Research Article

Neural Network Method for Solving Time-Fractional Telegraph Equation

Wubshet Ibrahim and Lelisa Kebena Bijiga

Department of Mathematics, Ambo University, Ambo, Ethiopia

Correspondence should be addressed to Wubshet Ibrahim; wubshetib@yahoo.com

Received 7 April 2021; Accepted 11 May 2021; Published 27 May 2021

Academic Editor: Kamal Shah

Copyright © 2021 Wubshet Ibrahim and Lelisa Kebena Bijiga. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Recently, the development of neural network method for solving differential equations has made a remarkable progress for solving fractional differential equations. In this paper, a neural network method is employed to solve time-fractional telegraph equation. The loss function containing initial/boundary conditions with adjustable parameters (weights and biases) is constructed. Also, in this paper, a time-fractional telegraph equation was formulated as an optimization problem. Numerical examples with known analytic solutions including numerical results, their graphs, weights, and biases were also discussed to confirm the accuracy of the method used. Also, the graphical and tabular results were analyzed thoroughly. The mean square errors for different choices of neurons and epochs have been presented in tables along with graphical presentations.

1. Introduction

Fractional differential equations can be used to model many real-life problems. Recently, fractional partial differential equations have received much attention of the researchers due to their wide applications in the area of biological sciences and medicine [1–3]. Moreover, the study conducted in [4, 5] has emphasized on the property of the solution of fractional differential equations like its stability and existence. Particularly, fractional telegraph equations arise in many science and engineering fields such as signal analysis, random walks, wave propagation, electrical transmission line, and so on [6, 7], but they are hard to solve. Accordingly, many methods have been utilized to find solutions of fractional differential equations, for instance, spectral methods [8–10], finite-element method [11, 12], differential transform method [13], and other methods [14, 15]. Moreover, Hosseini et al. [16] had discussed fractional telegraph equation using radial basis function approach. Furthermore, Zhang and Meerschaert and Tadjeran [17, 18] employed a finite-difference approach for a solution of fractional partial differential equation. Recently, solving fractional differential equations by the neural network method has become an active research area.

A neural network is a type of machine learning algorithm which has amazing ability to solve large-scale problems. It is based on the idea of minimizing the loss function that best approximates the solution to mathematical problems. Nowadays, neural networks are becoming the best solution method to most challenging mathematical problems. The continuously rising success of neural network techniques applied to differential equations (ODEs and PDEs) [19, 20] has stimulated research in solving fractional differential equations with the neural network method. Here, this study focuses on solving fractional telegraph equation with the neural network method. As neural network technology is rising rapidly both in terms of methodological and algorithmic developments, we believe that this is a timely contribution that can benefit researchers across a wide range of scientific domains.

Lagaris et al. [21] solved both ordinary and partial differential equations with neural network approaches. The trial solutions have been given which are fixed to satisfy the boundary conditions. Later on, Piscopo et al. [22] have...
extended this trial solution by adding the boundary conditions into a loss function. This paper extends this idea to solving time-fractional telegraph equation. To the researchers’ knowledge, there has been a little study on solving fractional partial differential equations with the neural network approach. In [23], fractional diffusion equation with Legendre’s polynomial-based neural network algorithm has been discussed. Pang et al. [24] proposed fractional physics informed neural networks to solve fractional differential equations. The main contribution of this paper is to discuss the artificial neural network algorithm for solving time-fractional telegraph equations. In this paper, we consider a time-fractional telegraph equation [25]:

\[
\begin{align*}
C D^\alpha_t u(x, t) + C D_0^\alpha u(x, t) &= ku_x(x, t) + f(x, t), \\
u(x, 0) &= \phi_1(x), u_t(x, 0) = \phi_2(x), \\
u(0, t) + \lambda_1 u_x(0, t) &= \mu_1(t), \\
u(L, t) + \lambda_2 u_x(L, t) &= \mu_2(t), \\
0 < t \leq T, 0 < x < L, \frac{1}{2} < \alpha \leq 1.
\end{align*}
\]

The rest of this paper is organized as follows. In Section 2, a short review of fractional calculus is presented. In Section 3, the algorithms for solving equation (1) are given. In Section 4, numerical examples are solved to illustrate the effectiveness of the neural network method. Section 5 gives the conclusion.

