**Communications in Mathematics and Applications**
Vol. 17, No. 1, pp. 51–64, 2026
ISSN 0975-8607 (online); 0976-5905 (print)
Published by RGN Publications
DOI: 10.26713/cma.v17i1.3600

http://www.rgnpublications.com

Research Article

# Numerical Technique for the Solution of Fractional-Order Black-Scholes Differential Equation Using Neural Network Method

Bushra Ismail[1] and Salisu Ibrahim*[2]

[1] *Department of Mathematics, Harran University, Sanliurfa, Turkey*
[2] *Department of Information Technology, Faculty of Applied Science, Tishk International University, Erbil, Iraq*
***Corresponding author:** ibrahimsalisu46@yahoo.com

**Received:** January 4, 2026 **Revised:** February 25, 2026 **Accepted:** March 12, 2026

**Abstract.** The proposed paper presents a novel result for the *fractional-order Black-Scholes differential equation* (FOBSDE) using *neural network method* (NNM). The proposed approach is constructed using utilizes spectral approximation and shifted Legendre polynomials and neural network optimization. The accuracy and efficiency of the method are demonstrated by numerous numerical experiments, and demonstrate that the method is superior to conventional numerical methods. Such results indicate what neural networks are able to do in solving complex fractional partial differential equations, which is a strong foundation of financial modeling.

**Keywords.** Fractional Partial Differential Equation, Neural Networks, Fractional Black-Scholes, $L^2$-error, Shifted Legendre

**Mathematics Subject Classification (2020).** 35R11, 34B15, 34D20, 34A06, 37Mxx, 65-XX



## 1. Introduction

*Fractional calculus* (FC) has attracted the attention of engineers and physicists. Also, in some neural network models, the fractional computation has been applied. For this reason, the study of *fractional neural networks* (FNN) is crucial for numerous significant outcomes in chaos dynamics, practical applications, and stability analysis (Podlubny [17]).

In recent years, extensive research has been conducted on the *fractional-order Black-Scholes differential equations* (FOBSDEs) across various industries, owing to their broad applicability, including the use of neural network techniques to address the fractional equations of the Black-Scholes model (Black and Scholes [3]) demonstrated the efficacy of a two-layered artificial neural network in solving both fractional and ordinary Black-Scholes partial differential equations (PDEs) through the application of Adam optimization (Zhang *et al*. [21]). In the context of the time fractional Black-Scholes model, scholars have developed an extreme learning machine that uses a Legendre wavelet neural network and proved to have better efficiency (Zhang *et al*. [21]). To address the Black-Scholes equations, they have applied neural networks to real stock option data and found that the short-term predictions were more accurate than the traditional analytical solutions (Maitra *et al*. [16]). It has proposed the *Fractional-Order Black-Scholes-Merton* (FOBSM) model which combines neural networks and the use of fractional calculus to model in more detail the complex dynamics of markets such as tail behavior, memory effects, and volatility clustering. All these researchers have demonstrated how neural network methods may be used to increase flexibility and precision of option pricing under different versions of the Black-Scholes equation (Bajalan and Bajalan [1]). Nevertheless, these models did not receive so much attention till 2012 when students demonstrated impressive results in ImageNet Large Scale Visual Recognition Competition, arousing the interest of people in Neural Networks once again in the context of Artificial Intelligence (Cartea and del-Castillo-Negrete [4]).

Legendre [**?**] also discovered a second-order differential equation as he was investigating gravitational attraction and potential theory about 1782. The solutions of this equation were polynomials, which we today refer to as Legendre polynomials, and they proved fundamental to the solution of the equation of Laplace in spherical coordinates, and the representation of the spherical harmonics in physics (Lichtner-Bajjaoui [13]). Originally, these polynomials were only defined over $[-1,1]$ but by the middle of the 20th century, applied mathematicians and engineers required orthogonal basis functions over finite intervals, such as $[0,1]$, to use in both boundary-value and spectral methods. To satisfy this, researchers transformed the argument of the classical polynomials with a simple linear change of variable and resulting in shifted Legendre polynomials which were written as $L_N(x) = P_N(2x-1)$. This transformation preserves the orthogonality and smoothness of the original polynomials but substitutes domain with $[0,1]$, (or $[0,X]$, with an arbitrary length). This normalization renders them particularly convenient in current computational applications, such as Galerkin and collocation spectral methods, and more recently in equations of fractional order, and spectral hybrids based on neural networks (Santos and Ferreira [19]).

Another innovative use of the principle of pricing accuracy is the *Fractional Order Black-Scholes-Merton* (FOBSM) model that combines the classical Black-Scholes-Merton model with neural networks and a new method of accounting (fractional) (Qu and Liu [18]). This study presents a novel first-value approach to solving fractional different equations, using neural networks of cosine functions. We have obtained numerical solutions to both single and integral fractional differential equations using an iterative training process. The efficiency of such an approach is demonstrated using computer-generated illustrations and numerical findings (Biswas *et al*. [2]). A Legendre wavelet neural network coupled with an extreme learning machine is able to solve the time fractional Black-Scholes model successfully (Karniadakis *et al*. [12]).

In the current research, a two-layer *artificial neural network* (ANN) is developed to address the Black-Scholes partial differential equation (PDE), which is important in both ordinary and fractional orders. This technique is especially useful in pricing various options and solving various partial differential equations in various areas, due to its fault tolerance and fine-tuning features, which enhance its speed, accuracy, and reliability(Yasen [20]).

