Optimization and Engineering
https://doi.org/10.1007/s11081-024-09910-7

RESEARCH ARTICLE

A Physics informed neural network approach for solving time fractional Black-Scholes partial differential equations

Samuel M. Nuugulu[1] · **Kailash C. Patidar**[2] · **Divine T. Tarla**[2]

Received: 30 April 2024 / Revised: 9 July 2024 / Accepted: 12 July 2024

Abstract
We present a novel approach for solving time fractional Black-Scholes partial differential equations (tfBSPDEs) using Physics Informed Neural Network (PINN) approach. Traditional numerical methods are faced with challenges in solving fractional PDEs due to the non-locality and non-differentiability nature of fractional derivative operators. By leveraging the ideas of Riemann sums and the refinement of tagged partitions of the time domain, we show that fractional derivatives can directly be incorporated into the loss function when applying the PINN approach to solving tfBSPDEs. The approach allows for the simultaneous learning of the underlying process dynamics and the involved fractional derivative operator without a need for the use of numerical discretization of the fractional derivatives. Through some numerical experiments, we demonstrate that, the PINN approach is efficient, accurate and computationally inexpensive particularly when dealing with high frequency and noisy data. This work augments the understanding between advanced mathematical modeling and machine learning techniques, contributing to the body of knowlege on the advancement of accurate derivative pricing models.

Keywords Time-fractional Black-Scholes PDE · Option pricing; Machine learning · Physics Informed Neural Networks · Theoretical data science · Error analysis

1 Introduction

The Black-Scholes equation has long been regarded as the cornerstone of quantitative finance, providing a fundamental framework for pricing financial derivative contracts. Its elegant formulation allows for the calculation of option prices under certain assumptions, making it an indispensable tool in risk management and invest-

✉ Samuel M. Nuugulu
snuugulu@unam.na

[1] Department of Computing, Mathematical and Statistical Sciences, University of Namibia, Private Bag 13301, Windhoek, Namibia

[2] Department of Mathematics and Applied Mathematics, University of the Western Cape, Private Bag X17, Bellville 7535, South Africa

Published online: 02 August 2024

ment strategy formulations. However, the traditional Black-Scholes model assumes that asset returns follow a Gaussian distribution and that the underlying dynamics are described by classical diffusion processes, which may not fully capture the complexities observed for example in high frequency trading stock data. The evolution of chaotic nature of financial markets has lead to a quest for robust methods that can explain stylized fact about markets such as memory effects.

In recent years, the introduction of fractional calculus has revolutionized the modeling of financial systems, offering a more comprehensive description of the underlying assets dynamics. Fractional calculus generalizes the concept of differentiation and integration to non-integer orders, enabling the incorporation of memory and long-range dependencies into mathematical models (Cen et al. 2018; Nuugulu et al. 2021). The tfBSPDE arises when considering the fractional dynamics in the pricing of financial derivatives, and it represents a powerful extension of the classical Black-Scholes partial differential equation (BSPDE). The tfBSPDE introduces fractional derivatives to model anomalous diffusion in financial markets. Fractional calculus extends traditional derivatives to non-integer orders, allowing for a more accurate representation of the long-memory and heavy-tailed characteristics observed in financial time series, see for example (Mainardi 2022; Nuugulu et al. 2021) and references therein.

Solving the tfBSPDE poses significant challenges due to the non-local and non-differen-tiable nature of the fractional operators involved. Conventional numerical methods, such as finite difference schemes, have proven to be effective for solving classical PDEs but may fail in handling fractional derivatives accurately and efficiently. Many scholars such as Chen et al. (2023), Nikan et al. (2024), have tried to improve on the traditional numerical methods by using mesh free approach such as radial basis function (RBF) method when solving fractional PDEs. Although mesh free, RBF approach poses some complication in the choice of the basis function.

On the other hand, PINN have emerged as a promising approach for solving complex PDEs by incorporating physical laws and constraints directly into the learning process (Cai et al. 2021; Pang et al. 2019). The PINN approach incorporates the fractional derivative into the governing PDE to achieve reasonable approximation to the solution. The PINN architecture is equipped with a loss function that enforces both the differential equation and initial and boundary conditions, ensuring accurate solutions even with limited data. The key idea of the PINN methodology is to directly incorporate the fractional derivative into the tfBSPDE while simultaneously employing neural networks to learn the underlying solution. By training the neural networks with both historical financial data and the governing equation, the PINN enforces the physical consistency of the solution and adapts to the complexities inherent in real financial systems. Recently, this idea has been used as a choice over other methods to solve PDEs as found in, Ibrahim et al. (2023), Ibrahim et al. (2023), Jamshaid et al. (2024), Lou et al. (2021), just to mention but a few.

Unlike traditional solution methods such as finite difference that depends purely on the discretization of the PDEs, the PINN approach uses both data and the governing physical equation. Also, the dynamics of the governing PDE and the data is learned in the PINN approach bringing out the relationships between the variables in the problems. Moreover, PINN are nonlinear and can capture nonlinear market dynamics during training that may never be feasible to locate by studying the conver-

gence and consistency properties of the solution of the PDE as is the case when using finite difference methods. PINN adaptively learn the underlying dynamics without needing explicit discretization of the domain, which is a significant advantage over traditional numerical methods that can become computationally expensive in solving for fractional PDEs.

The fractional Brownian motion is a generalization of the standard Brownian motion, incorporating long-term dependencies or memory effects into the stochastic process. The Hurst parameter is a key parameter that determines the level of persistence or memory in the fractional Brownian motion (fBm). The Hurst parameter, α, takes values between 0 and 1, where:

- $0 < \alpha < 0.5$ represent an anti-persistence regime, where positive shocks are followed by negative shocks and vice versa.
- $0.5 \leq \alpha < 1$ represent the persistent regime, where positive shocks tend to be followed by positive shocks and negative shocks by negative shocks.
- $\alpha = 1$ corresponds to a standard Brownian motion, indicating no memory.

For more details on the economic implication of α in each region of $(0, 1]$, see (Nuugulu et al. 2021) and the references therein. In general, the Hurst parameter can be estimated as

$$\alpha = 0.5 + \frac{\log(\frac{R}{\sigma})}{\log(N)}, \tag{1.1}$$

where R is the range of the time series, N is the number of data points and σ is the volatility of the stock prices. In the context of the tfBSPDE, the Hurst parameter characterizes the fractional Brownian motion (fBm) component of the underlying asset's price dynamics. It affects the diffusion term of the PDE, which represents the stochastic component of the underlying asset's evolution. The Hurst parameter determines the memory or autocorrelation structure of the fBm, thereby influencing the price dynamics and the option pricing results derived from the Black-Scholes model, see (Ostaszewicz 2012) for more details.

Due to the presence of the Hurst parameter that is often calculated directly from the available data, according to (1.1), patterns such as persistence, volatility clustering etc, in high frequency and noisy data can be modelled using fractional Black-Scholes equations. Moreover, the tfBSPDE is incorporated into the PINN and data fed into the network for training simultaneously, offering a platform to learn the market dynamics and the relationship in the data. This simultaneous learning of the process dynamics and the fractional derivatives operator increases the accuracy of the predicted solution. This accuracy may significantly deviate from the real value if only the PDE were used as in finite difference methods or if the network were only trained on the available data.

Furthermore, by incorporating a fractional derivative of order α in the time domain of the Black-Scholes PDE, the model accounts for the long-range dependence and memory effects in the underlying asset's price. The Hurst parameter α plays a crucial role in capturing these characteristics and can be estimated from historical financial data or calibrated based on empirical observations.

We take note that, the concept of fractional calculus fits our modeling very consistently because if one look at stock price movements as fractals, i.e., market structures over long timeframes are consistently preserved over smaller timeframes (Mandelbrot and van Ness 1968). There are two very important features of Fractional Brownian motions: Self similarity and the long range dependency (hereditary properties).

Using fractional Brownian motions in fractional models comes with some level mathematical complexity. Jumarie (2009) proposed a solution to this problem by suggesting an approach which considers a non-random fractional stochastic dynamics subject to the usual Brownian motion instead of analogously replacing the governing Brownian motion with a fractional Brownian motion. The models resulting from the Jumarie's idea are generally refered to as fractional partial differential equations (fPDE). There are three classes of fPDE in the Black-Scholes PDE framework, namely: Time fractional Black-Scholes equation (tfBSPDE), space fractional Black-Scholes equation (sfBSPDE) and time-space fractional Black-Scholes equation (tsfBSPDE). In this paper, we will look at a tfBSPDE.

There exist extensive literature on financial derivatives pricing, particularly pricing of European options using finite difference methods has been extensively discussed in the works of Golbabai et al. (2019), Nuugulu et al. (2021), Tagliani and Milev (2013), just to mention but a few. Another notable method used in pricing options is the moving least-square method as applied in Golbabai and Nikan (2020). These approaches are to some extend limited in terms of accuracy, stability and consistency, which further deteriorates with the complexity of the underlying model. More so, with fractional calculus based models, the process of using finite difference methods in solving tfBSPDE becomes even more complicated and involved as there is a need for some post-processing to improve the accuracy of the methods in terms of stability and convergence, see for example (Nuugulu et al. 2021) and the references therein.

In an efffort to address some challenges highlighted in traditional numerical methods, some recent work has emerged. For example in Owoyemi et al. (2020), the idea of the Laplace transform is used to approximate solutions to some fPDEs, although still some numerical approximation techniques are used in the inversion of the transformation from the Laplace domain to the original domain. The work of Owoyemi et al. (2020) suggest that, if the residual function is discretized in order to do away with the fractional order derivatives in the underlying fPDE model, one bypasses the difficulty in performing automatic differentiation on the fractional order derivatives. According to Pang et al. (2019) automatic differentiation is still not possible in most available deep learning frameworks such Tensorflow and Pytoch.

In this paper, we suggest an approach that neither requires taking the Laplace transforms nor apply finite differencing to fPDEs but instead directly incorporates the fractional derivatives into the governing PINN framework and then residual are optimized using the ADAM optimization technique, for details on the technique see for example (Kidger 2022; Lee et al. 2021) and references therein. We futher noe that, the discretization of the residual function allows for the minimization of the loss function at every point in the time domain.