2. Preliminaries

Definition 1 (see [26]). The function \( \Gamma : (0, \infty) \rightarrow \mathbb{R} \), defined by

\[
\Gamma(x) = \int_0^\infty t^{-x} e^{-t} dt,
\]

is called Euler’s gamma function (Euler’s integral of the second kind).

Definition 2 (see [27]). Let \( 1 \leq p \).

Definition 3 (see [27]). Let \( u(x) \in L^1_1(a, b) \), and the left-sided and right-sided Riemann–Liouville fractional integrals of order \( \alpha > 0 \) for a function \( u(x) \) on \((a, b)\) are defined as

\[
L_p[a, b] = \left\{ f : [a, b] \rightarrow \mathbb{R}; \quad f \text{ is measurable on } [a, b] \text{ and } \int_a^b |f(x)|^p dx < \infty \right\},
\]

\[
L_\infty[a, b] = \left\{ f : [a, b] \rightarrow \mathbb{R}; \quad f \text{ is measurable and essentially bounded on } [a, b] \right\}.
\]

In other words, \( L_p[a, b] \) is (for \( 1 \leq p < \infty \)) the usual Lebesgue space.

Definition 4 (see [27]). The left-sided and right-sided Riemann–Liouville fractional derivatives of order \( \alpha > 0 \), for a function \( u(x) \) on \((a, b)\), are defined by

\[
D^\alpha_{a+} u(x) = \left( \frac{d}{dx} \right)^n \frac{1}{\Gamma(n-\alpha)} \int_a^x (x-t)^{n-\alpha-1} u(t) dt,
\]

\[
D^\alpha_{b-} u(x) = \left( \frac{d}{dx} \right)^n \frac{1}{\Gamma(n-\alpha)} \int_x^b (t-x)^{n-\alpha-1} u(t) dt,
\]

respectively. Here \( \Gamma(\cdot) \) denotes Euler’s gamma function.

Definition 5 (see [26]). The Caputo fractional derivative \( C D^\alpha_t (u(t)) \) of order \( \alpha \) of a function \( u(t) \) on \((0, T)\) is defined by

\[
D^\alpha_{a+} u(x) = \left( \frac{d}{dx} \right)^n \frac{1}{\Gamma(n-\alpha)} \int_a^x (x-t)^{n-\alpha-1} u(t) dt,
\]

\[
D^\alpha_{b-} u(x) = \left( \frac{d}{dx} \right)^n \frac{1}{\Gamma(n-\alpha)} \int_x^b (t-x)^{n-\alpha-1} u(t) dt,
\]

respectively, where \( n = [\alpha] + 1 \) and \( a < x < b \).
In particular, when \(0 < \alpha < 1\), we have
\[
C^\alpha D_t^n u(t) = \frac{1}{\Gamma(1 - \alpha)} \int_0^t (t - s)^{-\alpha} u'(s) ds.
\] (7)

\[
(-\Delta)^{\alpha/2} u(x_j) = \delta^\alpha_{\Delta x, 1} u(x_j) + O(\Delta x),
\]

\[
(-\Delta)^{\alpha/2} u(x_j) = (1 - \beta) \delta^\alpha_{\Delta x, 1} u(x_j) + \beta \delta^\alpha_{\Delta x, 0} u(x_j) + O((\Delta x)^2),
\]

\[
(-\Delta)^{\alpha/2} u(x_j) = \frac{(11 - 6\beta)(1 - \beta)}{12} \delta^\alpha_{\Delta x, 1} u(x_j) + \frac{-6\beta^2 + 11\beta + 1}{6} \delta^\alpha_{\Delta x, 0} u(x_j) + O((\Delta x)^3).
\] (9)

\[
C D_t^{\alpha/2} u(x, t) + C D_t^{\alpha/2} u(x, t) - f(x, t) = 0,
\]

\[
u(x, 0) - \phi_1(x) = 0,
\]

\[
u(t, x, 0) - \phi_2(x) = 0,
\]

\[
u(0, t) + \lambda_1 u_x(0, t) - \mu_1(t) = 0,
\]

