A Multi-Layer Neural Network Approach for Solving Fractional Heat Equations

Amina Ali[1,*], Norazak Senu[2,*] Ali Ahmadian[3]

[2]Universiti Putra Malaysia, Department of Mathematics and Statistics, Selangor, Malaysia
[1] University of Sulaimani, Department of Mathematics, College of Education, Sulaymaniyah, Iraq
[3] Mediterranea University of Reggio Calabria, Decisions Lab, Reggio Calabria, Italy
[3] Lebanese American University, Department of Computer Science and Mathematics, Beirut, Lebanon
[3] Istanbul Okan University, Faculty of Engineering and Natural Sciences, Istanbul, Turkey

Abstract

In this study, a new multi-layer neural network (MLNN) approach designed to solve fractional heat equations (FHEs) is introduced. To handle the fractional derivative, the Laplace transform for approximation was applied. The results of our approach with those obtained using the finite difference method(FDM) are compared. The findings highlight the flexibility and computational efficiency of the proposed approach, making it a promising technique for solving FHEs.

Keywords: Fractional heat equations, Laplace transform, adam optimizer, and neural network.

Nomenclature

x	space
t	time
p	Network's parameter
α	Fractional order
δ	Learning rate

Notations

$\frac{\partial}{\partial x}, \frac{\partial}{\partial t}$	Partial derivative

1. Introduction

In recent years, fractional differential equations(FDE) have attracted significant attention, largely thanks to the rapid advancement of fractional calculus. This growing field has found wide-ranging applications across numerous disciplines, including mathematics, physics, chemistry, biology, medicine, mechanics, control theory, signal and image processing, environmental science, finance, and more[1,2]. Fractional order differential equations can be generalized to the traditional differential equations that involve non-integer orders. They often occur in systems with memory effects, modeled using power-law kernels that capture nonlocal relationships in time and space. These equations become a powerful framework for describing memory and hereditary properties in various materials and processes. Their strong physical foundation has revealed a new topic of scientific research, leading to the development of novel theoretical insights and numerical technique [4,5].

2. Mathematical Framework

In this study, the following FHEs are considered solved through the proposed approach.

$$\frac{\partial^{\alpha} U(x,t)}{\partial t^{\alpha}} - U(x,t)\frac{\partial U(x,t)}{\partial x} - \frac{\partial^2 U(x,t)}{\partial x^2} = F(x,t) \quad (1)$$
$$0 \le x \le 1, 0 \le t \le 1, 0 < \alpha \le 1,$$

with $U(x,0) = g_1(x)$, $U(0,t) = g_2(t)$, and $U(1,t) = g_3(t)$. $\frac{\partial^{\alpha} U(x,t)}{\partial t^{\alpha}}$ can be approximated by Laplace transform as follows:

$$\frac{\partial^{\alpha} U(x,t)}{\partial t^{\alpha}} \approx \alpha \frac{\partial U(x,t)}{\partial t} + (1-\alpha)[U(x,t) - U(x,0)].$$

A trial solution $\hat{U}(X,p)$ is formulated to satisfy the initial boundary value conditions [7], with the goal of solving equation (1)+

$$\hat{U}(X,p) = Q(x,t) + x(1-x)tN(X,p),$$
$$Q(x,t) = (1-x)g_2(t) + xg_3(t) + g_1(x) - (1-x)g_1(0) - xg_1(1)$$

The following expression is used to define the loss function:

$$L(X,p) = \frac{1}{2}\sum_{l=1}^{M}\left[\frac{\partial^{\alpha} U(x,t)}{\partial t^{\alpha}} - U\frac{\partial U}{\partial x} - \frac{\partial^2 U}{\partial x^2} - F(x,t)\right]^2 \quad (2)$$

Here, M denotes the total number of discretized points in both x and t. The Adam optimizer is employed to minimize equation (2). Finally, the model parameters are revised using the following equations:

$$m^k = \zeta_1 m^{k-1} + (1-\zeta_1)\frac{\partial L}{\partial p},$$
$$v^k = \zeta_2 v^{k-1} + (1-\zeta_2)(\frac{\partial L}{\partial p})^2,$$
$$\hat{m}^k = \frac{m^k}{(1-\zeta_1)},$$
$$\hat{v}^k = \frac{v^k}{(1-\zeta_2)},$$
$$p^{k+1} = p^k - \delta \frac{\hat{m}^k}{\sqrt{(\hat{v}^k+\epsilon)}}.$$

