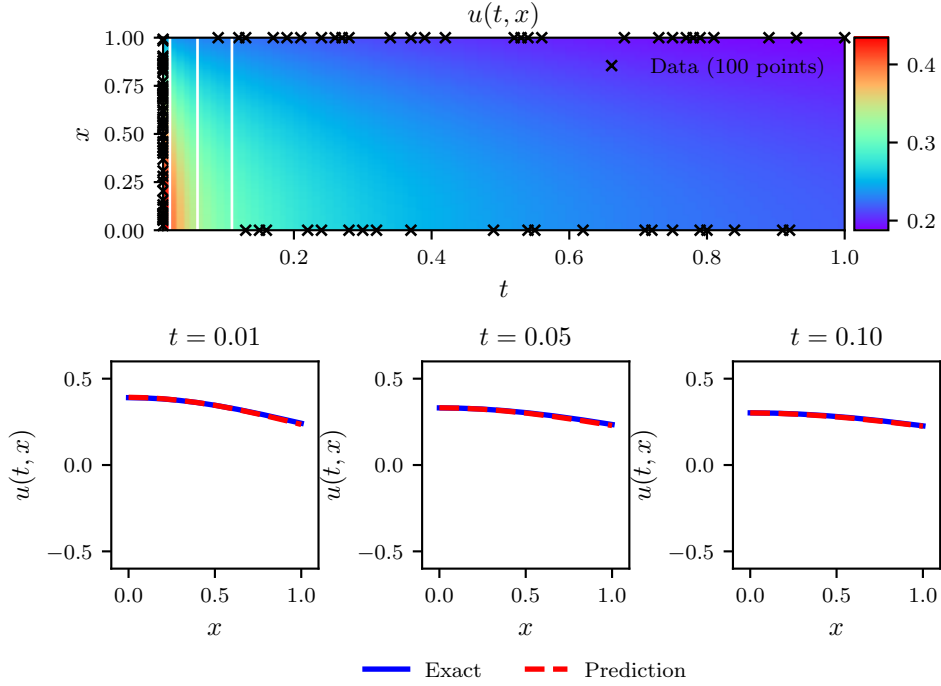


Figure 8: Comparison between the exact solution of the equation at $\alpha = 0.3$ and the exact solution and the predicted solution at three different times of $t = 0.01, 0.05, 0.10$.

Table 1: The mean error and mean variance between the predicted solutions and exact solutions of the conformable time-fractional diffusion equation when $\alpha = 0.3, 0.5, 0.8$.

α	0.3	0.5	0.8
Average error	$1.8 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$
Mean square error	$4.2 \cdot 10^{-7}$	$2.6 \cdot 10^{-7}$	$3.7 \cdot 10^{-5}$

singular solution region while particularly close to 0 in all other regions. For the equation at $\alpha = 0.8$, the predicted solution has a good approximation out of the singular solution region in figures 15-16.

Table 1 presents the mean error and mean variance between the predicted solution and the exact solution of the conformable time-fractional diffusion equation when $\alpha = 0.3, 0.5, 0.8$. It can be observed that, with the increase of the fractional order α of the conformable derivative, the singularity of the solution of the equation increases as it tends to 0. Also, the error of the solution of the equation increases as well. Therefore, we propose a weighted neural network method to deal with the complex nonlinear behavior caused by the singular solution of the equation when the fractional order α tends to 1.

In our original neural network method, the loss function is defined as the equation (4). It is a difficulty that when α close to 1, the complex nonlinear behavior caused by the singular solution of

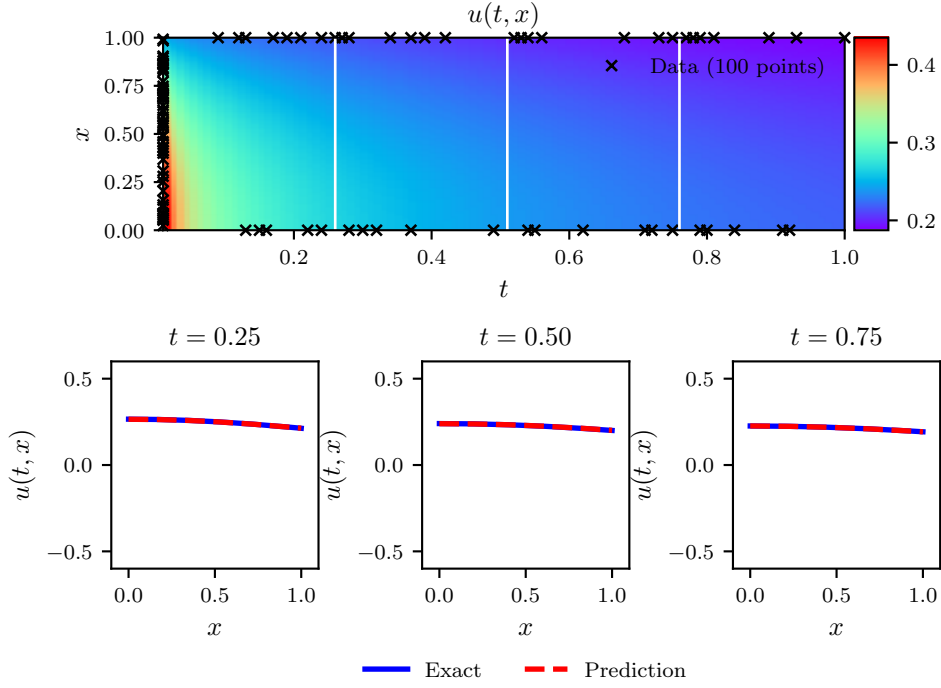


Figure 9: Comparison between the exact solution of the equation at $\alpha = 0.3$ and the exact solution and the predicted solution at three different times of $t = 0.25, 0.50, 0.75$.

equation is difficult to be simulated. Therefore, we consider to weight the neural network $u(t, x)$ and $f(t, x)$ and vest the mean square error (MSE) greater weight. By doing so, the predicted solution of the neural networkmethod can be constrained and the influence of singular solution of the equation can be reduced better.