\[
u(L, t) + \lambda_2 u_x(L, t) - \mu_2(t) = 0,
\]

\[0 < t \leq T, 0 < x < L, \frac{1}{2} < \alpha \leq 1.
\] (11)
A numerical solution \( \tilde{u}(x,t) \) to problem (1) is the one which approximately reduces the sum of the mean square of the left-hand side of equation (11), which is similar to the mean square error (MSE) loss function of a neural network. In references [21, 22], the trial solution to ordinary and partial differential equations was presented. Now, we have constructed the trial solution, \( u(x,t) \), for fractional telegraph equation which is the output of the neural network.

If we discretize the domain of the inputs, \((x,t) \in (0,1) \times (0,1)\), into a finite number of training points, say \( m \), which is chosen from equally spaced \( N_m \times N_m \) grid, then \( \tilde{u}(x,t) \) can be obtained by determining the weights and biases that minimize the loss function of the network on the training points.

The following loss function is used:

\[
\xi(w, b, x, t) = \frac{1}{m} \sum_{i=1}^{m} \left( \phi_1(x_i, t_i) - \tilde{u}(x_i, t_i) \right)^2 + \frac{1}{N_m} \sum_{j=1}^{N_m} \left( u(x_j, 0) - \phi_1(x_j) \right)^2 \\
+ \frac{1}{N_m} \sum_{j=1}^{N_m} \left( u(t_j, 0) - \phi_2(x_j) \right)^2 + \frac{1}{N_m} \sum_{j=1}^{N_m} \left( u(0, t_j) + \lambda_1 u_x(0, t_j) - \mu_1(t_j) \right)^2 \\
+ \frac{1}{N_m} \sum_{j=1}^{N_m} \left( u(L, t_j) + \lambda_2 u_x(L, t_j) - \mu_2(t_j) \right)^2.
\]

The optimization can then made via backpropagation by the gradient descent method to find the weights and biases of the network by calculating the gradient of the loss function. The algorithm is implemented in Python 3 all examples were done on Google Colab connected to “Google Compute Engine Backend (GPU),” RAM: 0.80/12.72 GB, Disc: 38.40/68.40 GB.

4. Application

In this section, the neural network method for solving time-fractional telegraph equation is tested using examples.

Example 1. Consider the time-fractional telegraph equation [16]
\[
\begin{cases}
C D^{1-n} u(x, t) + C D^1_t u(x, t) + \nu u_{xx}(x, t) + f(x, t), \\
0 < t \leq 1, 0 < x < 1, 0 \leq \alpha \leq 1, \\
u(0, t) = 0, u_t(0, x) = 0, \\
(15)
u(0, t) = 0, u(1, t) = t^3 (\sin(1))^2.
\end{cases}
\]

where \(f(x, t) = 6(\sin(x))^2((t^{3-n}/(4-\alpha)) + (t^{4-n}/(5-\alpha))+ (\nu t^2)/6) - 2\pi t^2 \cos(2x)\). The exact solution is \(u(x, t) = t^3 (\sin x)^2\). The neural network is trained for 10000, 30000, and 50000 epochs, respectively, with the following hyperparameters: learning rate = 0.001, momentum = 0.99, and domain grid \(N_m \times N_m = 15 \times 15\). The exact and approximated solutions, respectively, are shown in Figure 2, and the network’s weights and biases with \(\alpha = 0.5\), neurons = 50, and epochs = \(5 \times 10^4\) are given below:

\[
w_{\alpha} = [0.22709955 0.48470542 0.30106887 - 0.5710319 - 0.28263545 0.3730778 \\
- 0.62165207 - 0.06325174 0.6000664 - 0.5803307 0.75455767 0.0209149 \\
- 0.3475611 - 0.10525793 1.1346662 0.05548833 0.08919823 - 1.1622386 \\
0.8179567 - 0.1868935 - 0.12351528 - 0.29477507 0.31532037 - 0.14484051 \\
- 0.77425826 - 0.05583666 - 0.70853186 - 0.15046331 0.39233088 - 0.22020972 \\
- 0.42609808 0.3210923 - 1.0803542 1.0820986 0.48231712 1.1275666 \\
- 0.17422408 0.77835166 - 0.338276 - 0.4842986 0.6304897 - 0.22598706 \\
- 0.1556276 - 0.74421215 - 1.0273035 - 0.39799082 - 0.06088707 - 0.10041072 \\
- 0.10655779 - 0.87650526],
\]