The research paper is aimed at creating and finding access to a numerical method of solving the *fractional-order Black-Scholes differential equation* (FOBSDE) using neural networks. It uses Caputo derivative fraction and shifted Legendre polynomials to design the neural network solution. Numerical tests can only be applied within a certain range of fractional orders and parameters, which cannot be assured to perform beyond these limits (Zhang *et al*. [22]). It includes the use of fractional calculus to give a more detailed insight into market memory, volatility clustering, and heavy-tailed dynamics. Conversely, neural networks introduce a mesh-free hybrid flexible framework that addresses both the issues of stiffness and complexity in fractional partial differential equations, and is more accurate and efficient than finite-difference or spectral approaches (Ibrahim and Koksal [9]).

Our approch to solve FOBSPDEs using the propose neural network method has not exist in literature, therefore is presentet for the first time. Ibrahim and Koksal have explored more recently commutativity in sixth-order LTVSs (Ibrahim [5], Ibrahim and Abbdulnasir [6]). A new commutative approach was suggested and its properties considered by Ibrahim and Köksal [8]. In addition, decomposition and realization of fourth-order LTVSs were also described and results of simulation presented by Ibrahim and Koksal [10], and Lin *et al*. [14].

This paper is aimed at discussing the solution of the *fractional-order Black-Scholes differential equation* (FOBSDE). In this work, Caputo fractional derivatives and shifted Legendre polynomials are used in the development of the solution in the form of a neural network (Luo *et al*. [15]). In recent years, spawning such variations as *fractional PINNs* (fPINNs) and other deep-learning-driven PDE solvers. These techniques are based on neural network approximation of systems with memory effects, and anomalous diffusion. Indicatively, space-time fractional PINN frameworks have been created to address space-time fractional advection-diffusion equations and to determine the unknown fractional parameters at a high accuracy level. Moreover, a number of enhanced architectures have been introduced to improve the performance of the training and to achieve better accuracy of the solutions to nonlinear and fractional PDE models, such as gradient-enhanced PINNs and convolutional-recurrent neural networks. These findings indicate that recent deep-learning-based PDE solvers offer a significant alternative to classical numerical schemes and are currently a research line worth pursuing in solving complex fractional differential equations that occur in finance, physics, and engineering (Ibrahim and Boulaaras [7]).

The overview of this paper is described below. The definition and properties of fractional derivative and integration were presented in Section 2. Section 3 deals with the basic principles of neural network method. Section 4 presents the findings with examples to support the results. Section 5 of the paper provides the conclusion.

## 2. Mathematical Preliminaries

### 2.1 Fractional Calculus

Let $t_0 \in \mathbb{R}_+ = [0,\infty)$ be the initial time. Let $L_1^{\text{loc}}(J,\mathbb{R}^n,\cdot)$ be the linear space of all locally Lebesgue integrable functions $m : J \to \mathbb{R}^m$, $J \subset \mathbb{R}$. Let $\|\cdot\|$ be a norm in $\mathbb{R}^n$.

**Definition 2.1.** Riemann-Liouville fractional integral of order $p \in (0,1)$ is given by [17],

$$
{}_{t_0}I_t^p m(t) = \frac{1}{\Gamma(p)} \int_{t_0}^t \frac{m(s)}{(t-s)^{1-p}} ds, \quad t > t_0,\ p \in \mathbb{R}^+, \tag{2.1}
$$

where $m \in L_1^{\text{loc}}([t_0,\infty),\mathbb{R})$ and $\Gamma(\cdot)$ is the Gamma function.
This is called by some authors the left Riemann-Liouville fractional integral of order $p$.
Note sometimes the notation ${}_{t_0}D_t^{-p} m(t) = {}_{t_0}I_t^p m(t)$ is used.

**Definition 2.2.** Riemann-Liouville derivative of order $p \in (0,1)$ is given by [17],

$$
{}_{t_0}^{RL}D_t^p m(t) = \frac{d}{dt}({}_{t_0}I_t^{1-q} m(t)) = \frac{1}{\Gamma(1-p)} \frac{d}{dt} \int_{t_0}^t (t-s)^{1-p} m(s) ds, \quad t \ge t_0, \tag{2.2}
$$

where $m \in L_1^{\text{loc}}([t_0,\infty),\mathbb{R})$.
This is sometimes called the left Riemann-Liouville.

**Definition 2.3** ([17])**.** Caputo fractional derivative of order $p \in (0,1)$,

$$
{}_{t_0}^{C}D_t^p m(t) = \frac{1}{\Gamma(1-p)} \int_{t_0}^t (t-s)^{1-p} m'(s) ds, \quad t \ge t_0, \tag{2.3}
$$

where $m \in AC^1([t_0,\infty),\mathbb{R})$.
Note that the Caputo fractional derivative exhibits the following characteristics

$$
D^\alpha C = 0, \quad (C \text{ is constant}). \tag{2.4}
$$

Note that the operator which defines eq. (2.3) is linear because

$$
D^\alpha(\lambda f(t) + \mu g(t)) = \lambda D^\alpha f(t) + \mu D^\alpha g(t), \tag{2.5}
$$

where $\lambda$ and $\mu$ are constants.