Although the PINN approach has recently emerged as a promising method for solving fractional PDEs it does have its suffer from some challenges, such as, uniqueness of global minimums, scalability, etc. In fact, in Section 4.2 we provide a more elab-

orate account on these challenges as well as highlight on how we have attempted to resolve some of these issues in the current work.

The main contributions of this paper can be summarized as follows: it presents a systematic framework for solving the tfBSPDE using PINN techniques and explains how fractional differentiation is performed to avert the challenges of automatic differentiation of fractional order derivatives in the PINN framework. These addresses the issues of computational and mathematical complexities experienced when using approaches such as finite differencing, see (Nuugulu et al. 2021) as well as those experienced using Laplace transformation techniques, see (Owoyemi et al. 2020; Pang et al. 2019) among others. Secondly, the paper present a way of generating stock data using the fractal stock price dynamics. Thirdly, fractional operators are inherently non-local, the non-locality can be difficult to handle with traditional numerical methods. The PINN approach proposed in the current work utilizes neural networks, which have a universal approximation capability to approximate solutions to differential equations. PINN can easily handle the non-local nature of fractional operators by learning the relationship between different points in the domain. Furthermore, fractional models are non-differentiable in the traditional sense making it impossible to implement PINN in solving such models, since, the backpropagation algorithm in Tensorflow and other deep learning frameworks only works for integer order derivatives. In the current work, we incorporated the fractional derivatives directly into the PINN loss function by using Riemann sums and refined tagged partitions of the time domain, thereby presenting a way of handing non-differentiability of fractional derivatives making it possible to solve tfBSPDE. The effectiveness of the PINN methodology herein is presented via numerical experimentation showcasing its potential in capturing complex financial market dynamics and generating accurate option price predictions.

The rest of this paper is structured as follows: In Sect. 2, we present definitions and preliminaries on fractional calculus. In Sect. 3, we describe the algorithm of computing fractional derivatives under the PINN framework, as well as the architecture of the underlying neural network. Section 4, presents the numerical results, and discusses its implications in the context of option pricing. In Sect. 5, we give the concluding remarks and the scope for further research.

2 Model

This section presents a brief overview of the mathematical preliminaries needed to understand the concepts of fractional calculus implemented in this paper. The section also gives the specification of the tfBSPDE under consideration.

2.1 Preliminaries

In this section, we present the definition and mathematical preliminaries of fractional calculus that are necessary in this paper.

Definition 2.1 Let z be some complex number and let $\Re(z)$ be the real part of z. Then for $\Re(z) > 0$, the Gamma function is defined as

$$\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt. \tag{2.2}$$

Definition 2.2 For some complex numbers z and ω satisfying $\Re(z) > 0$ and $\Re(\omega) > 0$, then the β-function is defined as

$$\beta(z, \omega) = \int_0^1 \tau^{z-1}(1-\tau)^{\omega-1} d\tau. \tag{2.3}$$

Note that the beta function is commutative, i.e., $\beta(z, \omega) = \beta(\omega, z)$. Furthermore,

$$\beta(z, \omega) = \frac{\Gamma(z)\Gamma(\omega)}{\Gamma(z+\omega)}.$$

Definition 2.3 The two parameter Mittag-Leffler function is defined as

$$E_{\iota,\zeta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\iota k + \zeta)}, \ \iota > 0, \ \zeta \in \mathbb{R}. \tag{2.4}$$

Equation (2.4) implies that

$$E_{1,1}(z) = e^z,$$
$$E_{\frac{1}{2},1}(z) = e^{z^2}\mathrm{erfc}(-z),$$

where erf(z), is called the error function defined as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t} dt, \ x \in \mathbb{R},$$

and $\mathrm{erfc}(z) = 1 - \mathrm{erf}(z)$ is the complementary error function.

The m derivative of a two parameter Mittag-Leffler function is defined as

$$E^m_{\iota,\zeta}(z) = \sum_{k=0}^{\infty} \frac{(k+m)!}{k!} \frac{z^k}{\Gamma(k\iota + m\beta + \zeta)}. \tag{2.5}$$

Definition 2.4 Let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a sufficiently smooth, continuous but not necessarily differentiable function on $\mathbb{R}$. Given that $0 < \alpha < 1$, the Caputo fractional derivative of $f(x)$ denoted by $D^\alpha_c f(x)$, $x \in \mathbb{R}$ is defined as

$$D^\alpha_c f(x) = \frac{1}{\Gamma(1-\alpha)} \int_d^x \frac{f'(t)}{(x-t)^\alpha} dt, \tag{2.6}$$

where d is the lower limit of integration, $f'(t)$ denotes the derivative of $f(t)$, and $\Gamma(\cdot)$ represents the gamma function.

If $f(x)$ is not differentiable, then the derivative $f'(t)$ does not exist. However, it is still possible that the integral may have a kernel at the origin. For the derivative of a constant function, say $f(x) = K$ for some constant K, the Caputo fractional derivative is defined as

$$D_c^{\alpha} f(x) = \frac{1}{\Gamma(1-\alpha)} \int_d^x \frac{f'(t)}{(x-t)^{\alpha}} dt = \frac{1}{\Gamma(1-\alpha)} \int_a^x \frac{0}{(x-t)^{\alpha}} dt = 0.$$

Thus, for a constant function $f(x) = K$, the derivative of the Caputo fractional derivative is indeed zero. It is important to note that the behavior of the Caputo fractional derivative depends on the specific properties and conditions of the function being considered. The existence of the Caputo fractional derivative for non-differentiable functions is a subject of ongoing research and may have different interpretations and definitions in different mathematical contexts.

Definition 2.5 Let $f : \mathbb{R} \longmapsto \mathbb{R}$ be a continuous function, but not necessarily differentiable. Then the Reimann-Liouville fractional derivative of order α is given by;

$$D_{RL}^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_0^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau, \tag{2.7}$$

where $n-1 < \alpha \leq n$ and $n \in \mathbb{N}$.

The Riemann-Liouville derivative of a constant is non-zero and the derivative of any function which is constant at the origin, for example the Mittag-Leffler function may have singularities at the origin. Due to the shortcomings in the Riemann-Lioville definition, its applications in modeling a number of nonlinear real life problems have been limited. A modified definition by Jumarie (2006) along with the fractional (Generalized) Taylor series theory, found in Jumarie (2006); Pannas (yyy), addresses the issues of singularities as well the non-zero constraints of constant functions.

The revised Riemann Liouville fractional derivative presented by Jumarie (2007) overcomes the shortcomings of the Riemann Liouville fractional derivatives.

Definition 2.6 Let $f : \mathbb{R} \longmapsto \mathbb{R}$ be a continuous function, but not necessarily differentiable. Then the Jumarie fractional derivative of order α denoted by D_J^{α} is defined under two conditions, the case where f is constant and the case where f is not constant. If $f(x)$ is not a constant function,

$$D_J^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_0^t \frac{f(\tau) - f(0)}{(t-\tau)^{\alpha}} d\tau, \tag{2.8}$$

where $n-1 < \alpha \leq n$ and $n \in \mathbb{N}$. If $f(x) = \mathcal{A}$, where $\mathcal{A}$ is a constant,

$$D_J^{\alpha} f(t) = \begin{cases} \dfrac{\mathcal{A}}{\Gamma(n-\alpha)} t^{1-\alpha-n}, & \alpha \leq n-1, n \in \mathbb{N} \\ 0, & \alpha > n-1, n \in \mathbb{N}. \end{cases} \tag{2.9}$$

Even though the Jumarie definition takes into consideration the existence of fractional derivatives of constants and also explains the derivative at the origin, when considering time fractional derivatives in the context of the Black-Scholes equation, the Caputo definition is often preferred because it allows for a more natural incorporation of initial and boundary conditions, see (Nuugulu et al. 2021, 2023) and references therein. On the other hand, the Caputo fractional derivative considers the history of the function being differentiated, while the Jumarie fractional derivative may not. By using the Caputo definition, one ensure that the fractional derivative operator in the tfBSPDE captures the appropriate behavior of the underlying financial market dynamics. This includes incorporating the initial and the boundary conditions associated with the underlying financial instrument (Podlubny 2001).

In contrast, the Jumarie definition of fractional derivative does not consistently align with initial and boundary conditions. Therefore, when solving tfBSPDEs, the Caputo definition is generally preferred as it provides a more suitable framework for incorporating the necessary conditions and ensuring consistency with the financial context (Nuugulu et al. 2021). Therefore, in this paper the fractional operator is defined in the Caputo sense.

2.2 Model specification

Consider a European put option whose maturity time is T. Let $U(S,t)$ be the price of the option as a function of the stock price S and time $t \in [0,T]$. Let σ be the volatility of the stock and r the risk-free interest rate. Then, the governing classical BSPDE is given by,

$$\frac{\partial U}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 U}{\partial S^2} + rS\frac{\partial U}{\partial S} - rU = 0. \tag{2.10}$$

Using Definition 2.6, Nuugulu et al. (2021), presented the derivation of the time fractional Black-scholes equation under a dividend paying stock as found in Jumarie (2008, 2010). If we consider a non dividend paying stock whose price at time t is given by $S(t)$, then the stock price dynamics under a fractal stochastic process are given by

$$dS(t) = rS(t)dt + \sigma S dB_\alpha(t). \tag{2.11}$$

Using fractional Maruyama representation, $dB_\alpha(t) = B(t)(dt)^{\frac{\alpha}{2}}$, the stock price dynamics can be represented by

$$dS(t) = rS(t)dt + \sigma S B(t)(dt)^{\frac{\alpha}{2}}, \tag{2.12}$$

where $B(t)$ is a standard Brownian motion process. Using stock price dynamics defined by equation (2.12), we arrive at the following tfBSPDE for non-dividend paying stocks

$$U_t^{\alpha} + (rSU_s - rU)\frac{t^{1-\alpha}}{\Gamma(2-\alpha)} + \frac{\Gamma(1+\alpha)}{2}\sigma^2 S^2 U_{SS} = 0, \ \ 0 < \alpha < 1, \tag{2.13}$$

subject to the following initial and boundary conditions

$$\begin{aligned} U(S,T) &= \max(K-S,0), \\ U(0,t) &= Ke^{-rt}, \\ \lim_{S\to\infty} U(S,0) &= 0. \end{aligned} \tag{2.14}$$

Details on the derivation of similar tfBSPDEs can be found in Jumarie (2008, 2010); Kumar et al. (2014); Nuugulu et al. (2021, 2023) just to name but a few.