\[
w_{\beta} = [-0.8508401 0.34184146 - 1.6948189 - 0.69514495 - 0.09526117 0.10145996 \\
0.67527497 0.3013304 0.13460399 1.253331 - 0.42857054 0.05657079 \\
0.13219093 - 0.4924035 0.5570062 - 0.34572813 - 0.33585164 0.13919753 \\
0.98318714 - 0.46117327 - 0.5684453 - 0.67042255 - 0.26190457 0.24914835 \\
- 0.45094573 - 0.14183266 - 0.98392165 0.41158986 - 0.6243741 - 0.46813807 \\
- 0.31825843 - 0.66352504 0.38143885 - 1.1904637 0.6841169 0.18293405 \\
- 0.01670698 0.09148049 - 0.277907 - 0.12544444 0.01953477 - 0.05040276 \\
- 0.64919883 0.13167143 - 0.5969736 - 0.06241253 - 0.22530134 0.34802237 \\
- 0.4693796 0.78824145],
\]

\[
w_{\gamma} = [-0.27188507 - 0.39588678 - 1.3488795 - 0.11360911 - 0.49185053 0.10852274 \\
0.2896792 0.2204578 0.24846004 0.4356355 1.4058646 0.31145126 \\
0.7254508 - 0.14880532 1.1028032 - 0.205357 - 0.2936519 - 0.9101087 \\
- 0.37215596 - 0.25380853 0.26887977 - 0.07968626 - 0.11693105 0.15452379 \\
0.38469067 - 0.772217 1.8316675 0.27511346 - 0.04076689 0.06723411 \\
0.06209861 - 0.39242387 - 1.135868 0.7484505 - 0.09107573 - 0.6481716 \\
- 0.02400028 0.04516968 0.11897138 - 0.46179143 - 0.6552162 - 0.05483521 \\
- 0.02160627 - 0.09792262 0.7942127 - 0.20355819 - 0.13058168 0.33987236 \\
- 0.13059096 0.66211694],
\]
Table 1 presents the variations of mean square error of Example 1 with different epochs, neurons, and \( \alpha \). The result indicates that as the values of epochs, neurons, and \( \alpha \) increase, the mean square error diminishes. Similarly, the approximate and exact solutions are shown in Figure 3 with values of \( \alpha = 0.7 \), neurons = 50, and epochs = \( 5 \times 10^4 \). Also,

\[
\begin{align*}
\hat{b}_{fh} &= [-0.817806 - 0.16366439 1.7479048 - 0.03289943 0.5446239 - 0.7773745 \\
&- 0.09653217 - 0.27597392 - 0.59627736 - 0.48429355 0.35829368 0.73302686 \\
&- 0.9513404 - 0.4224985 - 1.9109377 - 0.41555068 0.34191132 0.7894791 \\
&- 0.856203 - 0.6832685 - 0.89504695 - 0.96519065 - 1.0105102 0.08612484 \\
&- 0.37744412 - 0.66712581 1.3613127 - 0.46127853 - 0.1396212 - 1.072931 \\
&- 0.24233888 0.6321648 - 0.02284709 1.3586844 0.32123783 0.2007616 \\
&- 0.33574694 0.8309613 - 0.3023889 0.38618526 1.0148876 0.17742322 \\
&- 0.78868747 - 0.10176767 0.5988298 0.0963082 - 0.00688106 - 0.4881537 \\
&- 0.54485315 0.24851702],
\end{align*}
\]

\[
b_{hf} = 0.3600201.
\]

Figure 4 shows the scatter plot of error function for the same parameters of \( \alpha = 0.7 \), neurons = 50, and epochs = \( 5 \times 10^4 \).