2.1 MLNN Model Construction

The neural network architecture employed in this study is most accurately described as a multi-layer neural network (MLNN). As illustrated in Fig. 1, the network is composed of an input layer, four hidden layers, and an output layer. Each hidden layer contains kkk neurons. To maintain clarity in notation, each neuron is labeled using a superscript that indicates its corresponding layer. This convention also extends to the network's weights and biases.The weights are denoted by $w_{kk}^{[l]}$, where l ranges from 1 to 4, representing the connections between layers.The input to the network is represented by the vector $X = (x,t)^T$, and the output, written as $N(x,t,p)$, comes from the final node $N^{[5]}$, , which corresponds to the number of unknowns in the FHEs. The sigmoid activation function is used for all hidden layer neurons and is defined as: $\phi(z) = \frac{1}{1+e^{-z}}$.

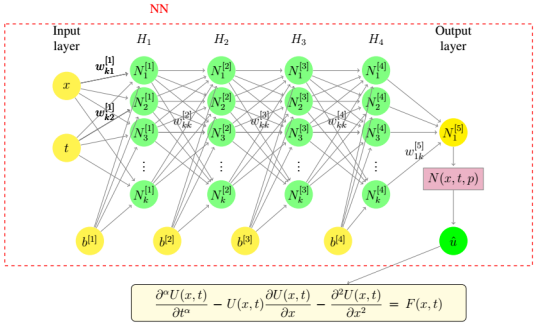

Fig 1. The designed MLNN structure

3. Numerical Results

In this section, the effectiveness of MLNNs in solving FHEs is demonstrated through illustrative examples. The neural network is trained over 10000 iterations using a discretized grid of 121 mesh points within the domain $[0,1] \times [0,1]$.

3.1 Example

Consider the following FHEs, as presented by [6]:

$$\frac{\partial^\alpha U(x,t)}{\partial t^\alpha} - U(x,t)\frac{\partial U(x,t)}{\partial x} - \frac{\partial^2 U(x,t)}{\partial x} = 0,$$
$$0 \le x \le 1, 0 \le t \le 1, 0 < \alpha \le 1,$$

$U(x,0) = 2 - x$, $U(0,t) = \frac{2}{1+t}$, and $U(1,t) = \frac{1}{1+t}$.
The exact solution for $\alpha = 1$ is $U(x,t) = \frac{2-x}{1+t}$.

Table 1: Numerical results obtained using different values of α .

		MLNN		
(x,t)	Exact	$\alpha = 0.1$	$\alpha = 0.8$	$\alpha = 1$
(0,0)	2	2	2	2
(0.1, 0.1)	1.72727	1.67767	1.72727	1.72726
(0.2, 0.2)	1.49999	1.44184	1.49999	1.49998
(0.3, 0.3)	1.30769	1.30767	1.30768	1.307621
(0.4, 0.4)	1.14285	1.25923	1.14285	1.14282
(0.5, 0.5)	1.000000	0.98421	0.99999	0.99996
(0.6, 0.6)	0.87500	0.87355	0.87499	0.87495
(0.7, 0.7)	0.76470	0.77261	0.76469	0.76466
(0.8, 0.8)	0.66666	0.67811	0.66665	0.66662
(0.9, 0.9)	0.57894	0.58792	0.57894	0.57892
(1,1)	0.5	0.5	0.5	0.5
CPU Time(s)		80.86	67.80	76.31

Table 2: Comparison of absolute errors between MLNN and FDM at $\alpha = 1$.

(x,t)	MLNN	FDM
(0.1,0.1)	2.74×10^{-6}	3.80×10^{-3}
(0.2,0.2)	9.89×10^{-6}	7.18×10^{-3}
(0.3,0.3)	1.91×10^{-5}	8.33×10^{-3}
(0.4,0.4)	2.90×10^{-5}	7.89×10^{-3}
(0.5,0.5)	3.79×10^{-5}	6.63×10^{-3}
(0.6 , 0.6)	4.35×10^{-5}	5.09×10^{-3}
(0.7,0.7)	4.46×10^{-5}	3.55×10^{-3}
(0.8, 0.8)	3.87×10^{-5}	2.17×10^{-3}
(0.9, 0.9)	2.45×10^{-5}	9.87×10^{-3}