3 Machine learning algorithm

Automatic differentiation is not defined for fractional order derivatives in Tensorflow. To address this issue, this section presents a methodology for incorporating fractional derivatives into the loss function of PINN models. Consequently, a detailed algorithm for transforming fractional order derivatives into a form that supports automatic differentiation is provided.

3.1 Algorithm for evaluating the time-fractional derivative

Since the backpropagation algorithm within the PINN approach does not support non-integer order derivatives, Pang et al. (2019) proposed a method where fractional PINN employ automatic differentiation to analytically derive the integer-order derivatives of the network's output and then use numerical discretization to approximate fractional derivatives.

Consider the following residual equation

$$\begin{aligned} f(t,x,\alpha,U,U_x,U_{xx}) = \frac{1}{\Gamma(1-\alpha)}\int_0^t \frac{1}{(t-\tau)^{\alpha}} U_\tau d\tau - (rU(x,t) - rSU_x(x,t)) \\ \times \frac{t^{1-\alpha}}{\Gamma(2-\alpha)} + \frac{\Gamma(1+\alpha)}{2}\sigma^2 S^2 U_{xx}(x,t), \end{aligned} \tag{3.15}$$

to compute the residuals in (3.15) within the training process, we compute all the derivatives of the network with respect to the input parameters.

Definition 3.1 Let $\Omega = [0, X) \times [0, T]$ for some real numbers X and T. Let $U(x,t) \in C^{2,1}(\Omega)$, the definite integral of $U(x,t)$ with respect to t is defined as

$$\int_0^t U(x,t)dt := \lim_{n\to\infty} \sum_{i=1}^{n} U(x,t_i)\Delta t, \tag{3.16}$$

for some t_i in the partition of $[0, T]$.

In this context, t_i is taken to be the value of t at the right end points of the rectangles in (3.16), i.e.,

$$t_i = t_0 + i\Delta t,$$

for some uniform step size in time denoted by Δt. Recall equation (2.6), the Caputo derivative of the function $U(x, t)$ of order α with respect to t is defined as

$$D_c^{\alpha} U(x,t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t U_{\tau}(x,\tau)(t-\tau)^{-\alpha} d\tau, \quad 0 \leq \tau \leq t. \tag{3.17}$$

For some number of collocation points N, consider some finite partition of $[0, t]$ given by $0 = t_0 < t_1 < \ldots < t_N = t$. Let $\tau_i \in [t_j, t_{j+1}]$, $j = 0, 1, 2, \ldots, N$; $i = 1, 2, \ldots, N$; where $[t_j, t_{j+1}]$ is a sub-interval of some partition of $[t_0, t]$ and

$$t_j = \tau_0 < \tau_1 < \ldots < \tau_N = t_{j+1},$$

is a partition of $[t_j, t_{j+1}]$ then we have a $N \times (N+1)$ matrix of tine points given by

$$\begin{bmatrix} \tau_0^1 & \tau_1^1 & \ldots & \tau_{N-1}^1 & \tau_N^1 \\ \tau_0^2 & \tau_1^2 & \ldots & \tau_{N-1}^2 & \tau_N^2 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ \tau_0^N & \tau_1^N & \ldots & \tau_{N-1}^N & \tau_N^N \end{bmatrix}.$$

For each $t_k, k = 1, 2, \ldots, N$, the integrand in equation (3.17) is computed as follows

$$\omega_k = \sum_{j=1}^{k} U_{\tau_j}(t_k - \tau_j)^{-\alpha} \Delta\tau. \tag{3.18}$$

Using equation (3.18) the fractional derivative is given as

$$D^{\alpha} U(x,t) = \frac{1}{\Gamma(1-\alpha)} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix}. \tag{3.19}$$

The PINN can be computed by differentiating the $Net(x, t, \tau, P)$ with respect to x, t, τ, and the trainable parameters P (i.e. the weights and the biases) where τ is a secondary input of the model (i.e., an input obtained from another input, in this case from t).

Consider a feed forward neural network with inputs t, x and τ, then the PINN networks' output is defined as

$$\hat{U}(x_{h_n}, t_{h_n}) := Net(x, t, \tau, P^*) = \sum_{h=1}^{m} \sum_{j=1}^{n} \left(\psi_{h_j} g\left(t w_{h_j} + x\eta_{h_j} + \tau\xi_{h_j} + \beta_{h_j}\right)\right), \tag{3.20}$$

where $U(x_{h_n}, t_{h_n})$ is the output of the network at the n^{th} neuron of the h^{th} hidden layer. $P^* = \{\psi, w, \eta, , \xi, \beta, b\}$ is the set of parameters that optimizes $(U - \hat{U})$ obtained using backpropagation algorithm. The function $g(.)$ in (3.20) is the sigmoid activation function as defined below.

The objective is to obtain the set of optimal weights and biases of the network ($P^* = \{\psi, w, \eta, , \xi, \beta, b\}$) which optimizes the loss function. Details of how these weights and biases are updated is presented in Sect. 3.2.

Definition 3.2 Let $g : \mathbb{R} \longrightarrow (0, 1)$ be a continuous and differentiable function on $\mathbb{R}$. The sigmoid activation function g is defined as

$$g(y) = \frac{1}{1 + e^{-y}}, \quad y \in \mathbb{R}. \tag{3.21}$$

For the network in equation (3.20), let the output at each node be

$$y_{h_j} = t w_{h_j} + x\eta_{h_j} + \tau\xi_{h_j} + \beta_{h_j}, \tag{3.22}$$

where w_{h_j} are the weights for the time input into the h^{th} hidden layer and η_{h_j} are the weights for the spatial input into the h^{th} hidden layer, β_{h_j} are the biases at the h^{th} hidden layer with $j = 1, 2, \ldots, n$, and $k = 1, 2 \ldots, m$. Here, n is the number of neurons in each hidden layer and m is the number of hidden layers. Therefore, the output of the function is $g(y) = sigmoid(y)$, where y represent the output from the previous hidden layer.

To solve the minimization problem, we find P^* such that $F(x, t, P^*) = \min F(x, t, P)$, where F represent the cost function and is defined as

$$F(x, t, P) := \frac{1}{2N_x N_t} \sum_{i=1}^{N_x} \sum_{j=1}^{N_t} {E_{i,j}}^2, \tag{3.23}$$

where

$$E_{i,j}(s) := D^{\alpha}\hat{U}(x_j, t_i) - \left(r\hat{U}(x_j, t_i) - r x_j \hat{U}_x(x_j, t_i)\right) \frac{t_i^{1-\alpha}}{\Gamma(2-\alpha)} + \frac{\Gamma(1+\alpha)}{2} \sigma^2 x_j^2 \hat{U}_{xx}(x_j, t_i).$$

Table 1 The MAE for $N = 160$, and $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9, 1\}$

α	MAE	C_t	T_l
0.1	5.7018e−2	7.1288	151.3320
0.3	5.4539e−2	7.2333	148.1973
0.5	3.6191e−2	7.5252	152.0091
0.7	3.5211e−2	7.4258	147.9930
0.9	2.7930e−2	7.0197	140.5297
1.0	3.1157e−2	7.8567	241.0010

Here $j = 1, 2, 3 \ldots, N_r$; $i = 1, 2, 3 \ldots, N_r$, and N_r is the number of collocation points within the domain. We introduce the loss at terminal and boundary points and add to the cost function (3.23) to obtain the total loss (3.24) which is to be optimized. Let the terminal and boundary loss be denoted T_{loss} and B_{loss} respectively, let N_0 and N_b be the number of collocation points at the initial and boundary conditions, then

$$\begin{aligned} T_{\text{loss}} &= \frac{1}{N_0}\left[\sum_{j=0}^{N_0}\left[\left(U(x_{x_j}, t_{N_0}) - \hat{U}(x_j, t_{N_0})\right)\right]^2\right], \\ B_{\text{loss}} &= \frac{1}{N_b}\left[\sum_{i=0}^{N_b}\left[\left(U(x_{N_b}, t_i) - \hat{U}(x_{N_b}, t_i)\right)\right]^2\right], \\ \text{Loss} &= F + T_{\text{loss}} + B_{\text{loss}}. \end{aligned} \tag{3.24}$$

Our aim is to find the appropriate set of parameters P^* such that $Net(x, t, \tau, P^*)$ gives an optimal approximation of $U(x, t)$. Similar ideas can be found in Cervera (2019); Raissi et al. (2017) and the references therein. The procedure for finding P^* is presented in section 3.2.

To capture the accuracy of the PINN approach compared to the standard Black-Scholes model, the maximum absolute error is calculated. This error is defined as the maximum of the absolute values of the differences between the predicted payoff and the payoff from the standard Black-Scholes model at each point in time, i.e.,

$$\text{MAE} = \max(\|\hat{U}_i - U_i\|), \;\; i = 0, 1, 2, \ldots, N; \;\; N \in \{N_0, N_b, N_r\}, \tag{3.25}$$

where $\hat{U}$ is the payoff obtained by solving the tfBSPDE using PINN approach and U is the solution from standard Black-Scholes model. The results from equation (3.25) are shown in tables 1, 3, and 4, while result from the training loss obtained according to equation (3.24) is shown on Fig. 4.