Example 2. Consider the time-fractional telegraph equation [25]

\[
\begin{align*}
C_{\alpha}^D_t (u(t)) + C_{\alpha}^D_t (u(t)) &= u_{xx} (x, t) + f (x, t), \\
0 < t \leq 1, 0 < x < 1, &\frac{1}{2} \leq \alpha < 1, \\
u(x, 0) &= \sin (x + 1), u_x (x, 0) = 0, \\
u(0, t) &= \sin (1) (t^3 + 1), u(1, t) + 3 u_x (1, t) = (t^3 + 1) (\sin (2) + 3 \cos (2)),
\end{align*}
\]
Table 1: The comparison of mean square error (MSE) of Example 1 with different epochs and neurons.

| Epoch | Neurons | $\alpha = 0.5$ | $\alpha = 0.7$ | $\alpha = 0.9$ |
|-------|---------|----------------|----------------|----------------|
|       |         | MSE            | MSE            | MSE            |
| 10000 | 30      | 0.000833       | 0.000970       | 0.001802       |
|       | 50      | 0.000329       | 0.000504       | 0.000229       |
|       | 80      | 0.000401       | 0.000323       | 0.000283       |
| 20000 | 30      | 0.000481       | 0.000732       | 0.000858       |
|       | 50      | 0.000105       | 0.000269       | 0.000161       |
|       | 80      | 0.000189       | 0.000187       | 0.000169       |
| 50000 | 30      | 0.000257       | 0.000367       | 0.000166       |
|       | 50      | 0.000034       | 0.000117       | 0.000107       |
|       | 80      | 0.000090       | 0.000094       | 0.000092       |

Figure 3: The exact and approximate solutions of Example 1 for $\alpha = 0.5$ with epoch = 50000.

Figure 4: The scatter plot of error function for Example 1 for $\alpha = 0.7$ and epoch = 50000.
Approximate solution

Figure 5: The exact and approximate solutions of Example 1 for $\alpha = 0.5$ with epoch = 50000.

where $f(x,t) = 6 \sin(x+1)((t^{1-2\alpha} / \Gamma(4 - 2\alpha)) + (t^{1-\alpha} / \Gamma(4 - \alpha))) + \sin(x+1)(t^2 + 1)$. The exact solution is $u(x,t) = (t^2 + 1)\sin(x+1)$. The neural network is trained for 50000 epochs, with the following hyperparameters:

- Learning rate = 0.001, momentum = 0.99, and domain grid $N_m \times N_m = 15 \times 15$. The approximated solution is shown in Figure 5, and the network’s weights and biases with $\alpha = 0.5$ and neurons = 30 are given below:

\[
\begin{align*}
\omega_{,x} &= [0.39832613 1.0388726 0.40763474 - 0.02663939 0.92963773 0.8732149 \\
&- 0.02245052 0.08768917 0.32706422 - 0.46646506 - 0.07493997 - 0.14409448 \\
&- 0.3022759 0.01672059 0.42478934 0.76246777 - 1.2880261 0.27322719 \\
&0.02753602 - 0.08018653 - 0.40032297 - 0.21076477 0.06081868 0.07885141 \\
&0.25435236 - 1.1495655 0.9316741 - 1.9513986 - 0.7625092 - 0.6898152], \\
\omega_{,t} &= [1.1917269 - 0.24062987 - 0.3268083 0.5642846 0.477486 - 0.7187211 \\
&0.0262744 0.19974099 0.0064544 - 0.5323977 - 0.13844271 - 1.0950574 \\
&1.5210402 - 0.18306692 0.19848405 - 0.94518507 - 2.3407142 - 0.33137983 \\
&- 2.5880466 - 0.24242482 1.2827591 - 0.09771743 - 1.2863828 1.0360979 \\
&- 0.13701648 0.35354376 - 0.3998722 2.8061397 - 0.8332252 - 1.9516511], \\
\omega_{,h} &= [1.3249726 0.7116116 - 1.4519405 1.0781193 0.90997136 - 1.0058972 \\
&0.31220737 - 1.6432514 1.7020102 - 0.7055373 0.18836688 - 0.75097185 \\
&2.586712 0.2544526 1.0796609 - 1.5409948 1.23299 - 1.1104541 \\
&0.8619889 - 0.48276564 - 1.5010551 2.3442566 2.3573098 - 0.67585474 \\
&- 0.74543667 0.6313507 - 0.76804924 - 0.47706455 - 2.21064 - 0.7420342], \\
\end{align*}
\]
In Figure 6, the left graph shows the scatter plot of the error function and the right graph shows the scatter plot of the exact solution. This graph is plotted for $50 \times 50$ mesh grid points. The graph reflects that the error between the exact and approximate solutions is exactly similar.