### 2.2 Fractional Black-Scholes

#### 2.2.1 Classical Black-Scholes Equation

The Black-Scholes equation is a specific kind of PDE that describes the price $V(S,t)$ of a financial derivative. This price is influenced by the underlying asset price $S$ and the time variable $t$,

$$
\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0. \tag{2.6}
$$

#### 2.2.2 Fractional Black-Scholes Equation

The *Fractional Black-Scholes* (FBS) equation takes the classic Black-Scholes partial differential equation (PDE) and extends the model by incorporating fractional calculus, which deals with derivatives that are not just whole numbers. With this upgrade, one can more readily capture the memory effects and long-range dependence and atypical diffusion patterns that we frequently observe in financial markets- things that the traditional model simply does not exhibit,

$$D_t^\alpha V(S,t)+\frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0, \quad 0<\alpha\le 1. \tag{2.7}$$

## 2.3 Neural Network

The initial process in the way neurons compute information is whereby inputs are collected, every input is multiplied by a weight, thereafter, a bias element is included. The entire process is referred to as the weighted sum or the linear combination. Mathematically it can be formulated as:

$$Q=\sum_{i=1}^{n} w_i x_i+b, \tag{2.8}$$

where:

$Q$ is the weighted sum,

$w_i$ is the weight of $h$ the $i$-th input,

$x_i$ is the $i$-th input to the neuron,

$b$ is the term of bias, which is a special parameter that enables the adjustment of the output and the weighted sum.

The weighted sum is very vital because it is the initial input signal to a neuron prior to non-linear transformations occurring. The process enables the network to do a linear transformation of the inputs so that the importance (or weight) of each input in the neuron output is altered.

## 2.4 Shifted Chebyshev Polynomials

In 1854, the Russian mathematician Pafnuty Chebyshev introduced the nth degree polynomial with a leading coefficient of one, specifically within the interval $[-1,1]$,

$$\frac{1}{2^{m-1}}\cos(m\cos^{-1}(x))=\frac{1}{2^{m-1}}T_m(x). \tag{2.9}$$

The recurrence formula for the *Chebyshev Polynomials* (CPs) is generated as

$$T_{m+1}(x)=2xT_m(x)-T_{m-1}(x),\ T_0(x)=1,\ T_1(x)=x, \quad m=1,2,3,\dots. \tag{2.10}$$

The first four CPs are given by

$$T_0(x)=1,$$
$$T_1(x)=x,$$
$$T_2(x)=2x^2-1,$$
$$T_3(x)=4x^3-3x.$$

Additionally, the CPs are defined within the interval $[0,1]$ by transforming the variable $x$ into the form $x=2t-1$. These are known as shifted CPs and have specific definitions that follow:

$$T_m^*=T_m(2t-1), \quad m=0,1,2,\dots.$$

The first few shifted CP defined in $t\in[0,1]$ are given as

$$T_0(t)=1,$$
$$T_1(t)=2t-1,$$

$$T_2(t) = 1 + 8t^2 - 8t,$$
$$T_4(x) = 32t^3 - 48t^2 + 18t - 1.$$

## 2.5 Shifted Gegenbauer Polynomials

The shifted Gegenbauer polynomials $C_i^{\propto}(z)$ are defined on the finite interval $[0,L]$ as a modified version of the classical Gegenbauer polynomials $C_i^{\propto}(z)$, which originally apply to the interval $[-1,1]$. These polynomials come about through a change of variable,

$$z = \frac{2t}{L} - 1. \tag{2.11}$$

Accordingly, the shifted Gegenbauer polynomial is expressed as

$$C_i^{\propto}(t) = C_i^{\propto}\left(\frac{2t}{L} - 1\right), \quad 0 \le t \le L, \tag{2.12}$$

where $\propto > -\frac{1}{2}$ and $i \in z^+$.

These polynomials satisfy the *recurrence relation*,

$$(i+1)C_{i+1}^{\propto}(t) = 2\left(\frac{2t}{L}\right)(\propto + i)C_i^{\propto}(t) - (2\propto + i - 1)C_{i-1}^{\propto}(t), \quad i = 1,2,\dots$$

and the *orthogonality condition*,

$$\int_0^L C_i^{\propto}(t)C_j^{\propto}(t)w_L^{\propto}(t)dt = \lambda_{L,i}^{\propto}\xi_{ij},$$

where the *shifted weight function* is given by

$$w_L^{\propto}(t) = (Lt - t^2)^{\propto - \frac{1}{2}}$$

and $\lambda_{L,i}^{\propto}$ is the normalization factor.

## 2.6 Legendre Polynomial

On the interval $[1,1]$, the Legendre polynomials are defined by the following recursive equations (Yasen [20]),

$$l_0(x) = 1,$$
$$l_1(x) = x,$$
$$l_{i+1}(x) = \frac{2i+1}{i+1}xl_i(x) - \frac{i}{i+1}l_{i-1}(x), \quad i = 1,2,3,\dots.$$

Legendre polynomial is in the form of analytic and is given as

$$L_i(x) = \sum_{k=0}^{i} \frac{(-1)^k(i+k)!}{(i-k)!}\frac{(1-x)^k}{2^k(k!)^2}, \quad i = 0,1,2,\dots. \tag{2.13}$$

The explicit analytical form of the shifted Legendre polynomial of degree $i$ on the interval $[0,X]$ is

$$L_{X,i}(x) = \sum_{s=0}^{i} \frac{(-1)^{i+s}(i+s)!}{(i-s)!}\frac{x^s}{X^s(s!)^2}, \quad i = 0,1,2,\dots \tag{2.14}$$

with boundary conditions given as

$$L_{X,i}(0) = (-1)^i,$$
$$L_{X,i}(X) = 1.$$

## 3. Material and Method

### 3.1 Neural Network Method

The numerical network method was employed to determine approximate solutions for FOBSDEs. This study included a comparative analysis, contrasting the approximate results with the exact solutions, which helped in creating an error analysis chart. The MAPLE software was utilized to simulate both the exact and approximate solutions.