3.2 Forward and backpropagation algorithm

In this section we describe how the network propagate its errors (i.e., the forward and backward propagation). For simplicity, consider a single data points (t, x) and a neural network which has m hidden layers and n neurons per hidden layer. For the

Table 2 Training loss for $N_{train} = 1280$, $\sigma = 0.3$, $r = 0.1$ and $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9, 1\}$

Epochs	Loss, $\alpha = 0.1$	Loss, $\alpha = 0.3$	Loss, $\alpha = 0.5$	Loss, $\alpha = 0.7$	Loss, $\alpha = 0.9$	Loss, $\alpha = 1$
0	1.09798832e+01	1.19249058e+01	1.25250816e+01	1.18544188e+01	2.17458725e+01	1.34371805e+01
300	9.35017364e−04	5.97219774e−03	1.40727106e−02	5.94627252e−03	2.62763118e−03	2.25139456e−03
600	9.68702021e−04	7.46840320e−04	3.94503120e−03	1.23724330e−03	8.43436457e−04	9.03850188e−04
900	5.70623856e−03	2.20597940e−04	1.89723144e−03	2.89412681e−04	4.51690285e−04	4.22074343e−04
1200	9.99799813e−05	1.67053382e−04	1.48838141e−03	1.81885844e−04	3.92782938e−04	3.07780050e−04
1500	9.25773056e−05	1.55658301e−04	1.36003597e−03	1.61142918e−04	3.81733291e−04	2.70470977e−04
1800	8.67273193e−05	1.42977471e−04	1.20567554e−03	1.40838325e−04	3.69109854e−04	2.33172424e−04
2100	8.11529972e−05	1.31437177e−04	1.03140529e−03	1.21810306e−04	3.26659501e−04	2.01713570e−04
2400	7.57626913e−05	1.20428318e−04	7.90397404e−04	1.04611776e−04	3.05765512e−04	1.60164811e−04
2700	7.07569852e−05	1.09754161e−04	4.92489606e−04	8.84896072e−05	2.97092425e−04	1.30031083e−04
3000	6.61506492e−05	9.99281183e−05	3.53457173e−04	7.41404874e−05	2.55896506e−04	1.02242680e−04
3300	6.19586353e−05	5.01633622e−05	2.91835313e−04	6.18081394e−05	2.35476764e−04	7.76931411e−05
3600	5.81689383e−05	2.93245503e−05	2.50534998e−04	5.15028296e−05	2.13851090e−04	5.68869073e−05
3900	5.47039526e−05	2.35172538e−05	2.15896114e−04	4.32629822e−05	2.01679242e−04	4.05302781e−05
4200	5.10430509e−05	2.01887487e−05	1.85272773e−04	3.67303364e−05	1.83811702e−04	2.73850492e−05
4500	4.78323345e−05	1.77086495e−05	1.58209179e−04	3.15148973e−05	1.62242519e−04	1.90052142e−05
4800	4.49484505e−05	1.52599696e−05	1.33556299e−04	2.49746190e−05	1.40182892e−04	1.40540715e−05
5100	4.32647284e−05	1.35862329e−05	1.11840258e−04	1.86803991e−05	1.23483507e−04	1.12750540e−05

Table 3 The MAE for $N_{\text{test}} = \{20, 40, 80, 160, 320\}$, and $\alpha = 0.9$

N	MAE	C_t	Total Loss
20	1.9981e−2	4.8622	153.5527
40	2.7288e−2	5.4489	150.0021
80	3.3170e−2	7.6896	150.9919
160	5.0930e−2	7.9997	154.5297
320	6.0139e−2	11.5350	154.1228

Table 4 The MAE for $N = 150, \alpha = 0.9, \sigma = 0.3, r = 0.05, h_l = \{2, 4, 6, 8\}$, and $n = \{10, 20\}$

h_l/n	MAE	C_t	Total loss
(2,10)	4.3557e−2	5.0794	152.2970
(4,10)	6.5511e−2	6.3794	150.6154
(6,10)	1.7144e−2	7.6142	141.1871
(8,10)	8.6281e−2	10.7191	155.1984
(2,20)	6.7617e−2	11.0577	163.8854
(4,20)	3.9970e−2	12.9145	146.3395
(6,20)	2.6090e−2	15.2042	155.4777
(8,20)	4.9680e−2	21.9830	157.0086

each hidden layer, compute the weighted sum of inputs at each neuron according to equation (3.1). Next, apply the sigmoid activation function on the sum of the outputs at each neuron to obtain the outputs of the hidden layer. We denote this output by a, i.e, for the h hidden layer containing n neurons,

$$a_{h_j} = \text{sigmoid}(y_{h_j}) = \frac{1}{1 + e^{-\sum_{j=1}^{n}\left(w_{h_j} t + \eta_{h_j} x + \xi_{h_j} \tau + b_{h_j}\right)}}. \tag{3.26}$$

For the output layer, we calculate the weighted sum of inputs y for the output neuron:

$$y_o = \sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}, \; j = 1, 2, \ldots, n. \tag{3.27}$$

Apply the sigmoid activation function to obtain the output $\hat{U}$ of the neural network:

$$\hat{U} = g(y_o). \tag{3.28}$$

At this point, we compute the loss function as follow, assume the desired output for the given input example is U. Compute the loss using the Mean Squared Error (MSE) loss function given in equation (3.24), i.e.,

$$\text{Loss} = \min \frac{(\hat{U} - U)^2}{2}. \tag{3.29}$$

We compute the gradients of the loss with respect to the weights and biases

$$\frac{\partial \text{Loss}}{\partial \psi_{h_j}} = a_{h_j} \left[\frac{e^{\left(-\sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}\right)}}{1 + e^{\left(-\sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}\right)}} \right] . \min \left[\hat{U} - U \right],$$
$$\frac{\partial \text{Loss}}{\partial b_{h_j}} = \left[\frac{e^{\left(-\sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}\right)}}{1 + e^{\left(-\sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}\right)}} \right] . \min \left[\hat{U} - U \right]. \tag{3.30}$$

The automatic differentiation within the hidden nodes is performed as

$$\begin{aligned} \frac{\partial \text{Loss}}{\partial w_{h_j}} &= \left(\frac{\partial \text{Loss}}{\partial \hat{U}} \right) \left(\frac{\partial \hat{U}}{\partial a_{h_j}} \right) \left(\frac{\partial a_{h_j}}{\partial w_{h_j}} \right), \\ &= \min \left(U - \hat{U} \right) \left(\frac{\phi_{h_j} e^{\sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}}}{\left[1 + e^{-\sum_{j=1}^{n} \psi_{h_j} a_{h_j} + b_{h_j}} \right]^2} \right) \left(\frac{t e^{-\sum y_{h_j}}}{\left[1 + e^{-y_{h_j}} \right]^2} \right). \end{aligned} \tag{3.31}$$

The partial derivative of the loss with respect to each $\theta_{h_j} \in \{\eta_{h_j}, \xi_{h_j}, b_{h_j}\}$, $h = 1, 2, \ldots, h_l$; $j = 1, 2, \ldots, n$; is evaluated in a similar manner.

For the choice of optimal hyperparameters of our model, the Adaptive Moment Estimation (Adam) optimization technique was used. The Adam optmizer is an optimization technique derived from the gradient descent algorithm. For a dense neural network architecture, Adam has shown to be more efficient as compared to Stochastic Gradient descent or momentum-based algorithms, see for example Kingma and Jimmy (2014), Lee et al. (2021) just to mention but a few. We use this technique to update the weights and biases of the model by adaptively tuning the learning rate for each parameter based on the first and second moments of the gradients of the loss function. These requires initialization of the following parameters: The earning rate l_r, the exponential decay rates for the moment estimates φ and ρ, and a small constant for numerical stability ϵ. Based on literature, see for example Kidger (2022), You et al. (2019) the references therein. The parameters φ and ρ are usually optimal at 0.9 and 0.999, respectively, while ϵ is usually taken as 10^{-8}. Additionally, this algorithm initializes the first moment vector m_t, the second moment vector v_t and the timestep t at zero, i.e., $m_0 = 0$, $v_0 = 0$, $t = 0$.

The steps in the Adam optimization algorithm are as follows: For each iteration of increment, set timestep counter as

$$t = t + 1.$$

For each hyperparameter $\theta_t \in P = \{\psi, w, \eta, , \xi, \beta, b\}$, compute the gradients $Loss_\theta$ of the objective function with respect to the parameter θ_t at time t as

$$\text{Loss}_{\theta_{t_j}} = \frac{\partial \text{Loss}_{t_j}}{\partial \theta_{t_j}}, \ \ j = 1, 2, \ldots, N; N \in \{N_0, N_b, N_r\}. \tag{3.32}$$

Update the biased first moment estimate as

$$m_{t_j} = \varphi_{t_j} m_{t_{j-1}} + (1 - \varphi_{t_j}) Loss_{\theta_{t_j}}. \tag{3.33}$$

Update the biased second moment estimate as

$$v_{t_j} = \rho_{t_j} v_{t_{j-1}} + (1 - \rho_{t_j})(Loss_{\theta_{t_j}})^2. \tag{3.34}$$

Compute bias-corrected first moment estimate as

$$\hat{m}_{t_j} = \frac{m_{t_j}}{1 - \varphi^t_{t_j}}. \tag{3.35}$$

Compute bias-corrected second moment estimate as

$$\hat{v}_{t_j} = \frac{v_{t_j}}{1 - \rho^t_{t_j}}. \tag{3.36}$$

Finally, we update the parameter as

$$\theta_{t_{j+1}} = \theta_{t_j} - l_r \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}. \tag{3.37}$$

The parameters are updated by moving them in the direction of the gradient. The step size is scaled by the learning rate l_r and adjusted according to the bias-corrected first and second moment estimates.

