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 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]:

\[
\begin{aligned}
C^D_{\alpha} u(x, t) + C^D_{\beta} u(x, t) &= ku_{xx}(x, t) + f(x, t), \\
\phi_1(x), u_t(x, 0) &= \phi_2(x), \\
\mu_1(t), u(0, t) + \lambda_1 u_x(0, t) &= \mu_1(t), \\
\mu_2(t), u(L, t) + \lambda_2 u_x(L, t) &= \mu_2(t), \\
0 < t \leq T, 0 < x < L, \frac{1}{2} < \alpha \leq 1.
\end{aligned}
\]

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 \( \Gamma: (0, \infty) \to \mathbb{R} \), defined by

\[
\Gamma(x) = \int_0^\infty t^{1-x} e^{-t} \, dt,
\]

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

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

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

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

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

Definition 3 (see [27]). Let \( u(x) \in L_1(a, b) \), 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

\[
I^\alpha_a u(x) = \frac{1}{\Gamma(\alpha)} \int_a^x (x - t)^{\alpha-1} u(t) \, dt,
\]

\[
I^\alpha_b u(x) = \frac{1}{\Gamma(\alpha)} \int_x^b (t - x)^{\alpha-1} u(t) \, dt,
\]

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

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

\[
D^\alpha_a u(x) := \left( \frac{d}{dx} \right)^n I^{\alpha-n}_a u(x)
\]

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

\[
D^\alpha_b u(x) := \left( \frac{d}{dx} \right)^n I^{\alpha-n}_b u(x)
\]

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

respectively, where \( n = \lceil \alpha \rceil + 1 \) and \( a < x < b \).

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

\[
C^D_{\alpha} u(t) = D^\alpha_a u(t) - D^\alpha_b u(t).
\]
\[ C D^\alpha_t u(t) = \begin{cases} \int_{0^+}^{t-n} \left( \frac{d}{dr} \right)^n u(t) = \frac{1}{\Gamma(n-\alpha)} \int_0^t (t-s)^{-n-\alpha} u^{(n)}(s)ds, & n-1 < \alpha < n, \\ \left( \frac{d}{dr} \right)^n u(t), & n = \alpha. \end{cases} \]

In particular, when \( 0 < \alpha < 1 \), we have

\[ C D^\alpha_t u(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} u'(s)ds. \]

Definition 6 (see [24]). Grünwald–Letnikov finite-difference schemes: based on the stationary grid \( x_j = (j-1)\Delta x \), for \( j = 1, 2, \ldots, N \), the shifted GL finite-difference operator for approximating the 1D fractional Laplacian is defined as

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

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

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

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

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

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

3. Solution Method

We constructed an artificial neural network (NN) with one hidden layer and \( n \) neurons which is shown in Figure 1. The output of the network, \( \tilde{u}(x,t) \), is written as

\[ \tilde{u}(x,t) = \sum_k g(x \ast w_k + t \ast w_k^f + b_k) \ast w_k^h + b_k^f, \]

where \( g: \mathbb{R}^n \rightarrow \mathbb{R}^n \) is the activation function, \( w = (w_k^x, w_k^h, w_k^f) \) and \( b = (b_k^h, b_k^f) \), \( k = 1, 2, \ldots, n \) are the weights and biases of the network, and a symbol “\( \ast \)” represents normal scalar multiplication, respectively. \( h \) and \( f \) denote hidden and final layers, respectively. Initially, we randomly generated the weights and biases of the network and then we adjusted them in the training processes to minimize the loss function. In short, we find the weights and biases of the neural network that best approximates the solution, \( u(x,t) \), of problem equation (1).
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, \( \tilde{u}(x, t) \), 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.

Let

\[
\varphi(x, t) = C D^\alpha_t u(x, t) + C D_x^\alpha u(x, t) - ku_{xx}(x, t) - f(x, t).
\] (12)

The following loss function is used:

\[
\xi(w, b, x, t) = \frac{1}{m} \sum_{i} \varphi(x_i, t_i)^2 + \frac{1}{N_m} \sum_{j} (u(x_j, 0) - \phi_1(x_j))^2 + \frac{1}{N_m} \sum_{j} (u(0, t_j) + \lambda_1 u_x(0, t_j) - \mu_1(t_j))^2 + \frac{1}{N_m} \sum_{j} (u(L, t_j) + \lambda_2 u_x(L, t_j) - \mu_2(t_j))^2.
\] (13)

The problem is then reduced to minimizing \( \xi(w, b, x, t) \) by adjusting the weights and biases in the network, for the given choice of hyperparameters (number of neurons, number of hidden layers, learning rate, momentum, and activation function), i.e.,

\[
\min_{w, b} \xi(w, b, x, t).
\] (14)

To compute the error, we need to calculate the derivatives of the network output, \( \tilde{u}(x, t) \), with respect to its input. Lagaris et al. [21] have presented how to obtain the automatic differentiation of partial derivatives. To obtain the fractional derivatives, the Grünwald–Letnikov (GL) method given in [28–30] is used. Then, a simple hybrid of the two derivatives is used.