Here, we shall discuss the workings of NNM to FOBSDE,

$$V^q(t,x) = \sum_{i=0}^{M_t}\sum_{j=0}^{M_x} u^q_{ij} L_{T,i}(t) L_{X,j}(x). \tag{3.1}$$

In Section **??**, we introduced the shifted Legendre polynomials, $L_{T,i}(t)$ and $L_{X,j}(x)$. As a result, the function $V^q(t,x)$ outlined in eq. (2.2) is continuous across the domain $[0,1]\times[0,1]$. The neural networks tweak the weights to minimize the loss function using the unknown weights $u^q_{ij}$, where $i \in \{0,1,\dots,M_x\}$ and $j \in \{0,1,\dots,M_t\}$ which are initially set randomly. The model described in eq. (3.1) optimized by integrating the trial result $V^q(t,x)$ with the unknown weights $u^q_{ij}$, and through a process of iterative training, these unknown weights will be adjusted. The basis functions $L_{T,i}(t)$, and $L_{X,j}(x)$ with spatial derivatives and time derived by (2.14). We will consider

$$T(t) = [L_{T,0}(t), L_{T,1}(t), \dots, L_{T,M_t}(t)],$$

where $L_{T,j}(t)$ is defined in (2.14).

*Case* 1: If $\alpha = 1$, we have

$$\frac{dT(t)}{dt} = \frac{d}{dt}[L_{T,0}(t), L_{T,1}(t), \dots, L_{T,M_t}(t)] = [0, L'_{T,1}(t), \dots, L'_{T,M_t}(t)], \tag{3.2}$$

where

$$L'_{T,j}(t) = \sum_{s=1}^{j} \frac{(-1)^{j+s}(j+s)!}{(j-s)!} \frac{x^{s-1}}{X^s(s!)(s-1)!}, \quad j = 1,2,\dots,M_t. \tag{3.3}$$

*Case* 2: For $\alpha \in (0,1)$, we have

$${}_0^cD_t^\alpha T(t) = [0, L^\alpha_{T,1}(t), \dots, L^\alpha_{T,M_t}(t)], \tag{3.4}$$

where ${}_0^cD_t^\alpha$ is the Caputo derivative and it is defined as

$${}_0^cD_t^\alpha L_{T,j}(t) = L^\alpha_{T,j}(t) = \sum_{s=0}^{j} \frac{(-1)^{j+s}(j+s)!t^{s-\alpha}}{(j-s)!(s!)T^s\Gamma(s+1-\alpha)}, \quad j = 1,2,\dots,M_t. \tag{3.5}$$

Consider the spatial derivatives as,

$$L(x) = [L_{X,0}(x), L_{X,1}(x), \dots, L_{X,M_x}(x)]^T. \tag{3.6}$$

Differentiating $L(x)$ leads to

$$\frac{dL(x)}{dx} = [L'_{X,0}(x), L'_{X,1}(x), \dots, L'_{X,M_x}(x)]^T \tag{3.7}$$

with the 2nd derivative of $L(x)$,

$$\frac{d^2L(x)}{dx^2} = [L''_{X,0}(x), L''_{X,1}(x), \dots, L''_{X,M_x}(x)]^T. \tag{3.8}$$

Regarding $\beta \in (1,2)$, we obtain

$$ {}_0^cD_x^\beta L(x) = [L_{X,0}^\beta(x), L_{X,1}^\beta(x), \dots, L_{X,M_x}^\beta(x)], \tag{3.9}$$

where ${}_0^cD_x^\beta$ is Caputo derivative and it is defined by

$$ {}_0^cD_x^\beta L_{X,i}(x) = \sum_{s=0}^{i} \frac{(-1)^{i+s}(i+s)!x^{s-\beta}}{(i-s)!(s!)X^s\Gamma(s+1-\beta)}. \tag{3.10}$$

The parameter for the training dataset is randomly selected,

$$ t_m = \frac{(m-1)T}{(M_m-1)},\ x_n = \frac{(n-1)X}{(M_n-1)}, \quad m \in \{1,2,\dots,M_m\},\ n \in \{1,2,\dots,M_n\}.$$

Substituting (2.2) into the model (2.1) and using eq. (2.3), the errors $er_{m,n}^q$ at sample point $(tm, Sn)$ for non-initial states $m \in \{1,2,\dots,M_m-1\}$, $n \in \{1,2,\dots,M_n-1\}$ are calculated as

$$\begin{aligned} er_{m,n}^q &= \sum_{i=0}^{M_t}\sum_{j=0}^{M_x} u_{i,j}^q L_{T,i}^\propto(t_m)L_{xj}(x_n) - \sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u_{i,j}^q L_{X,i}^\beta(x_m)L_{T,j}(t_n) \\ &\quad - v\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u_{i,j}^q L_{T,j}(t_m)L_{X,i}(x_n)\right)^\eta \left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u_{i,j}^q L_{T,j}(t_m)L_{X,i}'(x_n)\right) \\ &\quad - \lambda\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u_{i,j}^q L_{T,j}(t_m)L_{X,i}(x_n)\right) + \lambda\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u_{i,j}^q L_{T,j}(t_m)L_{X,i}(x_n)\right)^{\eta+111}. \end{aligned} \tag{3.11}$$

While for initial condition (2.1), $m=1$ and $n \in \{1,2,\dots,M_n\}$ and we have

$$ er_{m,n}^q = \sum_{i=0}^{M_t}\sum_{j=0}^{M_x} u_{i,j}^q L_{T,i}(t_1)L_{X,j}(x_n) - \varphi_1(x_n), \tag{3.12}$$

for boundary condition (2.2) $n = 1s$ and $m \in \{2,\dots,M_m\}$ so that

$$ er_{m,n}^q = \sum_{i=0}^{M_t}\sum_{j=0}^{M_x} u_{i,j}^q L_{T,i}(t_n)L_{x,j}(x_1) - \varphi_2(t_n), \tag{3.13}$$

and boundary condition (2.3) $n = M_n$ and $m \in \{2,\dots,M_m\}$ so that

$$ er_{m,n}^q = \sum_{i=0}^{M_t}\sum_{j=0}^{M_x} u_{i,j}^q L_{T,i}(t_m)L_{X,0}'(x_{M_n}) - \varphi_3(t_n). \tag{3.14}$$