The weights are initialized using normal or uniform distribution, this is done to ensure that the variance of the distributions of the weights are scaled correctly across layers. Suppose n_{h_j} represent the umber of neurons in the current layer and $n_{h_{j+1}}$ are the number of neurons in the next layer $j + 1$, the initial weights are sampled from a uniform distribution given by

$$W_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{h_j} + n_{h_{j+1}}}}, \sqrt{\frac{6}{n_{h_j} + n_{i+1}}}\right), \tag{3.38}$$

or a normal distribution given by

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{n_{h_j} + n_{h_{j+1}}}\right). \tag{3.39}$$

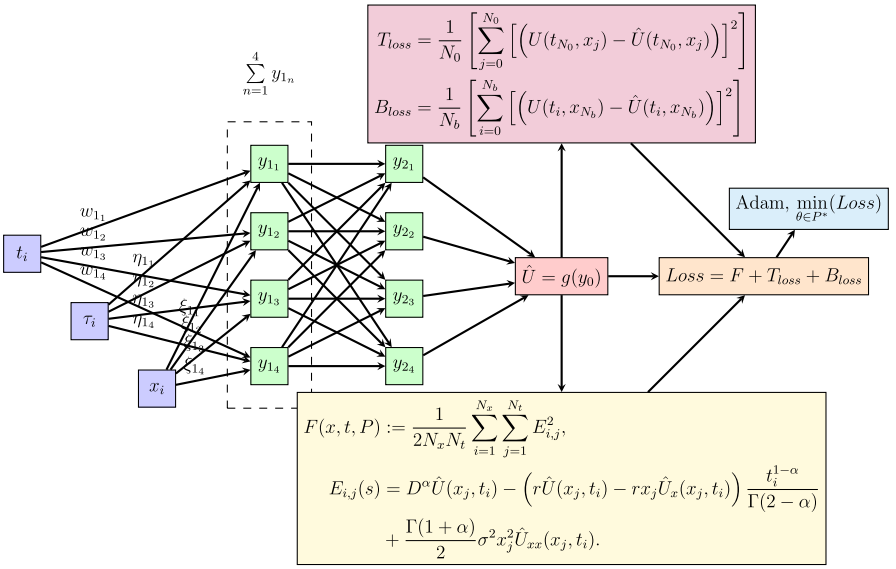


Fig. 1 PINN structure for solving the tfBSPDE with 2 hidden layers, 4 neurons per hidden layer and $i = 0, 1, 2, \ldots, N$; $N \in \{N_0, N_b, N_r\}$

The biases are initialized at zero.

Figure 1 presents the general structure of PINN according to the notation used in this paper. For visual clarity, we have used 2 hidden layers and 4 neurons per hidden layer.

4 Numerical experiments

In this section, we present the data generation process, simulations, some challenges faced by PINN approach in solving the tfBSPDE and the approach employed to optimize performance of the method.

Remark 4.1 All simulations in this paper were performed on a Windows 10 Pro Version $22H2, 12th$ Gen Intel(R) Core(TM) $i7-1255U,\ 1.70GHz$ processor, 16 GB RAM and a 64-bit operating system. We used python version 3.11.5 as the computing software and Tensorflow version 1.16.1 as the deep learning framework.

4.1 Data generation and simulations

The data is obtained as follows, first the bounds for the data are created, i.e.,

$$\begin{aligned}\text{lowerbound}(lb) &:= (t_{min}, x_{min}) = (0, 0),\\ \text{upperbound}(ub) &:= (t_{max}, x_{max}) = (1, 20).\end{aligned}$$

Using the time to maturity $T = 1$, we generate the time data at the initial, boundary and interior as follows. Let N_0 and N_b and N_r be some positive integers representing the maximum number of data points in time at the initial, boundary and interior of the domain respectively. Let $\Delta t_0 = T/N_0$, $\Delta t_b = T/N_b$, $\Delta t_r = T/N_r$, be the step-sizes in time at initial, boundary and interior of the domain respectively. Let

$$\begin{aligned} t_{0_i} &= i\,\Delta t_0,\ 0 = t_{0_i} < t_1 < \ldots < t_{N_0}, \\ t_{b_i} &= i\,\Delta t_b,\ 0 = t_{b_i} < t_1 < \ldots < t_{N_b}, \\ t_{r_i} &= i\,\Delta t_r,\ 0 = t_{r_i} < t_1 < \ldots < t_{N_r}, \text{ where } i = 0, 1, 2, \ldots, N, \\ & \quad N \in \{N_0; N_b; N_r\};\ \text{respectively.} \end{aligned}$$

For each t_k, $k = 1, 2, \ldots, N$, $N \in \{N_0, N_b, N_r\}$, we obtained $\tau_j \in \left[t_k, t_{k+1}\right]$, $j = 1, 2, \ldots, , N$, where $t_K = \tau_1 < \tau_2 < \ldots < \tau_N = t_{k+1}$. The data for stock variable x is generated as follows; It the initial, for $i = 0, 1, 2, \ldots, , N_0$,

$$x_{0_{i+1}} = x_{0_i},\ \forall i : 0 \le i \le N_0,\ i \in \mathbb{N}. \tag{4.40}$$

At the boundary, for each $i = 0, 1, 2, \ldots, N_b$,

$$x_{b_i} = x_0^b \exp\left(r t_{b_i} + \sigma(\Delta t_b)^\alpha dB_b(t,\alpha) - \frac{\sigma^2}{2} t_{b_i}^{2\alpha}\right), \tag{4.41}$$

where

$$x_0^b = \frac{1}{N_b}\sum_{i=1}^{N_b} x_i^b,\ x_i^b = x_i \frac{(x_{max} - x_{min})}{N_b}. \tag{4.42}$$

At the interior of the domain, for $i = 0, 1, 2, \ldots, N_r$,

$$x_{r_i} = x_0^r \exp\left(r t_{r_i} + \sigma(\Delta t_r)^\alpha dB_r(t,\alpha) - \frac{\sigma^2}{2} t_{r_i}^{2\alpha}\right), \tag{4.43}$$

where

$$x_0^r = \frac{1}{N_r}\sum_{i=1}^{N_r} x_i^r,\ x_i^r = x_i \frac{(x_{max} - x_{min})}{N_r}. \tag{4.44}$$

The quantities $dB_b(t,\alpha)$ and $dB_r(t,\alpha)$ are vectors of length N_b and N_r, respectively, representing the fractional Brownian motions at the boundary and the interior of the domain. The entries of these vectors are numbers in the interval [0, 2], chosen so that x_b and x_r belong to $[x_{min}, x_{max}]$, i.e., the solution set of solving $x_{min} \le x_i \le x_{max}$ for $dB(t,\alpha)$ is [0, 2]. With the vectors $\{t_0, t_b, t_r, x_0, x_b, x_r\}$ generated, the initial and boundary conditions become

$$U(x_0, t_0) = \max\left(K e^{-r t_0} - x_0, 0\right),$$

$$U(x_b, t_b) = 0.$$

The initial and boundary data is then concatenated to form a matrix X_{data} given by

$$(X_{\text{data}})_{N_b \times 2}([0, 2]) = [x_0,\ x_b]\,. \tag{4.45}$$

Similarly, for the value of payoff at initial and boundary conditions,

$$(U_{\text{data}})_{N_b \times 2} = [U_0,\ U_b]\,. \tag{4.46}$$

Finally in the interior,

$$(X_{\text{data}})_{N_r \times 2}([0, 2]) = [t_r,\ x_r]\,. \tag{4.47}$$

Note that due to concatenation, there must be equality between N_0 and N_b to avoid mismatch in dimensions. Under this framework, the data set is

$$\text{dataset} = [t; x; \tau]\,,$$

where

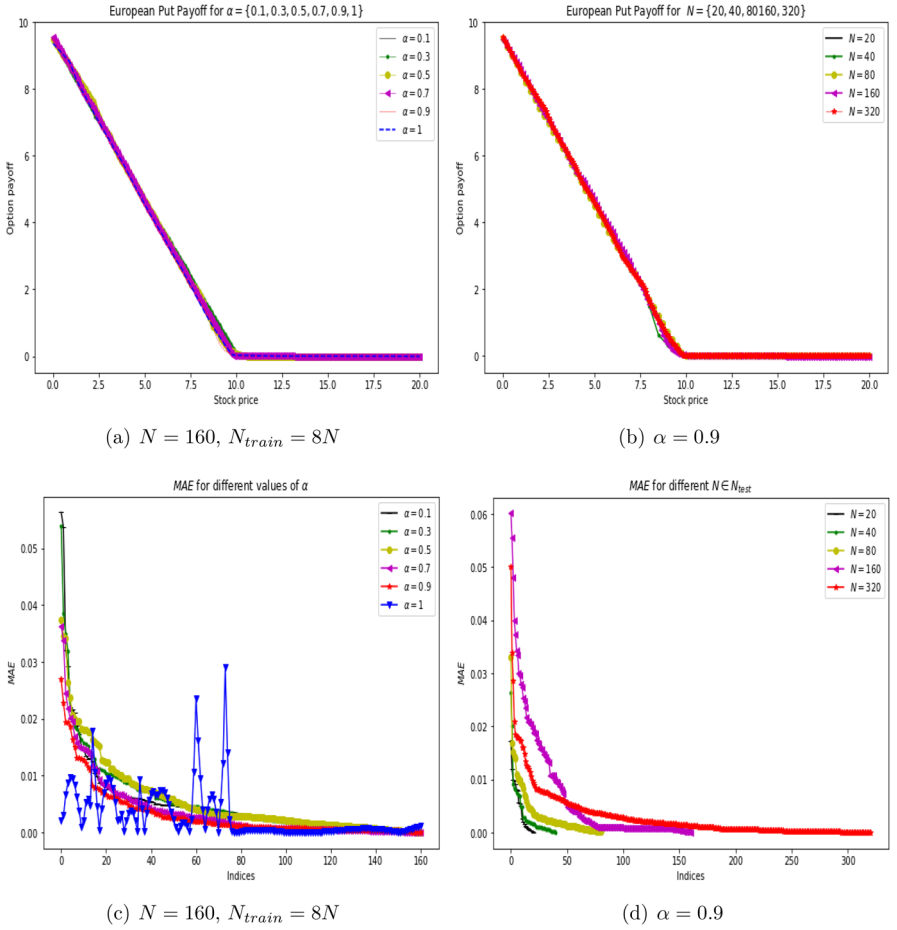
$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_N \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}, \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \vdots \\ \tau_N \end{bmatrix}.$$

In the remainder of this section, we present some numerical examples on solving equation (2.13) using the PINN approach.