The parameters of neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean square error (MSE)loss function, which is defined as follows:

$$MSE = w_u * MSE_u + w_f * MSE_f, \tag{6}$$

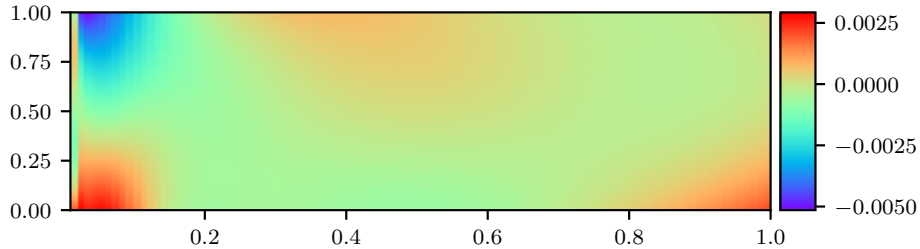


Figure 10: Error between the predicted solutions and exact solutions of the equation when $\alpha = 0.3$.

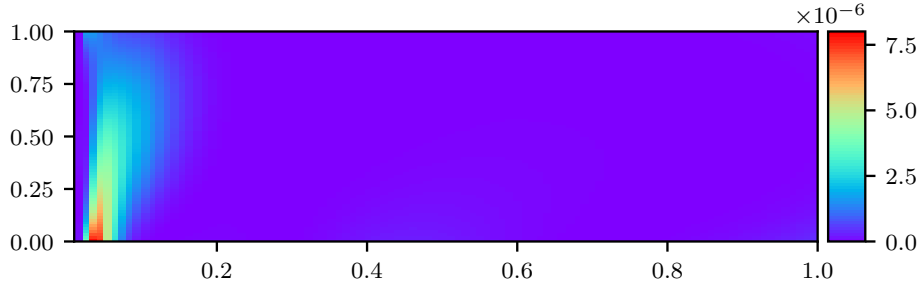


Figure 11: Variance of the predicted solutions and exact solutions of the equation when $\alpha = 0.3$.

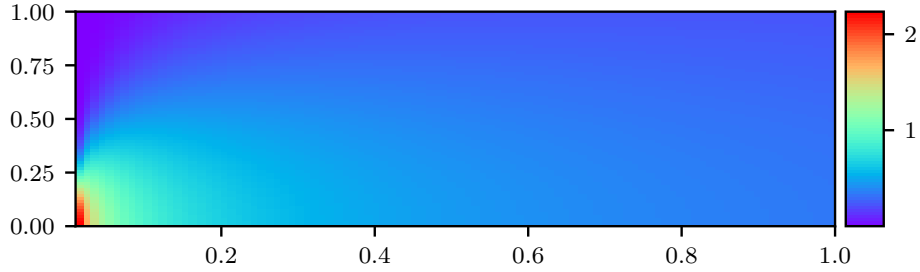


Figure 12: Analytical solution of the equation when $\alpha = 0.8$.

where w_u and w_f imply neural network $u(t, x)$ and $f(t, x)$, respectively. Since we have used this symbol w to represent the weight of the neural network, here we use w as the weight symbol instead to avoid confusion. Next, we provide experimental results to verify the accuracy of our weighted neural network method.

For the equation (1) with $\alpha = 0.8$, $\lambda = 0.5073$, we take $w_u = 1.0, w_f = 0.1$ in our weighted neural network method. Figure 17 shows the predicted results of our neural network method for the data-driven solutions of the conformable time-fractional diffusion equation. To make it more concrete, given a set of initial and boundary data with $N_u = N_{IC} + N_{BC} = 100$ and a set of collocation points with $N_f = 10000$, both of them are randomly distributed. After that, we exploit the mean square error loss function defined in equation (4) to train 3021 parameters of the deep neural network that has 9 hidden layers with 20 neurons per layer, and learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation. Similarly, the hyperbolic tangent function is selected as our activation function.

Results of the prediction error is measured as $1.4 \cdot 10^{-3}$ in the relative \mathbb{L}_2 norm. The comparison between the exact solution and the predicted solution at three different times of $t = 0.01, 0.05$ and 0.10 is shown, see figure 17. To our delight, we find that the accuracy of our weighted neural network method is fine at $t = 0.01, 0.05$ and 0.10 .