The $q$th error matrix is defined by $E^q = (er_{m,n}^q)_{M_m\times N_n}$. The norm is given by

$$ \|E^q\|_F^2 = \frac{1}{2}\sum_{i=0}^{M_m}\sum_{j=0}^{M_n}(er_{m,n}^q)^2. \tag{3.15}$$

Next, the weight adjustment formula (2.1) given by

$$ u_{i,j}^{q+1} = u_{i,j}^q + \Delta u_{i,j}^q. \tag{3.16}$$

For $q = 0,1,2,\dots,N$, with

$$ \Delta u_{i,j}^q = -\rho\frac{\partial\|E^q\|_F^2}{\partial u_{i,j}^q} = -\rho\sum_{i=0}^{M_m}\sum_{j=0}^{M_n} er_{m,n}^q \frac{\partial er_{m,n}^q}{\partial u_{i,j}^q}.$$

*Case* 1: When $m \in \{2,3,\dots,M_m-1\}$, and $n \in \{2,3,\dots,M_n\}$, we have

$$ \frac{\partial er_{m,n}^q}{\partial u_{i,j}^q} = L_{X,i}(x_n)L_{T,j}^\alpha(t_m) - L_{X,i}^\beta(x_n)L_{T,j}(t_m)$$

$$-v\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L_{X,i}(x_n)L_{T,j}(t_m)\right)^{\eta}(L'_{X,i}(x_n)L_{T,j}(t_m))$$
$$-v\eta\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L_{X,i}(x_n)L_{T,j}(t_m)\right)^{\eta-1}(L_{X,i}(x_n)L_{T,j}(t_m))\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L'_{X,i}(x_n)L_{T,j}(t_m)\right)$$
$$-\lambda(L_{X,i}(x_n)L_{T,j}(t_m))+\lambda(\eta+1)\left(\sum\sum u_{i,j}^{q}L_{X,i}(x_n)L_{T,j}(t_m)\right)^{\eta}(L_{X,i}(x_n)L_{T,j}(t_m)).$$

*Case* 2: When $m=1$, and $n\in\{2,3,\dots,\ M_n\}$, we obtain
$$\frac{\partial er_{1,n}^{q}}{\partial u_{i,j}^{q}}=L_{T,i}(t_1)L_{x,j}(x_n).$$
*Case* 3: When $n=1$, and $m\in\{2,3,\dots,M_m\}$, we have
$$\frac{\partial er_{m,1}^{q}}{\partial u_{i,j}^{q}}=L_{T,i}(t_m)L_{x,j}(x_1).$$
*Case* 4: When $n=M_n$, and $m\in\{2,3,\dots,M_m\}$, we have
$$\frac{\partial er_{m,n}^{q}}{\partial u_{i,j}^{q}}=L_{T,i}(t_m)L_{x,j}(x_{M_n}).$$
Hence,
$$\Delta u_{i,j}^{q}=-\rho\sum_{m=2}^{M_m-1}\sum_{n=2}^{M_n}er_{m,n}^{q}\Big\{L_{X,j}(x_n)L_{T,i}^{\alpha}(t_m)-L_{X,j}^{\beta}(x_n)L_{T,i}(t_m)$$
$$-\nu\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L_{Xj}(x_n)L_{T,i}(t_m)\right)^{\eta}(L'_{X,j}(x_n)L_{T,i}(t_m))$$
$$-\nu\eta\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L_{X,j}(x_n)L_{T,i}(t_m)\right)^{\eta-1}(L_{X,j}(x_n)L_{T,i}(t_m))\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L'_{X,j}(x_n)L_{T,i}(t_m)\right)$$
$$-\lambda(L_{X,j}(x_n)L_{T,i}(t_m))+\lambda(\eta+1)\left(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t}u_{i,j}^{q}L_{X,j}(x_n)L_{T,i}(t_m)\right)^{\eta}(L_{X,j}(x_n)L_{T,i}(t_m))\Big\}$$
$$-\rho\sum_{m=1}^{M_m}er_{m,1}^{q}(L_{X,j}(x_n)L_{T,i}(t_1))-\rho\sum_{n=2}^{M_n}er_{1,n}^{q}(L_{X,j}(x_1)L_{T,i}(t_m))$$
$$-\rho\sum_{n=2}^{M_n}er_{M_m,n}^{q}(L'_{X,j}(x_{M_n})L_{T,i}(t_m)).$$
Initial values of weight, $u_{i,j}^{0}$ for $i=0,1,\dots,M_t$ and $j=0,1,2,\dots,M_x$ The upper limit for training sessions with the neural network's learning rate are represented as is $N$ and $q$, respectively

## 4. Numerical Results

This section aims to validate the proposed method by applying the NNM.