Unless otherwise stated, all numerical examples and experiments presented in this section are based on equation (2.13), subject to the initial and boundary condition (2.14). The parameters are: learning rate, l_r = 1e-02, l_r = 1e-03, l_r = 5e-04, which decay in that order after every 1000 epochs; $x \in [0, 20]$, $t \in [0, 1]$, r is varied over $\{0.05, 0.075, 0.1, 0.125, 0.15\}$, $K = 10$, $\sigma = \{0.15, 0.2, 0.25, 0.3, 0.35\}$; number of epochs $e = 3000$, test set is varied over $N_{\text{test}} = \{20, 40, 80, 160, 320\}$, number of hidden layer varied over $h_l = \{2, 4, 6, 8, 10\}$; number of neurons per hidden layer varied over $\{10, 20\}$, α is varied over $\{0.1, 0.3, 0.7, 0.9, 1\}$, $N_0 = N_b = 50$, and the number of data points for the training set is $8N$ for $N \in N_{\text{test}}$, respectively. In all figures and tables presented in this paper, T_l denotes the total loss and C_t denotes the computational time in seconds.

To asses the effects of the Hurst parameter and the data size on the effectiveness of equation (2.13) to pricing European put options subject to the initial and boundary condition (2.14) using PINN, we consider the following examples.

Example 4.1 Let $r = 0.05$, K=10, $\sigma = 0.35$, $N_{\text{test}} = \{20, 40, 80, 160, 320\}$, $h_l = 4$, $n = 10$, $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9, 1\}$.



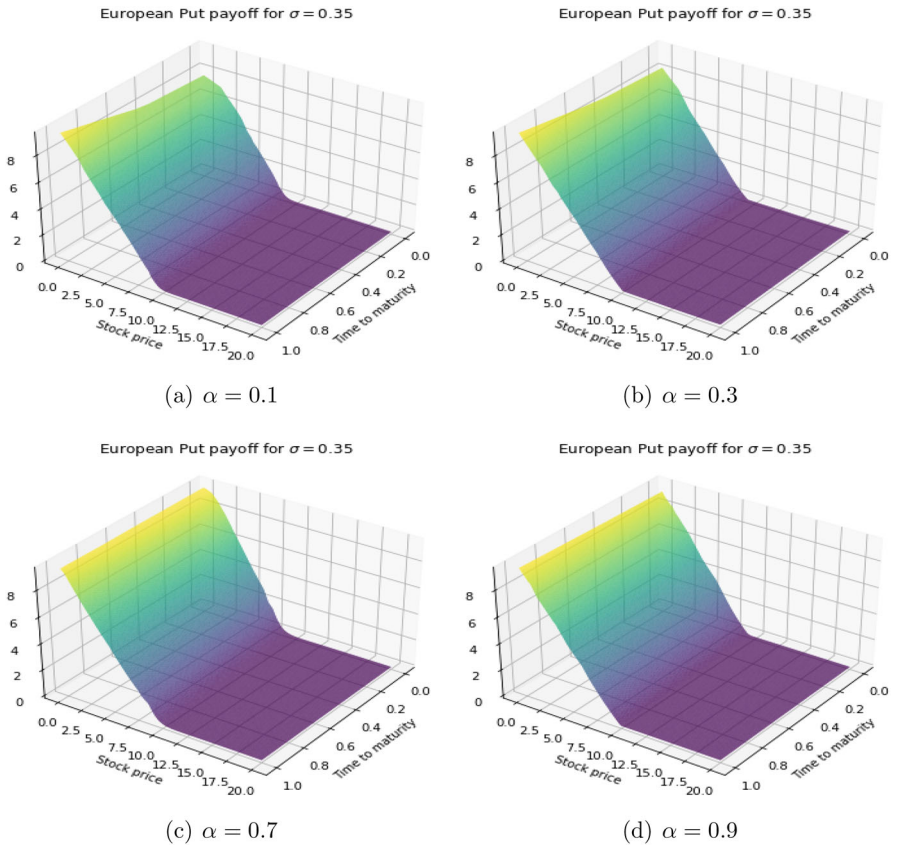
(a) $N = 160,\ N_{train} = 8N$ (b) $\alpha = 0.9$

(c) $N = 160,\ N_{train} = 8N$ (d) $\alpha = 0.9$

Fig. 2 Option payoff and absolute errors

Option maturity payoff and the maximum absolute error curves using data mentioned in Example 4.1 are presented in Fig. 2.

Figure 2a shows very smooth curves for all values of α except for $\alpha = 1$ which corresponds to the classical case. These results are well consistent with those in existing literature., see for example (Cen et al. 2018; Kumar et al. 2014; Nuugulu et al. 2021, 2023) just to mention but a few. In Fig. 2, we further observe that, the MAE error reduces with increase in the value of the Hurst parameter. For example, when $\alpha < 0.5$, the errors are a bit higher and reduces as $\alpha \to 1$ until at $\alpha = 1$ when it break down. As expected, since the MAE is calculated with respect to the standard Black-Scholes model, the errors should reduce as the model turns to the classical case, i.e., as $\alpha \to 1$. Table 2 suggest that the model learns better for smaller values of α compared to higher values of α making the tfBSPDE more robust in pricing European put than the classical case of the BSPDE. The minimum error occurs for $\alpha = 0.9$. The errors seem



(a) $\alpha = 0.1$ (b) $\alpha = 0.3$

(c) $\alpha = 0.7$ (d) $\alpha = 0.9$

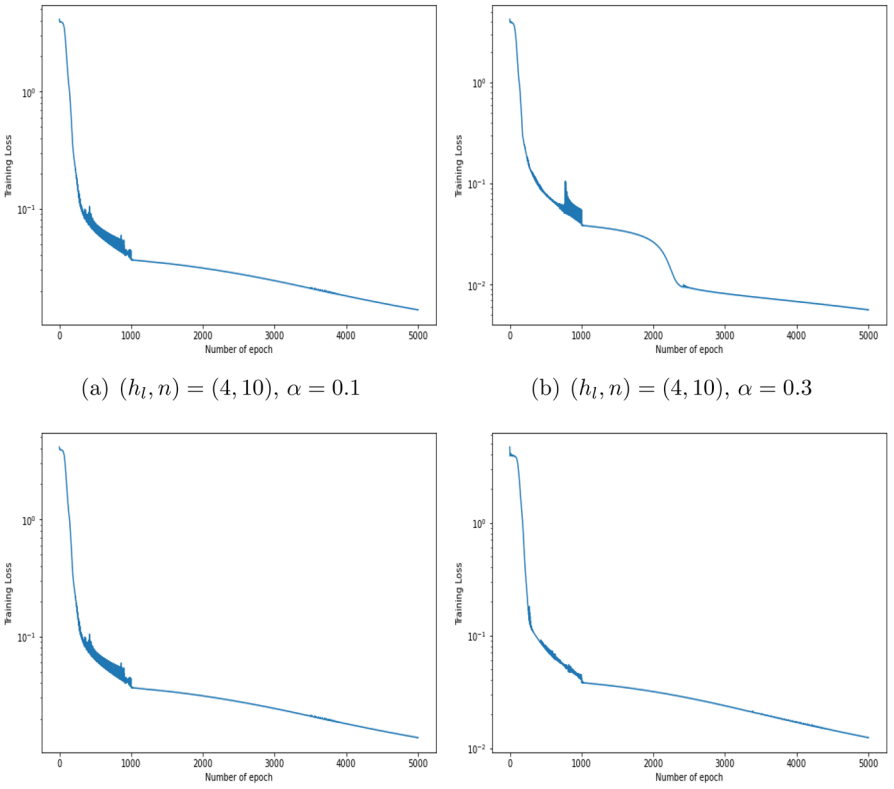
Fig. 3 European put payoff for $\alpha = \{0.1, 0.3, 07, 0.9\}$, $N = 160$, and $N_{\text{train}} = 8N$

to be adopting the dynamics and structure of the solution of equation (2.13), there by conforming with Eq. (3.23).

To asses the effects of some key options' parameters like interest rate and volatility on option prices, we consider example 4.2 which brings out the relation between r, σ and the absolute error when pricing European put options using equation (2.13) subject to the initial and boundary condition (2.14).

Example 4.2 Let $r = \{0.02, 0.05, 0.075, 0.1\}, K = 10, \sigma = \{0.15, 0.2, 0.250.3, 0.35\}$, $N = 150$, $h_l = 4$, $n = 10$, $\alpha = 0.9$.