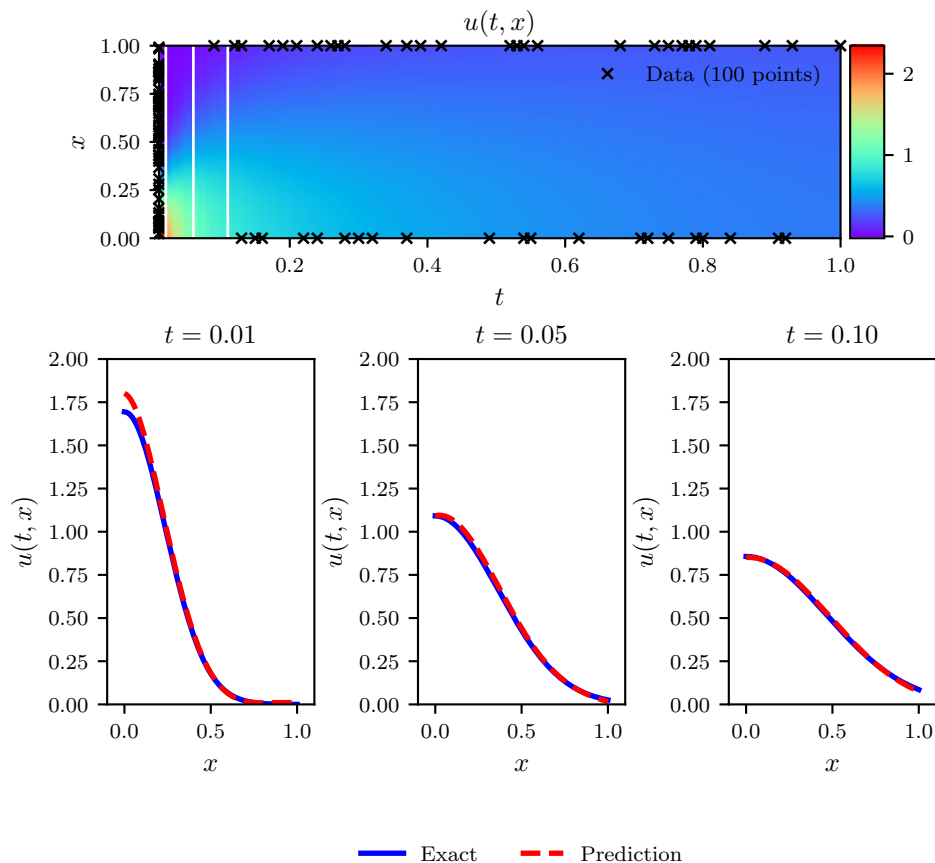


Figure 13: Comparison between the exact solution of the equation at $\alpha = 0.8$ and the exact solution and the predicted solution at three different times of $t = 0.01, 0.05, 0.10$.

In figure 18, we exhibit the error between predicted solution and exact solutions when $\alpha = 0.8$. At this time, we can see that the error is adequately close to 0, and the average error is about $1.4 \cdot 10^{-3}$. Figure 19 shows the variances of our predicted solution and exact solutions for $\alpha = 0.8$ and the mean square error is about $3.5 \cdot 10^{-7}$. Through figures 18-19, what we want to demonstrate is that the predicted solution is a good approximation of the exact solutions for the conformable time-fractional diffusion equation at $\alpha = 0.8$.

To further analyze the performance of our method, we conduct a systematic study to quantify the effects of different sampling points (training points, collocation points) and neural network structures (layers, the number of neurons in each layer) on the prediction accuracy. From Table 2, we show the relative \mathbb{L}_2 - error generated by different number of initial and boundary training data N_u and different number of collocation points N_f for conformable time-fractional diffusion equation when $\alpha = 0.5$. Here we keep the structure of the 9-layer deep neural network with 20 neurons per layer unchanged. The general trend is that in the case of enough collocation points N_f , the prediction accuracy will increase

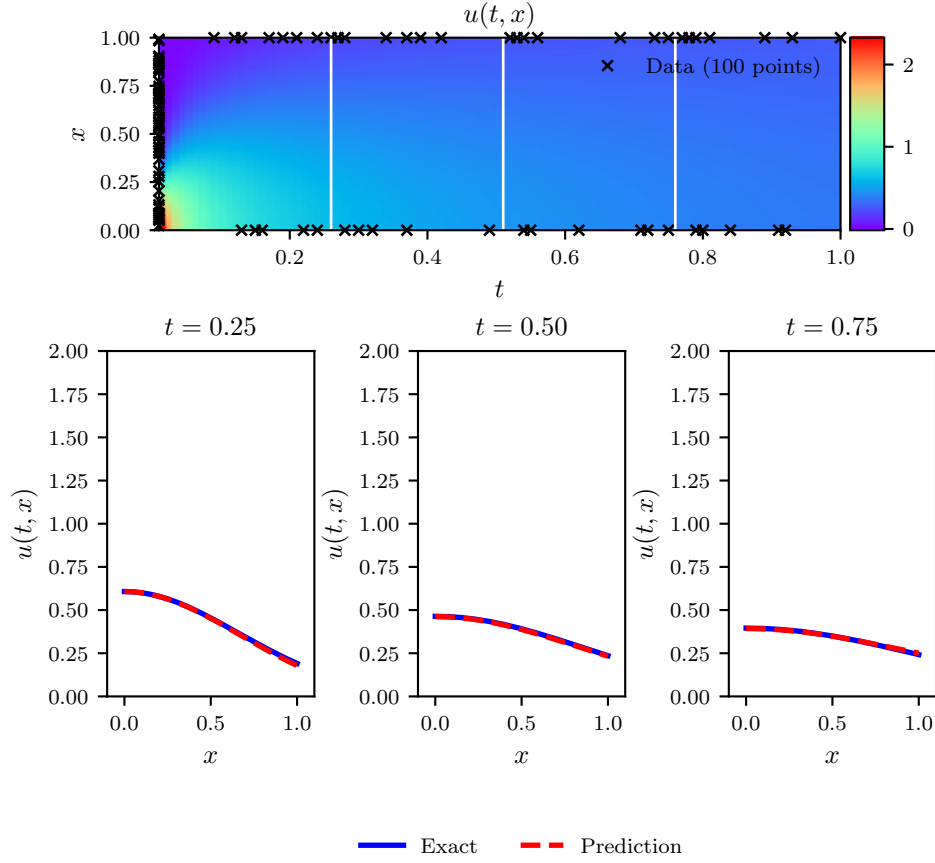


Figure 14: Comparison between the exact solution of the equation at $\alpha = 0.8$ and the exact solution and the predicted solution at three different times of $t = 0.25, 0.50, 0.75$.

with the increase of the total training data N_u . This observation highlights a key advantage of our neural network method: by configuring the point N_f to encode the structure of the underlying physical laws, a more accurate and data-efficient learning algorithm can be obtained.