The optimization can then be 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]
where \( f(x,t)=6(\sin(x))^2 ((t^{3-\alpha}/\Gamma(4-\alpha))+ (t^{1-\alpha}/\Gamma(5-\alpha)) + (t/6)) - 2\pi^4 \cos(2x) \). 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\times10^4 are given below:
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,

$$b_{.h} = [-0.817806 \quad -0.16366439 \quad 1.7479048 \quad -0.03289943 \quad 0.5446239 \quad -0.7773745$$

$$\quad 0.09653217 \quad -0.27597392 \quad -0.59627736 \quad -0.48429355 \quad 0.35829368 \quad 0.73302686$$

$$\quad 0.9513404 - 0.4224985 - 1.9109377 - 0.41555068 - 0.34191132 - 0.7894791$$

$$\quad 0.856203 - 0.6832685 - 0.89504695 - 0.96519065 - 1.0105102 - 0.08612484$$

$$\quad -0.37744412 - 0.66712581.3613127 - 0.46127853 - 0.1396212 - 1.072931$$

$$\quad -0.24233888 - 0.6321648 - 0.02284709 - 1.3586844 - 0.32123783 \quad 0.2007616$$

$$\quad 0.33574694 - 0.8309613 - 0.3023889 - 0.38618526 - 1.0148876 \quad 0.17742322$$

$$\quad -0.78868747 - 0.10176767 - 0.5988298 - 0.0963082 - 0.00688106 - 0.4881537$$

$$\quad -0.54485315 - 0.24851702],$$

$$b_{.f} = 0.3600201.$$  \hspace{1cm} (19)

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]

$$\begin{cases}
\mathcal{D}_t^{\alpha} u(t) + \mathcal{D}_t^\beta (u(t)) = u_{xx}(x,t) + f(x,t), \\
0 < t \leq 1, 0 < x < 1, \frac{1}{2} \leq \alpha < 1,
\end{cases}$$

$$u(x,0) = \sin (x + 1), u_t(x,0) = 0,$$

$$u(0,t) = \sin (1)(t^3 + 1), u(1,t) + 3u_x(1,t) = (t^3 + 1)(\sin (2) + 3 \cos (2)).$$

$$\begin{cases}
\mathcal{D}_t^{\alpha} u(t) + \mathcal{D}_t^\beta (u(t)) = u_{xx}(x,t) + f(x,t), \\
0 < t \leq 1, 0 < x < 1, \frac{1}{2} \leq \alpha < 1,
\end{cases}$$

$$u(x,0) = \sin (x + 1), u_t(x,0) = 0,$$

$$u(0,t) = \sin (1)(t^3 + 1), u(1,t) + 3u_x(1,t) = (t^3 + 1)(\sin (2) + 3 \cos (2)).$$

\hspace{1cm} (21)
Table 1: The comparison of mean square error (MSE) of Example 1 with different epochs and neurons.

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Neurons</th>
<th>$\alpha = 0.5$</th>
<th>$\alpha = 0.7$</th>
<th>$\alpha = 0.9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>30</td>
<td>0.000833</td>
<td>0.000970</td>
<td>0.001802</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.000329</td>
<td>0.000504</td>
<td>0.000229</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.000401</td>
<td>0.000323</td>
<td>0.000283</td>
</tr>
<tr>
<td>20000</td>
<td>30</td>
<td>0.000481</td>
<td>0.000732</td>
<td>0.000858</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.000105</td>
<td>0.000269</td>
<td>0.000161</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.000189</td>
<td>0.000187</td>
<td>0.000169</td>
</tr>
<tr>
<td>50000</td>
<td>30</td>
<td>0.000257</td>
<td>0.000367</td>
<td>0.000036</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.000034</td>
<td>0.000117</td>
<td>0.000107</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.000090</td>
<td>0.000094</td>
<td>0.000092</td>
</tr>
</tbody>
</table>

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.
where \( f(x,t) = 6 \sin(x+1)((t^{1-2\alpha}/\Gamma(4 - 2\alpha)) + (t^{1-\alpha}/\Gamma(4 - \alpha))) + \sin(x+1)(t^{1} + 1). \) The exact solution is \( u(x,t) = (t^{1} + 1)\sin(x + 1). \) 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:

\[
\begin{align*}
\omega_{,X} &= \{0.39832613 \ 1.0388726 \ 0.40763474 - 0.02663939 \ 0.92963773 \ 0.8732149 \ 0.02245052 \ 0.08768917 \ 0.32706422 - 0.46646506 \ 0.07493997 - 0.14409448 \ -0.3022759 \ 0.01672059 \ 0.42478934 \ 0.7624677 \ 1.2880261 \ 0.27322719 \ 0.02753602 - 0.08018653 - 0.40032297 - 0.21076477 \ 0.06081868 \ 0.07885141 \ 0.25435236 - 1.1495655 \ 0.9316741 - 1.9513986 - 0.7625092 - 0.6898152],
\omega_{,T} &= \{1.1917269 - 0.24062987 - 0.3268083 \ 0.5642846 \ 0.477486 - 0.7187211 \ 0.02627844 \ 0.19974099 \ 0.0064544 - 0.5323977 \ 0.13844271 - 1.0950574 \ 1.5210402 - 0.18306692 \ 0.19848405 - 0.94518507 - 2.3407142 - 0.33137983 \ -2.5880466 - 0.24242482 \ 1.2827591 - 0.09771743 - 1.2863828 \ 1.0360979 \ -0.13701648 \ 0.35354376 - 0.3998722 \ 2.8061397 - 0.8332252 - 1.9516511],
\omega_{,b} &= \{1.3249726 \ 0.711616 - 1.4519405 \ 1.0781193 \ 0.90997136 - 1.0058972 \ 0.31220737 - 1.6432514 \ 1.7020102 - 0.7055373 \ 0.18836688 - 0.75097185 \ 2.586712 \ 0.2544526 \ 1.0796609 - 1.5409948 \ 1.23299 - 1.1104541 \ 0.8619889 - 0.48276564 \ 1.5010551 \ 2.3442566 \ 2.3573098 - 0.67585474 \ -0.74543667 \ 0.6313507 - 0.76804924 - 0.47706455 - 2.21064 - 0.7420342],
\end{align*}
\]
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 x 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.

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 Kebeha Bijiga contributed equally to this study.

References