**Example 4.1.** Consider the FOBSPDE as
$${}_{0}^{c}D_t^{\propto}V(x,t)=V_{xx}(x,t)+(k_0-1)V_x(x,t)-k_0V(x,t),\quad t>0,\propto\in(0,1] \tag{4.1}$$

with IC

$$V(x,0)=\max(e^{x}-1,0), \tag{4.2}$$

where

$$k_0=\frac{2r}{\sigma^2} \tag{4.3}$$

and BCs

$$V(0,t)=\max(e^{t}-1,0)\cdot 0+\max(e^{t},0)\cdot(1-0),$$

$$V(0,t)=\max(e^{t},0), \tag{4.4}$$

$$V(l,t)=\max(e^{t}-1,0)E_{\propto}(-k_0t^{\propto})+\max(e^{t},0\cdot(1-E_{\propto}(-k_0t^{\propto}), \tag{4.5}$$

where

$$E_{\propto}(-k_0t^{\propto})=\sum_{k=0}^{\infty}\frac{(-k_0t^{\propto})k}{r(\propto\,k+1)},\quad 0<t<1,\ 0<x<1, \tag{4.6}$$

and

$$E_{\propto}(-k_0)=\sum_{k=0}^{\infty}\frac{(-k_0)^k}{r(\propto\,k+1)},\quad \text{for } l=1. \tag{4.7}$$

By using the neural network method to the example (2.4) of (4.1) with the means of (3.1), we get

$$\begin{aligned}\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}L_{T,i}(t)L_{x,j}(x)=&\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}L_{T,i}(t)\frac{d^2V(t,x)}{dx^2}\\&+(k_0-1)\left[\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}L_{T,i}(t)\frac{dV(t,x)}{dx}\right]\\&-k_0\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}L_{T,i}(t)L_{x,j}(x).\end{aligned} \tag{4.8}$$

By using the neural network method to the example (2.4) of (4.1) with the means of (2.14)-(3.8), we get the next equation.

$$\begin{aligned}&\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}\sum_{s=0}^{i}\frac{(-1)^{i+s}(i+s)!}{(i-s)!}\,\frac{x^s}{X^s(s!)^2}\sum_{s=0}^{i}\frac{(-1)^{i+s}(i+s)!t^{s-\propto}}{(i+s)!s!T^s\Gamma(s+1-\propto)}\\&=\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}\sum_{s=0}^{i}\frac{(-1)^{i+s}(i+s)!}{(i-s)!}\cdot\frac{t^s}{T^s(s!)^2}\sum_{s=2}^{j}\frac{(-1)^{i+s}(j+s)!}{(j+s)!}\cdot\frac{s(s-1)x^{s-2}}{X^s s^2(s-1)^2[(s-2)!]^2}\\&+(k_0-1)\left[\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}\sum_{s=0}^{i}\frac{(-1)^{i+s}(i+s)!}{(i-s)!}\cdot\frac{t^s}{T^s(s!)^2}\sum_{s=1}^{j}\frac{(-1)^{j+s}(j+s)!}{(j+s)!}\cdot\frac{x^{s-1}}{X^s s[(s-1)!]^2}\right]\\&-k_0\sum_{i=0}^{M_t}\sum_{j=0}^{M_x}u_{ij}^{q}\sum_{s=0}^{i}\frac{(-1)^{i+s}(i+s)!}{(i-s)!}\cdot\frac{t^s}{T^s(s!)^2}\sum_{s=0}^{i}\frac{(-1)^{i+s}}{(i-s)!}\cdot\frac{x^s}{X^s(s!)^2}.\end{aligned} \tag{4.9}$$

The estimate of the numerical solutions obtained by using NNM is provided by

$$V(x,t)=\max(\exp(x)-1,0)E_{\propto}(-k_0t^{\propto})+\max(\exp(x),0)(1-E_{\propto}(-k_0t^{\propto}). \tag{4.10}$$

It may be observed from Table 1 and Figure 3 that increasing the number of training sets minimizes the error, thereby demonstrating the higher accuracy of the NNM. Furthermore, Figures 1 and 2 present the exact and numerical solutions, respectively, approximated using the NNM. The graph is a perfect illustration base on the fact that there is a good agreement between the exact and numerical solutions.

**Table 1.** Numerical results of the proposed method, when $r = 0.05$; sigma $= 0.2$

| $\alpha$ | Max Error | Relative $L^2$ Error |
|---|---|---|
| 0.50 | $1.23 \times 10^{-3}$ | $2.87 \times 10^{-4}$ |
| 0.75 | $9.87 \times 10^{-4}$ | $2.35 \times 10^{-4}$ |
| 0.90 | $7.12 \times 10^{-4}$ | $1.80 \times 10^{-4}$ |

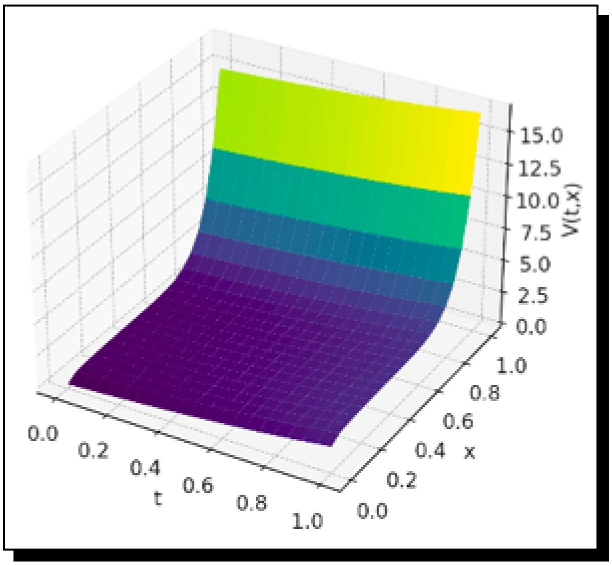


**Figure 1.** Exact solution of $V(t,x)$ over the domain $0 < x < 1$, $0 < t < 1$, and $r = 0.05$; sigma $= 0.2$, $\alpha = 0.5$