Finally, Table 3 collects the relative \mathbb{L}_2 errors generated by different hidden layers and neurons in each layer for the conformable time-fractional diffusion equation when $\alpha = 0.5$. Similarly, the total number of training points and collocation points is fixed at $N_u = 100$ and $N_f = 10,000$, respectively. As expected, we observe that as the number of layers and the number of neurons per layer increases, this also mean that the ability of the neural network to approximate more complex functions increases, and the prediction accuracy increases accordingly.

Table 2: Different initial and boundary number of training data N_u and different collocation points N_f , the relative \mathbb{L}_2 error between the predicted values and the exact solution $u(t, x)$. Here, the network structure is fixed at nine layers, with 20 neurons in each hidden layer.

$N_u \backslash N_f$	2000	4000	6000	8000	10000
20	$1.4 \cdot 10^{-2}$	$3.7 \cdot 10^{-2}$	$8.5 \cdot 10^{-3}$	$7.4 \cdot 10^{-3}$	$5.2 \cdot 10^{-3}$
40	$1.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$1.1 \cdot 10^{-2}$	$7.0 \cdot 10^{-3}$	$5.3 \cdot 10^{-3}$
60	$9.6 \cdot 10^{-3}$	$1.3 \cdot 10^{-2}$	$9.6 \cdot 10^{-3}$	$5.8 \cdot 10^{-3}$	$4.6 \cdot 10^{-3}$
80	$8.1 \cdot 10^{-3}$	$8.2 \cdot 10^{-3}$	$7.2 \cdot 10^{-3}$	$4.8 \cdot 10^{-3}$	$4.2 \cdot 10^{-3}$
100	$7.4 \cdot 10^{-3}$	$6.4 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$	$3.6 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$
200	$9.5 \cdot 10^{-3}$	$4.3 \cdot 10^{-3}$	$3.8 \cdot 10^{-3}$	$2.6 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$

5. Inverse problems

In this section, we turn our attention to the inverse problem for the conformable time-fractional diffusion equations. Assume in advance that we can get some training data, at this time we want to use it to learn parameterized equations. Usually, the data we derived is mixed with some uncorrelated noise, and we want to know whether our method is still applicable in this case.

Table 3: Relative \mathbb{L}_2 error between the predicted result and the exact solution $u(t, x)$ under different hidden layers and neurons per layer. Here, the total number of training points and matching points is fixed at $N_u = 100$ and $N_f = 10,000$, respectively.

\backslash Neurons	10	20	40
Layers			
2	$7.4 \cdot 10^{-2}$	$6.8 \cdot 10^{-2}$	$9.8 \cdot 10^{-3}$
4	$3.1 \cdot 10^{-2}$	$7.1 \cdot 10^{-3}$	$6.3 \cdot 10^{-3}$
6	$4.4 \cdot 10^{-2}$	$8.8 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$
8	$8.2 \cdot 10^{-3}$	$4.9 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$

$$MSE_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u(0, x_{data}^i) - g(x_{data}^i)|^2,$$

and

$$\begin{aligned} MSE_f &= \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |f(t_{data}^i, x_{data}^i)|^2 \\ &= \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |T^{\alpha} u(t_{data}^i, x_{data}^i) - \lambda u_{xx}(t_{data}^i, x_{data}^i)|^2, \end{aligned}$$

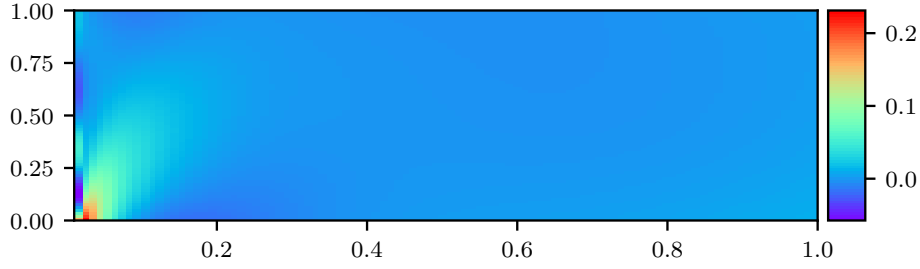


Figure 15: Error between the predicted and exact solutions of the equation when $\alpha = 0.8$.

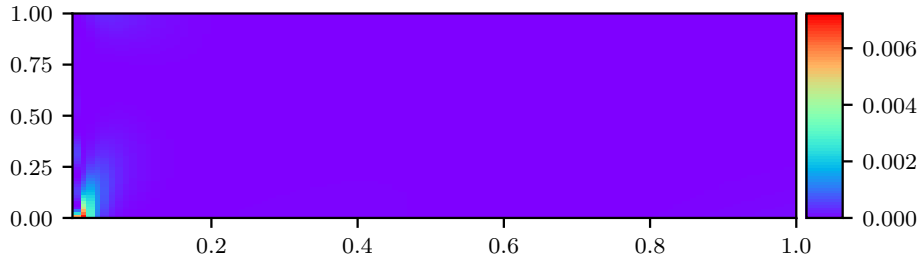


Figure 16: Variance of the predicted solutions and exact solutions of the equation when $\alpha = 0.8$.

where $\{x_{data}^i, g(x_{data}^i)\}_{i=1}^{N_{data}}$, N_{data} signify the training data on $u(t, x)$ and the number of training points, respectively. In particular, the loss MSE_u corresponds to the training data on $u(t, x)$ while MSE_f enforces the structure imposed by the conformable time-fractional diffusion equation at a finite set of collocation points, whose number and location are taken to be the same as the training data. At this time, λ is the parameter we want to learn.

Similarly, we bestow three examples to demonstrate the applicability of our neural network method in solving the inverse problem of conformable time-fractional diffusion equations. What is worth mentioning is that all experiments are performed on a computer, whose configuration is the same as in Section 4.