5. Conclusion

The novel way to solve time-fractional telegraph equation is proposed. The method extends the existing approach for differential equations (both ODEs and PDEs) [22] to fractional differential equations. This approach is applied to finding time-fractional telegraph equations for which exact solutions are known. Compared to traditional fractional differential equation (FDE) solvers, such as the finite-difference method and spectral method, which can only obtain the approximate solutions on the grid points, the proposed method can approximate the solutions of all points in the interval by training only few sample points. However, a lot of training is needed to get higher accuracy, especially when $1 < \alpha < 2$. This would be interesting work for future direction.

\begin{equation}
 b_{Jt} = [0.39966443 0.46180362 - 0.28170237 0.30222654 - 0.46586838 - 0.4193143 - 0.28170237 0.30222654 - 0.46586838 - 0.4193143 - 2.553217 - 0.91044205 0.2799298 0.5104434 - 2.3046823 0.17728463 - 0.18914123 0.0158242 - 0.11061913 - 0.26841065 0.8413445 - 0.02756048 0.23803908 1.841479 - 1.3463446 - 3.640385 1.4425491 - 0.7643006],
\end{equation}

\begin{equation}
 b_{Jf} = 1.9012741.
\end{equation}

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors’ Contributions

Wubshet Ibrahim and Lelisa Kebena Bijiga contributed equally to this study.

References

[1] A. Khan, T. Abdeljawad, J. F. Gómez-Aguilar, and H. Khan, “Dynamical study of fractional order mutualism parasitism food web module,” Chaos, Solitons & Fractals, vol. 134, Article ID 109685, 2020.

[2] P. Bedi, A. Kumar, T. Abdeljawad, and A. Khan, “Existence of mild solutions for impulsive neutral Hilfer fractional
evolution equations," Advances in Difference Equations, vol. 2020, pp. 1–16, 2020.

[3] A. Khan, H. M. Alshehri, J. Gómez-Aguilar, Z. A. Khan, and G. Fernández-Anaya, "A predator–prey model involving variable-order fractional differential equations with Mittag-Leffler kernel," Advances in Difference Equations, vol. 2021, no. 1, pp. 1–18, 2021.

[4] A. Khan, H. M. Alshehri, T. Abdeljawad, Q. M. Al-Mdallal, and H. Khan, "Stability analysis of fractional nabla difference COVID-19 model," Results in Physics, vol. 22, Article ID 103888, 2021.

[5] H. Khan, A. Khan, F. Jarad, and A. Shah, "Existence and data dependence theorems for solutions of an ABC-fractional order impulse system," Chaos, Solitons & Fractals, vol. 131, Article ID 109477, 2020.

[6] S. Momani, "Analytic and approximate solutions of the space- and time-fractional telegraph equations," Applied Mathematics and Computation, vol. 170, no. 2, pp. 1126–1134, 2005.

[7] R. C. Cascaval, E. C. Eckstein, C. L. Frota, and J. A. Goldstein, "Fractional telegraph equations," Journal of Mathematical Analysis and Applications, vol. 276, no. 1, pp. 145–159, 2002.

[8] E. H. Doha, A. H. Bhrawy, and S. S. Ezz-Eldien, "Efficient Chebyshev spectral methods for solving multi-term fractional orders differential equations," Applied Mathematical Modelling, vol. 35, no. 12, pp. 5662–5672, 2011.

[9] Y. Lin and C. Xu, "Finite difference/spectral approximations for the time-fractional diffusion equation," Journal of Computational Physics, vol. 225, no. 2, pp. 1533–1552, 2007.

[10] X. Li and C. Xu, "A space–time spectral method for the time fractional diffusion equation," SIAM Journal on Numerical Analysis, vol. 47, no. 3, pp. 2108–2131, 2009.