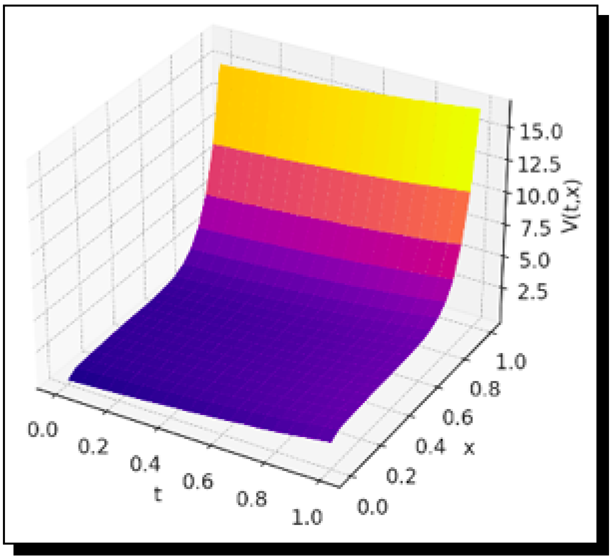


**Figure 2.** Approximate solution ($V_{\text{final}}$) of $V(t,x)$ over the domain $0 < x < 1$, $0 < t < 1$, and $r = 0.05$; sigma $= 0.2$, $\alpha = 0.5$

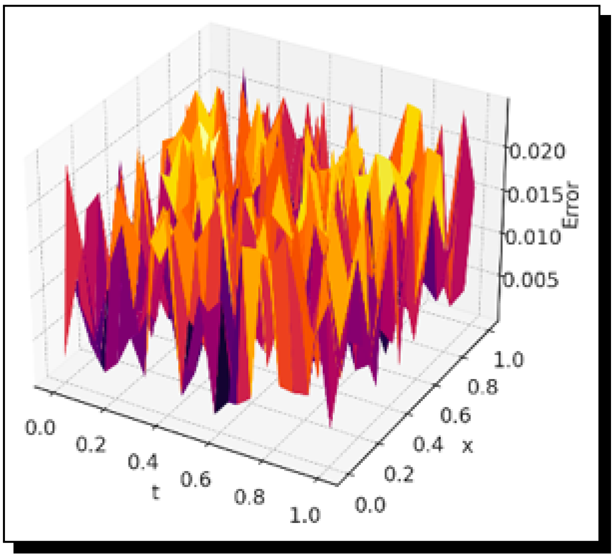


**Figure 3.** Absolute Error [Exact − $V_{\text{final}}$] of $V(t,x)$ over the domain $0 < x < 1$, $0 < t < 1$, and $r = 0.05$; sigma $= 0.2$, $\alpha = 0.5$

## 5. Discussion

The suggested numerical algorithm is a combination of neural network-based approximation and shifted Legendre wavelet spectral representation to the Black-Scholes differential equation of fractional order. The computational complexity of the approach is largely determined from a computational standpoint by how the neural network is trained and how many functioned shifts Legendre wavelet basis of the approximation. The spectral representation allows to reduce the dimensionality of the problem, i.e., the differential equation is mapped into an algebraic optimization problem, hence, the computational cost is less than the grid based numerical schemes, e.g. finite difference or finite element schemes. Nevertheless, the neural network in the training phase still could entail considerable amounts of computation, especially with an increase in the count of hidden neurons, training samples, or the number of training steps. Neural-network-based numerical solvers are also limited by a number of limitations, although they have the benefits. Slow convergence, local minimum of the optimization landscape, and reliance on the quality of initial weights may be problems of the training process. Moreover, the approach is sensitive to multiple hyperparameters, such as the learning rate, network architecture, and the number of hidden layers, and the number of terms in the approximation that are shifted Legendre wavelets. The lack of the correct choice of the parameters can cause the decrease in the accuracy or stability of the solution received. These parameters thus must be carefully tuned in order to attain maximum performance. The comparison with modern machine-learning-based PDE solvers, especially (PINNs) and other deep-learning models is another significant consideration point. Whereas PINNs directly treat the governing differential equations as a loss function and can deal with complex boundary conditions mesh-free, the current approach has the advantage of the spectral accuracy of shifted Legendre wavelets, which enhances smooth solution representations and minimizes the error in approximation. The hybrid neural-network spectral method offers better numerical stability and precision to the fractional Black-Scholes model as compared to purely data-driven neural networks. However, recent progress in deep-learning-driven PDE solvers might present an opportunity to high-dimensional or highly nonlinear financial models, implying that future research might consider hybrid models.

## 6. Conclusion

The given study has managed to elaborate and put into practice a neural net approach to addressing the fractional-order Black-Scholes differential equation. The approach employed made use of altered Legendre polynomials as the basis functions and modified the weights in the network via training it on the basis of gradient-based training and succeeded in the acquisition of the non-local attributes of the fractional derivatives and the boundary conditions. The numbers showed high accuracy of the numerical computation of various fractional orders $(-0.5, -0.75, -0.9)$ and financial variables, and the error margin of the differences was never more than $10^{-3}$. The utility of the method was against the precise solutions that reveal its dependability and high quality as opposed to the traditional techniques. This article signals the paradigm shift of how neural networks can transform the field of using fractions of calculus especially in improving quantitative finance modelling whose application can have literally no restriction in terms of control systems (Ibrahim *et al*. [11]).

### Competing Interests

The authors declare that they have no competing interests.

### Authors' Contributions

All the authors contributed significantly in writing this article. The authors read and approved the final manuscript.

## References


**[1]** S. Bajalan and N. Bajalan, Novel ANN method for solving ordinary and time-fractional Black-Scholes equation, *Complexity* **2021**(1) (2021), 5511396, DOI: 10.1155/2021/5511396.