Example 4. In this example, we consider equation (1) with $\alpha = 0.5$. In order to illustrate the effectiveness of our method, we create a training dataset by randomly generating $N = 2000$ points across the entire spatio-temporal domain from the exact solution corresponding to $\lambda = 0.5073$. Next, we employ $N_{data} = 2000$ training points to train 3021 parameters of the deep neural network that has 9 hidden layers with 20 neurons at each layer by using the Adam optimizer and L-BFGS optimizer to minimize the mean square error loss function defined in equation (4) and learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation. Here, the hyperbolic tangent function served as activation function. After training, the network is calibrated to predict the entire solution $u(t, x)$, as

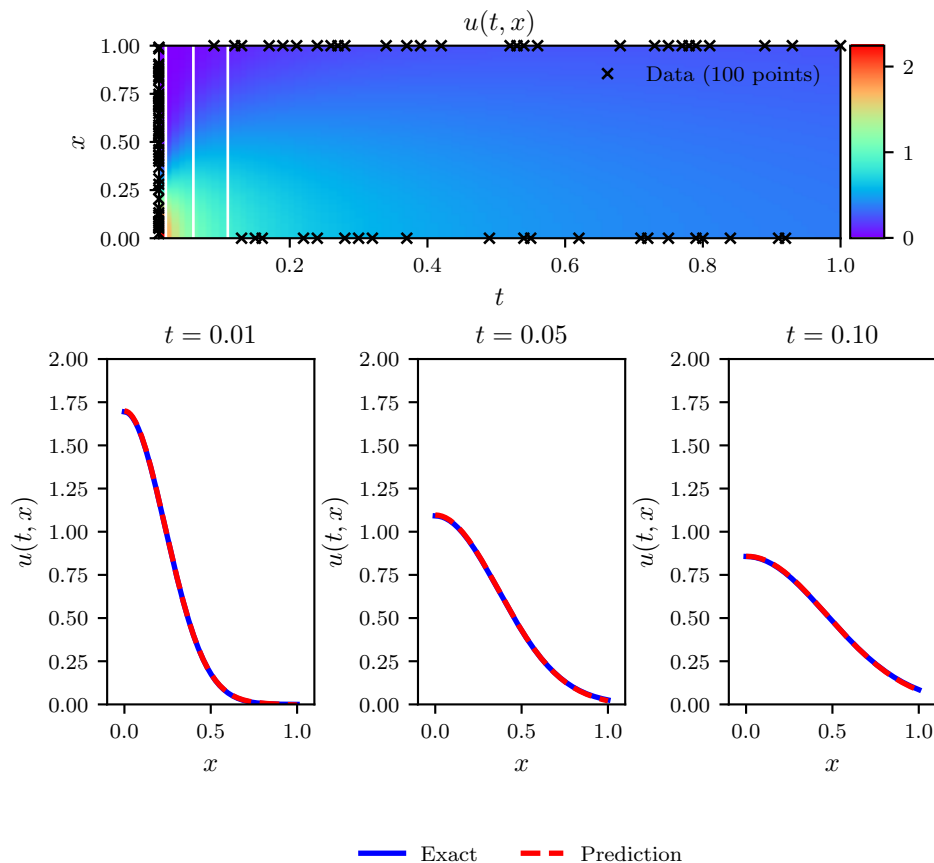


Figure 17: Comparison between the exact solution of the equation at $\alpha = 0.8$ and the exact solution and the predicted solution at three different times of $t = 0.01, 0.05, 0.10$.

well as the unknown parameter λ .

In figure 20 , we summarize the results for the conformable time-fractional diffusion equation with $\alpha = 0.5$. At the top panel, we supply the predicted solution of the equation whose analytical solution can be found in figure 2. At the middle panel, the graph of the predicted solution and the exact solution at $t = 0.01, 0.05, 0.10$ is drawn. At the bottom panel, we furnish the correct PDE and the identified PDE with clean data and 1% noise. We observe that our neural network method can accurately identify the unknown parameter λ , even if the training data is corrupted with noise. Specifically, the estimation errors of λ is 0.05% in the case of clean training data. Although the training data is corrupted with 1% uncorrelated Gaussian noise, the prediction is still robust, returning the error of λ is 0.60%.

Example 5. The equation (1) with $\alpha = 0.3$ is taken into account in this example. In order to illustrate the effectiveness of our method, we create a training dataset by generating $N = 2000$ points randomly across the entire spatio-temporal domain from the exact solution corresponding to $\lambda = 0.5073$. And

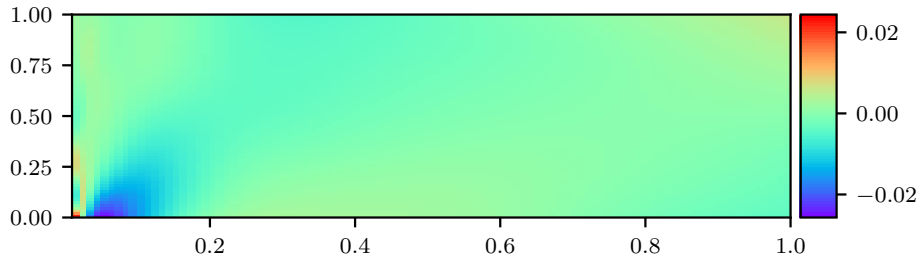


Figure 18: error of the weighted neural network predicted solutions and exact solutions of the equation when $\alpha = 0.8$.