Based on the results of example 4.1, the model performs better for higher volatile markets, again with respect to the Black-Scholes model, this is expected. From Fig. 5c, d, it may be misleading to conclude that the optimal set of input parameters based on the MAE are $\alpha = 0.9$, $\sigma = 0.3$, and $r = 0.1$. This is only true because we are evaluating the MAE with the BSM as the bench mark model, hence it is expected that the least errors will appear as $\alpha \to 1$, since for α close to 1, the tfBSPDE turns to the classical BSPDE. We therefore conclude that, the proposed model and solution



(a) $(h_l, n) = (4, 10)$, $\alpha = 0.1$ (b) $(h_l, n) = (4, 10)$, $\alpha = 0.3$

(c) $(h_l, n) = (4, 10)$, $\alpha = 0.7$ (d) $(h_l, n) = (4, 10)$, $\alpha = 0.9$

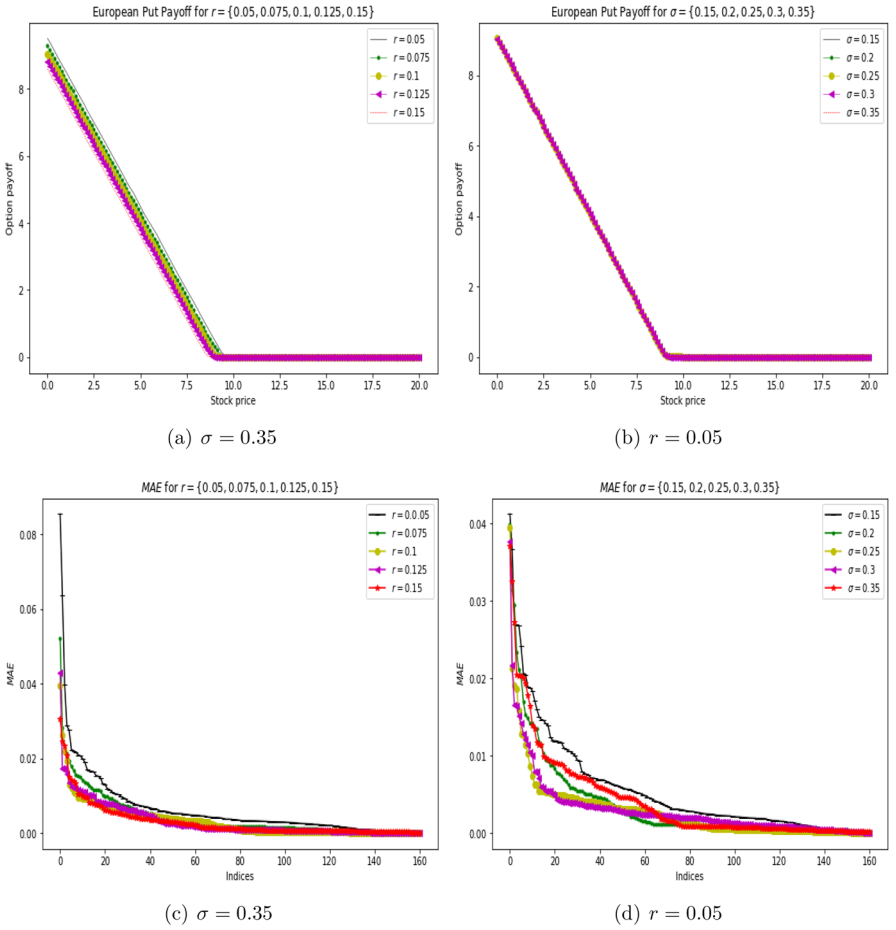
Fig. 4 Training Loss for $N = 160$, $N_{\text{train}} = 8N$, and $\alpha = \{0.1, 0.3, 0.7, 0.9\}$

method herein does performs best for smaller values of α, but since the BSM performs poorly for these market scenarios, we observe a higher error at these values compared to those at higher values of α. To validate these conclusions, and also asses the effects of the neural networks' structure, we consider the following Example 4.3.

Example 4.3 Effect on neural network structure in approximating the solution of equation (2.13) subject to (2.14): $\alpha = 0.9$, $\sigma = 0.3$, $r = 0.05$, $N = 150$, $N_{train} = 8N$, $h_l = \{2, 4, 6, 8\}$, $n = \{10, 20\}$.

We observe that the most smooth payoff curves are those in Figs. 7(a) and 7(b). We also observe that the errors from Table 4, are slightly smaller compared to those presented in Tables 1 and 3. Figure 6(b) suggest that the optimal structure is (4, 20). This is further justified by Fig. 7 and Table 4.

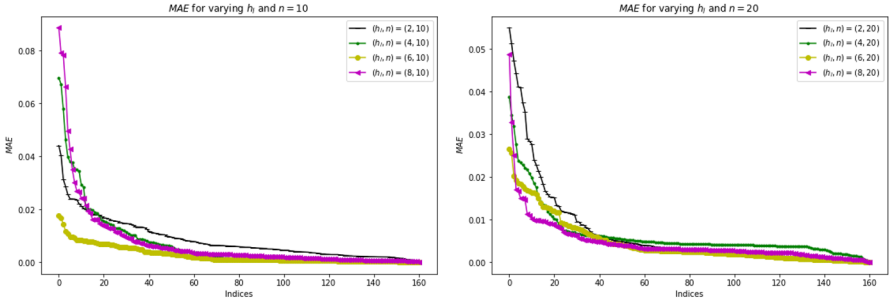
Through the option pricing procedure described in the current work, one can construct an appreciably accurate model for pricing options. To the best of our knowledge, the current approach is better than other numerical techniques such as finite difference methods (FDMs). It is now well known that stock prices dynamics do not necessarily follow smooth increments as it is the case when using FDMs to price options. However,



(a) $\sigma = 0.35$ (b) $r = 0.05$

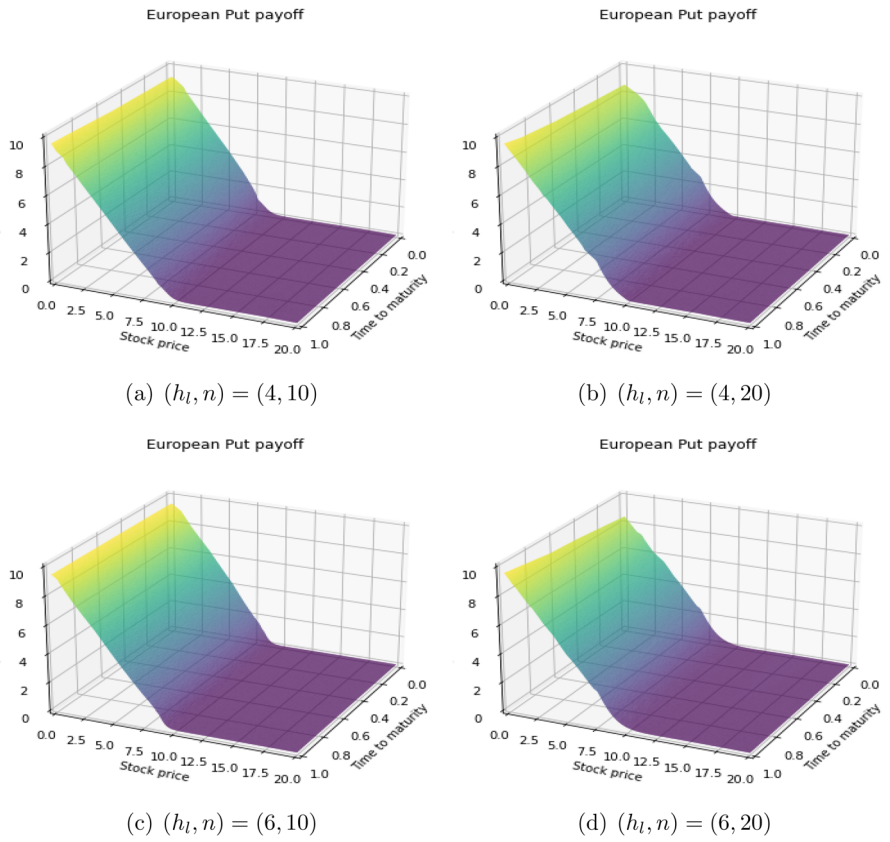
(c) $\sigma = 0.35$ (d) $r = 0.05$

Fig. 5 Behavior of payoff and MAE for varying values of r and σ



(a) $h_l = \{2, 4\},\ n = 10$ (b) $h_l = \{6, 8\},\ n = 10$

Fig. 6 Behavior of the MAE as we vary the number of hidden layers from 2 to 8 and $n \in \{10, 20\}$, for $\alpha = 0.9, \sigma = 0.3, r = 0.05$



(a) $(h_l, n) = (4, 10)$ (b) $(h_l, n) = (4, 20)$

(c) $(h_l, n) = (6, 10)$ (d) $(h_l, n) = (6, 20)$

Fig. 7 European put payoff, $h_l = \{4, 6\}$, and $n = \{10, 20\}$

under the PINN framework, stock prices can be obtained using the formula

$$x_j = x_0 e^{\sigma B_t{}^{\alpha} - r t_i - \frac{\sigma^2}{2} t_i{}^{2\alpha}}, \tag{4.48}$$

at each point in time t_i within the life of the option contract as presented in Sect. 4.1. While FDMs can only be implemented on a limited number of grid points due to computational memory issues, PINNs can evaluate a wider domain. For FDMs, error can be reduced with smoother data which is false of market conditions while for the PINN, error are reduced with noisier data which reflect market conditions better.

To further support the claim that the PINN approach is more appropriate than traditional numerical methods, one notes that most numerical methods for solving fractional Black-Scholes models uses the classical Black-Scholes model as the exact solution, despite its drawbacks when calibrated to market data. When solutions from methods like finite difference methods (FDMs) are compared to the payoff from classical Black-Scholes model, consistency and convergence are evaluated based on the

assumption that there is an exact solution (i.e., usually taken as the payoff from standard Black-Scholes model in the case of a European option), see for example (Cen et al. 2018; Kumar et al. 2014; Nuugulu et al. 2021) and some references therein.

From Table 2, we observe that since the PINN is able to generate an initial solution which is a linear combination of the weighted inputs and optimize the loss based on this solution to get a predicted solution that satisfies the global minimum of the governing payoff function, if it exists, one can study the extent to which the approximate solution approaches the true market dynamics even if the exact solution to the governing PDE is not known. In the current work, with 1280 data points, we were able to achieve a loss of the order $e-5$ which is quite reasonable for a network trained on data of this size. This also demonstrates the power of the PINNs in learning from sparse data, showcasing another advantage of PINN over other methods.

4.2 Limitations faced in solving tfBSPDE using PINN and some approach taken to optimize performance of the method

There are some issues that seem to limit the optimal performance of the PINN approach in solving the tfBSPDE. We now discuss how we tried to handle them where we could.

The stiffness of the equation: For small values of α, the tfBSPDE can be stiff (i.e., The long term memory effect caused by small values of α). This long term memory can lead to rapidly changing responses to historical data leading to significant stiffness as the solution needs to account for both short-term and long-term variations. Using the optimization algorithm by tuning the learning rate to update after every 1000 epochs starting from 0.001 and using 3000 epochs, we were able to minimize the effect of stiffness. This is intended to help in the rate at which the network remembers information through the batches. To further resolve the stiffness of the problem, we tune some hyperparamters, for example, the number of neurons and hidden layers.