**[2]** C. Biswas, A. Singh, M. Chopra and S. Das, Study of fractional-order reaction-advection-diffusion equation using neural network method, *Mathematics and Computers in Simulation* **208** (2023), 15 – 27, DOI: 10.1016/j.matcom.2022.12.032.

**[3]** F. Black and M. Scholes, The pricing of options and corporate liabilities, *Journal of Political Economy* **81**(3) (1973), 637 – 654, URL: https://www.jstor.org/stable/1831029.

**[4]** Á. Cartea and D. del-Castillo-Negrete, Fractional diffusion models of option prices in markets with jumps, *Physica A: Statistical Mechanics and its Applications* **374**(2) (2007), 749 – 763, DOI: 10.1016/j.physa.2006.08.071.

**[5]** S. Ibrahim, Commutativity of high-order linear time-varying systems, *Advances in Differential Equations and Control Processes* **27** (2022), 73 – 83, DOI: 10.17654/0974324322013.

**[6]** S. Ibrahim and I. Abbdulnasir, New commutative formulas for second-order linear time-varying systems, *AIP Conference Proceedings* **2554** (2023), 020006, DOI: 10.1063/5.0104208.

**[7]** S. Ibrahim and S. Boulaaras, Commutativity of linear time-varying fractional-order systems, *Asian Journal of Control* **28**(1) (2026), 249 – 257, DOI: 10.1002/asjc.3680.

**[8]** S. Ibrahim and M. E. Köksal, Decomposition of fourth-order linear time-varying systems into its third- and first-order commutative pairs, *Circuits, Systems, and Signal Processing* **42** (2023), 3320 – 3340, DOI: 10.1007/s00034-022-02272-4.

**[9]** S. Ibrahim and M. E. Koksal, Commutativity of sixth-order time-varying linear systems, *Circuits, Systems, and Signal Processing* **40** (2021), 4799 – 4832, DOI: 10.1007/s00034-021-01709-6.

**[10]** S. Ibrahim and M. E. Koksal, Realization of a fourth-order linear time-varying differential system with nonzero initial conditions by cascaded two second-order commutative pairs, *Circuits, Systems, and Signal Processing* **40** (2021), 3107 – 3123, DOI: 10.1007/s00034-020-01617-1.

**[11]** S. Ibrahim, A. Isah, M. Iqbal, P. Chang and D. Baleanu, Commutativity of cascaded connected fractional order linear time-varying systems, *Journal of Circuits, Systems and Computers* **34**(04) (2025), 2550095, DOI: 10.1142/S0218126625500951.

**[12]** G. E. Karniadakis, I.G. E. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, Physics-informed machine learning, *Nature Reviews Physics* **3**(6) (2021), 422 – 440, DOI: 10.1038/s42254-021-00314-5.

**[13]** A. Lichtner-Bajjaoui, *A Mathematical Introduction to Neural Networks*, Master thesis, Universitat de Barcelona, Barcelona, (2021), URL: https://hdl.handle.net/2445/180441.

**[14]** D. Lin, X. Liao, L. Dong, R. Yang, S. S. Yu, H. H.-C. Iu, Experimental study of fractional-order RC circuit model using the Caputo and Caputo-Fabrizio derivatives, *IEEE Transactions on Circuits and Systems I: Regular Papers* **68**(3) (2021), 1034 – 1044, DOI: 10.1109/TCSI.2020.3040556.

**[15]** K. Luo, J. Zhao, Y. Wang, J. Li, J. Wen, J. Liang, H. Soekmadji and S. Liao, Physics-informed neural networks for PDE problems: A comprehensive review, *Artificial Intelligence Review* **58** (2025), article number 323, DOI: 10.1007/s10462-025-11322-7.

**[16]** S. Maitra, V. Mishra, G. K. Kundu and K. Arora, Integration of fractional order Black-Scholes Merton with neural network, in: *15th International Conference on Innovations in Information Technology* (IIT, Al Ain, United Arab Emirates, 2023), pp. 228 – 233 (2023), DOI: 10.1109/IIT59782.2023.10366496.

**[17]** I. Podlubny, *Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of their Solution and some of their Applications*, Vol. 198, xxiv + 340 pages (1999), URL: https://www.sciencedirect.com/bookseries/mathematics-in-science-and-engineering/vol/198/suppl/C.

**[18]** H. Qu and X. Liu, A numerical method for solving fractional differential equations by using neural network, *Advances in Mathematical Physics* **2015**(1) (2015), 439526, DOI: 10.1155/2015/439526.

**[19]** D. de S. Santos and T. A. E. Ferreira, Neural network learning of Black-Scholes equation for option pricing, *Neural Computing and Applications* **37** (2025), 2357 – 2368, DOI: 10.1007/s00521-024-10761-7.

**[20]** B. I. Y. Yasen, *Kesirli Mertebeli Black Scholes Diferansiyel Denklemi İçin Sinir Aği Yöntemi*, Doctoral dissertation, Harran Universitesi, (2025), http://hdl.handle.net/11513/4246. (in Turkish)

**[21]** X. Zhang, J. Yang and Y. Zhao, Numerical solution of time fractional Black-Scholes model based on Legendre wavelet neural network with extreme learning machine, *Fractal and Fractional* **6**(7) (2022), 401, DOI: 10.3390/fractalfract6070401.

**[22]** H. Zhang, M. Zhang, F. Liu amd M. Shen, Review of the fractional Black-Scholes equations and their solution techniques, *Fractal and Fractional* **8**(2) (2024), 101, DOI: 10.3390/fractalfract8020101.