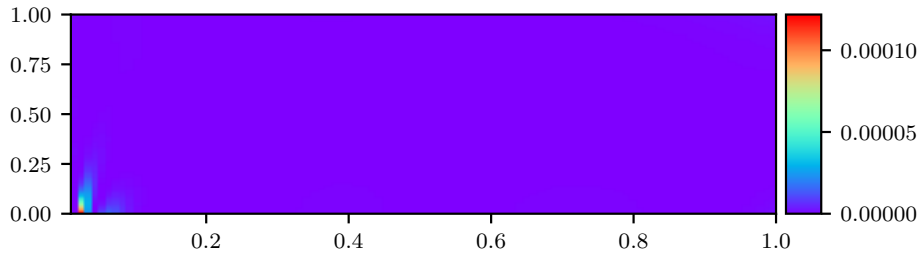


Figure 19: The variance of the weighted neural network predicted solutions and exact solutions of the equation when $\alpha = 0.8$.

then, we make use of $N_{data} = 2000$ training points to train 3021 parameters of the deep neural network which has 9 hidden layers with 20 neurons per layer by using the Adam optimizer and L-BFGS optimizer to minimize the mean square error loss function defined in equation (4) and learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation. The hyperbolic tangent function is singled out as our activation function over here. Later, the network is calibrated to predict the entire solution $u(t, x)$ and the unknown parameter λ .

In figure 21, we summarize the results for the conformable time-fractional diffusion equation with $\alpha = 0.3$. At the top panel, we exhibit the predicted solution of the equation while its analytical solution can be found in figure 7. At the middle panel, the graph of the predicted solution and the exact solution at $t = 0.01, 0.05, 0.10$ is drawn. At the bottom panel, we exhibit the correct PDE and the identified PDE with between clean data and 1% noise. It is a good news that proposed neural network method can accurately identify the unknown parameter λ by making use of our neural network method, even if the training data is corrupted with noise. Concretely, the estimation errors of λ is 0.11% in the case of clean training data. Also, even though the training data is corrupted with 1% uncorrelated Gaussian noise, the prediction is still robust, returning the error of λ is 0.70%.

Example 6. In this example, the equation (1) with $\alpha = 0.8$ is taken into consideration. Here, we set up a training dataset by randomly generating $N = 2000$ points across the entire spatio-temporal domain

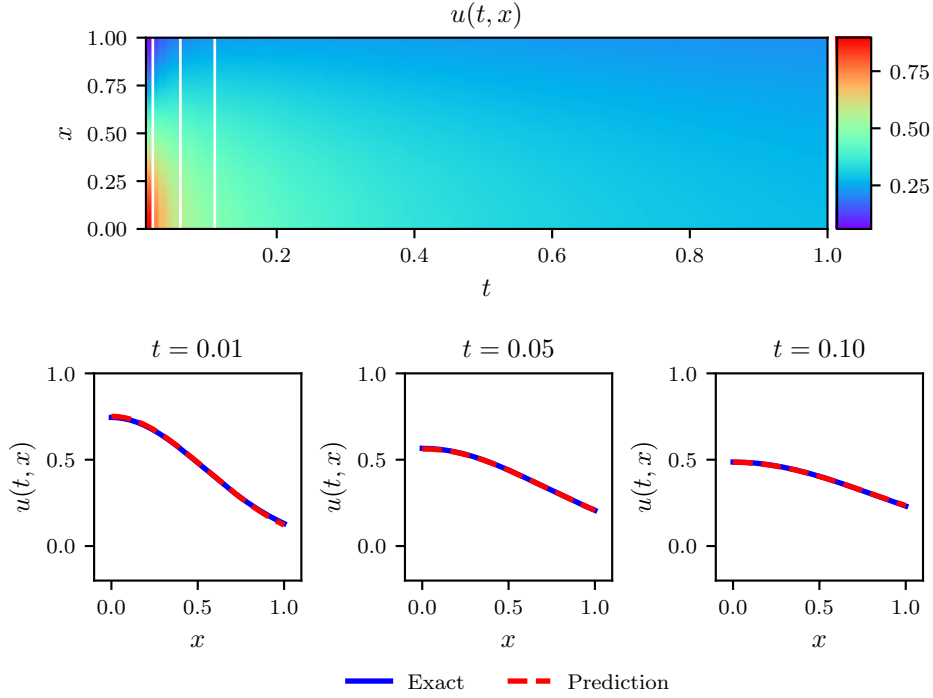


Figure 20: Equation with $\alpha = 0.5$: Top: Solution $u(t, x)$ along with the temporal locations of the three training snapshots. Middle: Training data and exact solution corresponding to the three temporal snapshots depicted by the dashed vertical lines in the top panel. Bottom: Correct partial differential equation along with the identified one obtained by learning λ .

from the exact solution corresponding to $\lambda = 0.5073$ to illustrate the effectiveness of our method. Then we use $N_{data} = 2000$ training points to train 3021 parameters of the deep neural network that has 9 hidden layers with 20 neurons at each layer by taking advantage of the Adam optimizer and L-BFGS optimizer to minimize the mean square error loss function defined in equation (4) and learn the solution $u(t, x)$ of the conformable time-fractional diffusion equation. At this point, the hyperbolic tangent function is used as our activation function. Afterwards, the network is calibrated to predict the entire solution $u(t, x)$, as well as the unknown parameter λ .

In figure 22, we sum up the results for the conformable time-fractional diffusion equation with $\alpha = 0.8$. At the top panel, we show the predicted solution of the equation whose image of analytical solution can be found in figure 12. At the middle panel, the graph of the predicted solution and the exact solution at $t = 0.01, 0.05, 0.10$ is drawn. At the bottom panel, we demonstrate the correct PDE