[11] J. Gao, M. Zhao, N. Du, X. Guo, H. Wang, and J. Zhang, "A finite element method for space-time fractional diffusion partial differential equations in the plane and its error analysis," Journal of Computational and Applied Mathematics, vol. 362, pp. 354–365, 2019.

[12] Y. Jiang and J. Ma, "High-order finite element methods for time-fractional partial differential equations," Journal of Computational and Applied Mathematics, vol. 235, no. 11, pp. 3285–3290, 2011.

[13] A. Arıkoğlu and I. Özkor, "Solution of fractional differential equations by using differential transform method," Chaos, Solitons & Fractals, vol. 34, no. 5, pp. 1473–1481, 2007.

[14] I. Podlubny, Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications, Elsevier, Amsterdam, The Netherlands, 1998.

[15] S. Das, K. Vishal, P. K. Gupta, and A. Yıldırım, "An approximate analytical solution of time-fractional telegraph equation," Applied Mathematics and Computation, vol. 217, no. 18, pp. 7405–7411, 2011.

[16] V. R. Hosseini, W. Chen, and Z. Avazzadeh, "Numerical solution of fractional telegraph equation by using radial basis functions," Engineering Analysis with Boundary Elements, vol. 38, pp. 31–39, 2014.

[17] Y. Zhang, "A finite difference method for fractional partial differential equation," Applied Mathematics and Computation, vol. 215, no. 2, pp. 524–529, 2009.

[18] M. M. Meerschaert and C. Tadjeran, "Finite difference approximations for fractional advection-dispersion flow equations," Journal of Computational and Applied Mathematics, vol. 172, no. 1, pp. 65–77, 2004.

[19] Z. Li, N. Kovachki, K. Azizzadenesheli et al., "Fourier neural operator for parametric partial differential equations," in Proceedings of ICLR Conference 2021, Vienna, Austria, May 2021.

[20] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: a deep learning library for solving differential equations," SIAM Review, vol. 63, no. 1, pp. 208–228, 2020.

[21] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Transactions on Neural Networks, vol. 9, no. 5, pp. 987–1000, 1998.

[22] M. L. Piscopo, M. Spannowsky, and P. Waite, "Solving differential equations with neural networks: applications to the calculation of cosmological phase transitions," Physical Review D, vol. 100, no. 1, Article ID 016002, 2019.

[23] H. Qu, Z. She, and X. Liu, "Neural network method for solving fractional diffusion equations," Applied Mathematics and Computation, vol. 391, Article ID 125635, 2021.

[24] G. Pang, L. Lu, and G. E. Karniadakis, "FPINNs: fractional physics-informed neural networks," SIAM Journal on Scientific Computing, vol. 41, no. 4, pp. A2603–A2626, 2018.

[25] W. Jiang and Y. Lin, "Representation of exact solution for the time-fractional telegraph equation in the reproducing kernel space," Communications in Nonlinear Science and Numerical Simulation, vol. 16, no. 9, pp. 3639–3645, 2011.

[26] A. A. Kilbas, H. M. Srivastava, and J. J. Trujillo, "Theory and applications of fractional differential equations," North-Holland Mathematics Studies, Vol. 204, Elsevier Science Inc., Amsterdam, The Netherlands, 2006.

[27] S. G. Samko, A. A. Kilbas, and O. I. Marichev, Fractional Integrals and Derivatives: Theory and Applications, Gordon and Breach Science Publishers, Philadelphia, PA., USA, 1993.

[28] L. Zhao and W. Deng, "A series of high-order quasi-compact schemes for space fractional diffusion equations based on the superconvergent approximations for fractional derivatives," Numerical Methods for Partial Differential Equations, vol. 31, no. 5, pp. 1345–1381, 2014.

[29] K. Diethelm, "An algorithm for the numerical solution of differential equations of fractional order," Electronic Transactions on Numerical Analysis, vol. 5, pp. 1–6, 1997.

[30] D. Baleanu, K. Diethelm, E. Scalas, and J. J. Trujillo, Fractional Calculus: Models and Numerical Methods, Vol. 3, World Scientific, Singapore, 2012.