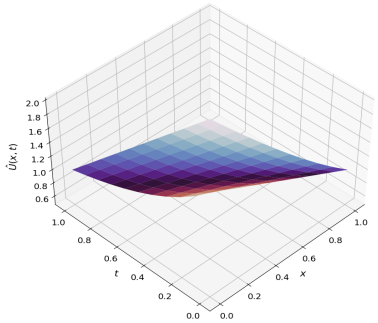

Fig. 2. MLNN solution at $\alpha = 1$

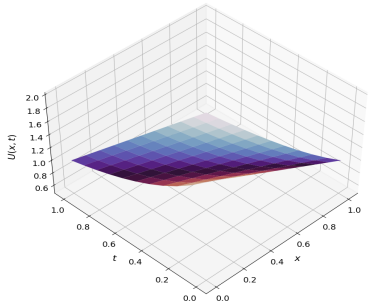

Fig. 3. Exact solution

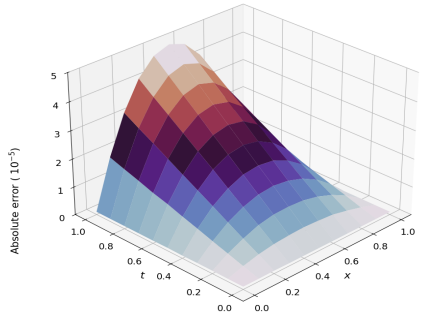

Fig. 4. Absolute error at $\alpha = 1$

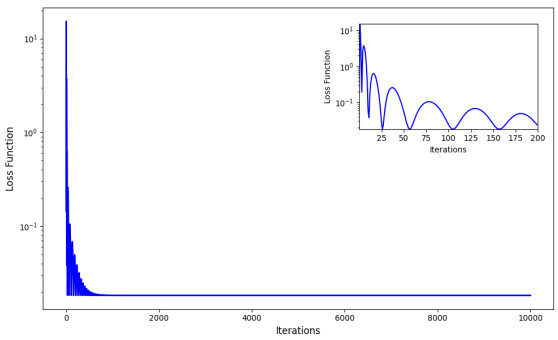

Fig. 5. Convergence of loss function

Table 1 presents the results obtained for different values of α. The results indicate that the accuracy of the MLNN model improves as α approaches 1. Moreover, the CPU time remains consistently low across all tested values of α. Table 2 provides a comparison of the absolute errors between the MLNN and the FDM at α=1. The results demonstrate that the MLNN consistently outperforms the FDM in terms of accuracy. Fig. 2 illustrates the MLNN-generated solutions for α=1, while Fig. 3 presents the corresponding exact solutions. The strong agreement between the two indicates that the MLNN model effectively approximates the exact solution. Fig. 4 presents the absolute errors at α=1, while Fig. 5 shows how the loss function changes over 10,000 iterations. The graph reveals a sharp decrease in the loss at the beginning, which then slowly levels off close to zero as the iterations progress. This pattern indicates that the model successfully converges during training.

4. Conclusion

This study presents a new algorithm that applies MLNNs to solve FHEs. The results show notable improvements over exact solutions. To train the network efficiently, the method uses Adam optimization. The algorithm is built around two key components: accurately approximating fractional derivatives and carefully designing the MLNN structure. Moreover, most network parameters were fine-tuned, which helped reduce the loss function to nearly zero by the end of each training cycle. When compared to traditional techniques like the FDM, the MLNN approach achieved higher accuracy, demonstrating its robustness and effectiveness.

References

[1] Li C and Cai M. Theory and numerical approximations of fractional integrals and derivatives. SIAM, 2019.

[2] Li C and Zeng F Numerical methods for fractional calculus, volume 24. CRC Press, 2015.

[3] Yu J and Feng Y. Group classification of time fractional black-scholes equation with time-dependent coefficients. Fractional Calculus and Applied Analysis, 27(5):2335–2358, 2024.

[4] Yu J and Feng Y. On the generalized time fractional reaction–diffusion equation: Lie symmetries, exact solutions and conservation laws. Chaos, Solitons & Fractals, 182:114855, 2024.

[5] Faheem M, Khan A, and Raza A. A high resolution hermite wavelet technique for solving space–time-fractional partial differential equations. Mathematics and Computers in Simulation, 194:588–609, 2022.

[6] Lagaris IE, Likas A and Fotiadis DI. Artificial neural networks for solving ordinary andpartial differential equations IEEETrans. Neural Networks 9 : 987–1000, 1998.