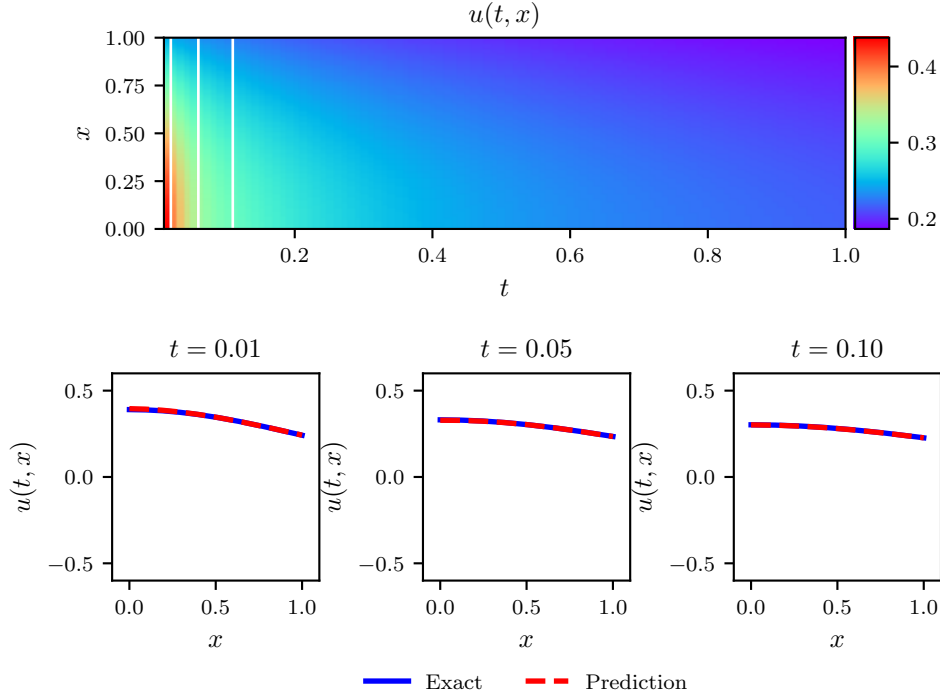


Figure 21: Equation with $\alpha = 0.3$: Top: Solution $u(t, x)$ along with the temporal locations of the three training snapshots. Middle: Training data and exact solution corresponding to the three temporal snapshots depicted by the dashed vertical lines in the top panel. Bottom: Correct partial differential equation along with the identified one obtained by learning λ .

and the identified PDE with both clean data and 1% noise. Through the figure, we observe that our neural network method can accurately identify the unknown parameter λ , even if the training data is corrupted with noise. Specifically, in the case of clean training data, the estimation errors of λ is 0.02%. Despite the training data is corrupted with 1% uncorrelated Gaussian noise, the prediction is still robust while the error of λ is 0.09%.

To sum up, our neural network method provides a high accuracy in solving the inverse problem of the conformable time-fractional diffusion equation, and correctly identify unknown parameters regardless of whether the training data is corrupted with noise or not. It is worth noting that it is incomparable to classical numerical methods.

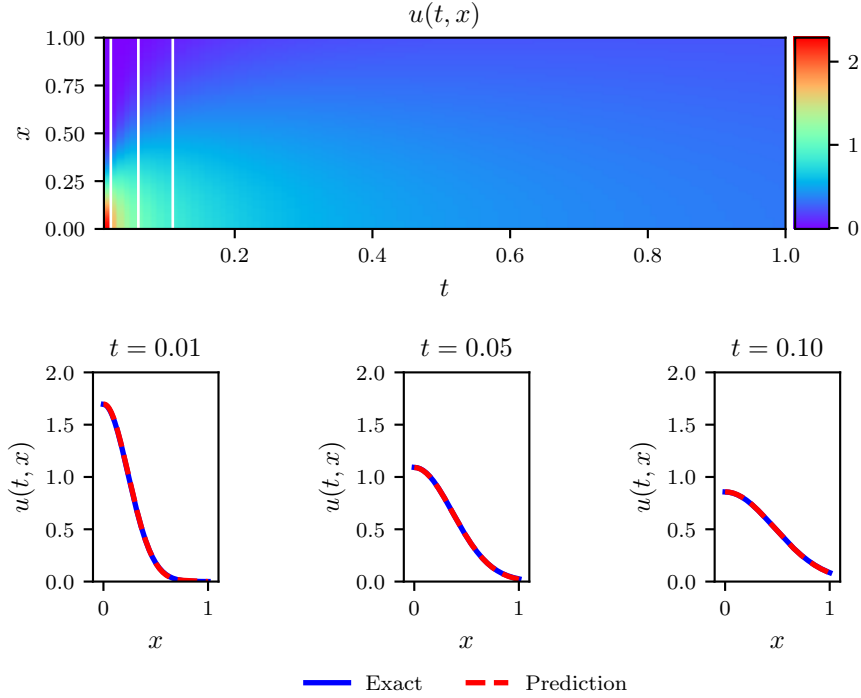


Figure 22: Equation with $\alpha = 0.8$: Top: Solution $u(t, x)$ along with the temporal locations of the three training snapshots. Middle: Training data and exact solution corresponding to the three temporal snapshots depicted by the dashed vertical lines in the top panel. Bottom: Correct partial differential equation along with the identified one obtained by learning λ .

6. Conclusion

In this paper, a neural network method is proposed to study the conformable time-fractional diffusion equation. Since conformable derivative has the nature of chain rule, automatic differential techniques can be applied, and our main idea is to use the PINN method to study the equation. To take this, we have filled in the defect that neural network can not be directly used to solve fractional differential equations in some previous articles. For the forward problem, we train the neural network to obtain predicted solution by using IC/BCs and give three numerical examples to validate the effectiveness. The error of predicted solution and exact solution is discussed. Particularly, when α approaches 1, the simulation effect of the singular solution part of the equation we study is not satisfactory. Therefore, we propose a weighted neural network method that it can constrain the singular solution better, so as to eliminate the influence of the singular solution. Besides, we conduct

a systematic study to quantify the effects of different sampling points (training points, collocation points) and neural network structures (layers, the number of neurons in each layer) on the prediction accuracy. The general trend shows that as the total number of training data N_u is increased when given a sufficient number of collocation points N_f or the number of layers and neurons is increased, the prediction accuracy is increased. For the inverse problem, we use the obtained data to train the neural network by minimizing the loss function we define and give a predicted value of the equation parameter λ . Three numerical examples are given, we show the correct PDE and the identified PDE with clean data and 1% noise. We observe that our method can accurately identify the parameters, even if the training data is corrupted with 1% uncorrelated gaussian noise.

Recently, many scholars have studied partial differential equations by means of neural network, and put forward many good methods. However, for solving the partial differential equations, the rise of deep learning methods also brings a lot of problems. We still have a lot of work to do on the conformable time-fractional diffusion equation. In the future work, we shall mainly consider the following two issues. One one hand, whether different sampling strategies will have some good effects on our neural network method or not is a question of interest to us. It may be a good choice to select sampling points clustered where the solution has large gradient. On the other hand, we are more concerned with how to select the best hyper-parameters of the neural network for the conformable time-fractional diffusion equation.

References

- [1] R. Metzler and J. Klafter, The random walk's guide to anomalous diffusion: a fractional dynamics approach, *Phys. Rep.*, 339 (2000), 1-77.
- [2] F. Mainardi, *Fractional Calculus and Waves in Linear Viscoelasticity: an Introduction to Mathematical Models*, World Scientific, River Edge, NJ, 2010.
- [3] D.A. Benson, S.W. Wheatcraft, and M.M. Meerschaert, Application of a fractional advection-dispersion equation, *Water Resources Research.*, 36 (2000), 1403-1412.
- [4] S. Karaa, Semidiscrete finite element analysis of time fractional parabolic problems: a unified approach, *SIAM J. Numer. Anal.*, 56 (2018), 1673-1692.
- [5] H.Y. Zhu and C.J. Xu, A fast high order method for the time-fractional diffusion equation, *SIAM J. Numer. Anal.*, 57 (2019), 2829-2849.
- [6] M. Samiee, M. Zayernouri, and M.M. Meerschaert, A unified spectral method for FPDEs with two-sided derivatives; Part II: Stability, and error analysis, *J. Comput. Phys.*, 385 (2019), 244-261.
- [7] W.N. E, and B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, *Commun. Math. Stat.*, 6 (2018), 1-12.

- [8] R. Maziar, P. Perdikaris, and 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.
- [9] X.W. Jin, S.Z. Cai, H. Li, and G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, *J. Comput. Phys.*, 426 (2021), 109951.
- [10] H.L. Sheng, and C. Yang, PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, *J. Comput. Phys.*, 428 (2021), 110085.
- [11] G.F. Pang, L. Lu, and G.E. Karniadakis, fPINNs: Fractional physics-informed neural networks, *SIAM J. Sci. Comput.*, 41 (2019), A2603–A2626.
- [12] G.F. Pang, M. D’Elia, M. Parks, and G.E. Karniadakis, nPINNs: Nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator, *Algorithms and applications*, *J. Comput. Phys.*, 422 (2020), 109760.
- [13] H.D. Qu, Z.H. She, and X. Liu, Neural network method for solving fractional diffusion equations, *Appl. Math. Comput.*, 391 (2021), 125635.
- [14] R. Khalil, M. Al Horani, A. Yousef, and M. Sababheh, A new definition of fractional derivative, *J. Comput. Appl. Math.*, 264 (2014), 65–70.
- [15] W.S. Chung, Fractional Newton mechanics with conformable fractional derivative, *J. Comput. Appl. Math.*, 290 (2015), 150–158.
- [16] D.R. Anderson, and D.J. Ulness, Properties of the Katugampola fractional derivative with potential application in quantum mechanics, *J. Math. Phys.*, 56 (2015), 063502.
- [17] N. Benkhattou, S. Hassani, and D.F.M. Torres, A conformable fractional calculus on arbitrary time scales, *J. King Saud University-Science.*, 28 (2016), 93–98.
- [18] O.S. Iyiola, O. Tasbozan, A. Kurt, and Y. Çenesiz, On the analytical solutions of the system of conformable time-fractional Robertson equations with 1-D diffusion., *Chaos, Solitons & Fractals.*, 94 (2017), 1–7.
- [19] G. Fernández-Anaya, S. Quezada-García, M.A. Polo-Labarrios, and L.A. Quezada-Téllez, Novel solution to the fractional neutron point kinetic equation using conformable derivatives, *Annals of Nuclear Energy.*, 160 (2021), 108407.
- [20] Y. Çenesiz, A. Kurt, and E. Nane, Stochastic solutions of conformable fractional Cauchy problems, *Statist. Probab. Lett.*, 124 (2017), 126–131.
- [21] A.A. Hyder, and A.H. Soliman, An extended Kudryashov technique for solving stochastic nonlinear models with generalized conformable derivatives, *Commun. Nonlinear Sci. Numer. Simulat.*, 97 (2021), 105730.
- [22] N.I. Mahmudov, and M. Aydın, Representation of solutions of nonhomogeneous conformable fractional delay differential equations, *Chaos, Solitons and Fractals.*, 150 (2021), 111190.
- [23] G.L. Xiao, and J.R. Wang, Representation of solutions of linear conformable delay differential equations, *Appl. Math. Lett.*, 117 (2021), 107088.

- [24] M.A. Bayrak, A. Demir, and E. Ozbilge, A novel approach for the solution of fractional diffusion problems with conformable derivative, *Numer. Methods Partial Differential Eq.*, (2021), 1–18.
- [25] B.K. Singh, S. Agrawal, A new approximation of conformable time fractional partial differential equations with proportional delay, *Appl. Numer. Math.*, 157 (2020), 419–433.
- [26] Lodhi, Sadia, Muhammad Anwaar Manzar, and Muhammad Asif Zahoor Raja, "Fractional neural network models for nonlinear Riccati systems," *Neural Computing and Applications* 31.1 (2019), 359-378.
- [27] Podlubny, Igor. *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*, Elsevier, 1998.