Second issue is that of existence of a unique global minimum. The fact that the loss function may have several local minima may turn to make the result of the optimization algorithm very sensitive.The optimization algorithm that we used addresses this issue very well as it combines the strength of the Adaptive Gradient Algorithm and Root Mean Square Propagation. By carefully incoperating the initial and boundary conditions into the PINN framework, we avoid the problem of instability of the solution that could have been caused by wrongly inputting initial and boundary conditions into the PINN framework.

There were also scaling problems with the PINN framework which were handled by normalizing the data before fitting for training. PINN can be highly sensitive to hyperparamters such as network depth and width which affect the accuracy and stability of the solution. This explains why in our experiments, we varied these parameters to locate the optimal.

5 Concluding remarks and the scope for further research

We have established a novel approach to solve the time fractional Black-Scholes equation. We discussed how the fractional derivative can be incorporated into the loss function so that the backpropagation algorithm can be implemented without any difficulty in evaluating a fractional order derivative. Through this work, we aim to contribute to the growing body of research on fractional calculus and its applications in quantitative finance, paving the way for more sophisticated and accurate models in the pricing and hedging of financial derivatives.

Integrating advanced mathematical models like fractional Black-Scholes equations and machine learning techniques like neural networks makes it possible to learn and explain unusual market dynamics and also aid in understanding better the nonlinear relationship that exist between variables, there by providing more accurate predictions.

In the current work, extensive simulations were performed to visualize the behavior of the maximum absolute error. The graphs of the training loss against the number of epochs shows that the model indeed converges to the target solution as the training progresses, see Fig. 4. The effect of networks complexity on the model's performance is described in figures 6, 7 and Table 4. Furthermore, the optimal structure is found to be $(h_l, n) = (4, 20)$. The results from our simulations shows that our model does appreciably well in predicting option prices. In Sect. 3, we explained mathematically how PINN network propagate forward and backward based on the backpropagation algorithm and the Adam optimization techniques. Furthermore we studied the behavior of changes in the interest rate and the volatility and discussed the effect on option prices and the maximum absolute error.

To the best of our knowledge, our model is quite robust and captures the market dynamics correctly as demonstrated through various results. If trained on a well processed and market consistent data, the model can give much better predictions. In upcoming studies, we aim to apply this method to complex models arising in fractal market scenarios. Additionally, we intend to identify a more resilient optimization algorithm suitable for this configuration, while also exploring caliberation of such models to real-time market data.

Acknowledgements Not applicable.

Author Contributions All authors contributed equally to the manuscript.

Funding Open access funding provided by University of Namibia. Research of KCP was supported by the South African National Research Foundation.

Availability of data and material Data available on request.

Declarations

Conflict of interest No Conflict of interest to declare.

Ethics approval and consent to participate The study does not include human or animals and no ethical approval was required.

Consent for publication No personal data are reported in the manuscript.

References

Cai E, Zheng M, Zhang X, Lin ZG, Karniadakis GE (2021) Identifiability and predictability of integer-and fractional-order epidemiological models using physics-informed neural networks. Nat Comput Sci 1(11):744–753. https://pubmed.ncbi.nlm.nih.gov/38217142/

Cen Z, Huang J, Xu A, Le A (2018) Numerical approximation of a time-fractional Black-Scholes equation, *Computers & Mathematics with Applications* **75(8)** 2874-2887. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Z.+Cen%2C++J.+Huang%2C++A.+Xu%2C+A.+Le%2C+Numerical+approximation+of+a+time-fractional+Black%E2%80%93Scholes+equation%2C+%7B%5Cit+Computers+%24%5C%26%24+Mathematics+with+Applications%7D+%7B%5Cbf+75%288%29%7D+%282018%29+2874-87&btnG=

Cervera JG (2019) Solution of the Black-Scholes equation using artificial neural networks, *In Journal of Physics: Conference Series IOP Publishing*, **(Vol. 1221)** 012044. https://www.researchgate.net/publication/333768027_Solution_of_the_Black-Scholes_equation_using_artificial_neural_networks

Chen Q, Sabir Z, Raja MAZ, Gao W, Baskonus HM (2023) A fractional study based on the economic and environmental mathematical model, *Alexandria Engineering Journal*, **65** 761-770. https://www.sciencedirect.com/science/article/pii/S1110016822006275

Eskiizmirliler S, Günel K, Polat R (2021) On the solution of the black-scholes equation using feed-forward neural networks, *Computational Economics*, **58** 915-941. https://www.researchgate.net/publication/346528561_On_the_Solution_of_the_Black-Scholes_Equation_Using_Feed-Forward_Neural_Networks

Golbabai A, Nikan O, Nikazad T (2019) Numerical analysis of time fractional Black-Scholes European option pricing model arising in financial market, *Computational and Applied Mathematics* **38** 1-24. https://link.springer.com/article/10.1007/s40314-019-0957-7

Golbabai A, Nikan O (2020) A computational method based on the moving least-squares approach for pricing double barrier options in a time-fractional Black-Scholes model, *Computational Economics 55*,**1** 119-141. https://link.springer.com/article/10.1007/s10614-019-09880-4

Hull JC (2009) *Options, Futures and other Derivatives*, Upper Saddle River, NJ: Prentice Hall, https://books.google.co.za/books/about/Options_Futures_and_Other_Derivatives.html?id=sEmQZoHoJCcC&redir_esc=y

Ibrahim A, Lort H, Tatlicioglu BE (2023) Numerical investigation and deep learning approach for fractal-fractional order dynamics of Hopfield neural network model,*Chaos, Solitons & Fractals*, **177** 114302. https://www.sciencedirect.com/science/article/abs/pii/S0960077923012043?via%3Dihub

Ibrahim A, Hussain A, Kanwal T (2023) Investigating the impact of memory effects on computer virus population dynamics: A fractal-fractional approach with numerical analysis, *Chaos, Solitons & Fractals*, **174** 113845. https://www.sciencedirect.com/science/article/abs/pii/S0960077923007464?via%3Dihub

Jamshaid UR, Makhdoom F, Ali A, Danish S (2024) Mathematical modeling and simulation of biophysics systems using neural network, *International Journal of Modern Physics B 38*, **05** 2450066. https://www.worldscientific.com/doi/10.1142/S0217979224500668

Jumarie G (2006) Modified Riemann-Liouville derivative and fractional Taylor series of nondifferentiable functions further results, *Computers & Mathematics with Application* **51(9-10)** 1367-1376. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=G.+Jumarie%2C+Modified+Riemann-Liouville+derivative+and+fractional+Taylor+series+of+nondifferentiable+functions+further+results%2C+%7B%5Cit+Computers+%24%5C%26%24+Mathematics+with+Application%7D+%7B%5Cbf+51%289-10%29%7D+%282006%29+1367-1376.&btnG=

Jumarie G (2007) Fractional partial differential equations and modified Riemann-Liouville derivative new methods for solution, *Journal of Applied Mathematics and Computing*, **24** 31-48. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=G.+Jumarie%2C+Fractional+partial+differential+equations+and+modified+Riemann-Liouville+derivative+new+methods+for+solution%2C+%7B%5Cit+Journal+of+Applied+Mathematics+and+Computing%7D%2C+%7B%5Cbf+24%7D+%282007%29+31-48.&btnG=

Jumarie G (2008) Stock exchange fractional dynamics defined as fractional growth driven by usual Gaussian white noise, Application to fractional Black-Scholes, *Insurance, Math, Econom*, **12** 271-287. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=G.+Jumarie%2C+Stock+exchange+fractional+dynamics+defined+as+fractional+growth+driven+by+usual+Gaussian+white+noise%2C+Application+to+fractional+Black-Scholes%2C+%7B%5Cit+Insurance%2C+Math%2C+Econom%7D%2C+%7B%5Cbf+12%7D+%282008%29+271-287.&btnG=

Jumarie G (2009) Table of some basic fractional calculus formulae derived from a modified Riemann-Liouville derivative for non-differentiable functions, *Applied Mathematics Letters*, **22(3)** 378-385. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=G.+Jumarie%2C+Table+of+some+basic+fractional+calculus+formulae+derived+from+a+modified+Riemann%E2%80%93Liouville+derivative+for+non-differentiable+functions%2C+%7B%5Cit+Applied+Mathematics+Letters%7D%2C+%7B%5Cbf+22%283%29%7D+%282009%29+378-385.&btnG=

Jumarie G (2010) Derivation and solutions of some fractional Black-Scholes equations in coarse-grained space and time. Application to Merton's optimal portfolio, *Computers & Mathematics with Applications*, **59(3)**, 1142-1164. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=G.+Jumarie%2C++Derivation+and+solutions+of+some+fractional+Black-Scholes+equations+in+coarse-grained+space+and+time.+Application+to+Merton%27s+optimal+portfolio%2C+%7B%5Cit+Computers+%24%5C%26%24+Mathematics+with+Applications%7D%2C+%7B%5Cbf+59%283%29%7D%2C+%282010%29+1142-1164&btnG=

Kidger P (2022) On neural differential equations, arXiv preprint arXiv:2202.02435,

Kingma DP, Jimmy B (2014) Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Adam%3A+A+METHOD+FOR+STOCHASTIC+OPTIMIZATION&btnG=

Klibanov MV, Golubnichiy KV, Nikitin AN (2021) Application of neural network machine learning to solution of Black-Scholes equation, ArXiv Preprint arXiv:2111.06642,. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=M.V.+Klibanov%2C+K.V.+Golubnichiy+and+A.N.+Nikitin%2C+Application+of+neural+network+machine+learning+to+solution+of+Black-Scholes+equation%2C+%7B%5Cit+ArXiv+Preprint+ArXiv%3A2111.06642%7D%2C+%282021%29.&btnG=

Kumar S, Kumar D, Singh J (2014) Numerical computation of fractional Black-Scholes equation arising in financial market, *Egyptian Journal of Basic and Applied Sciences*, **1(3-4)** 177-183. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=S.+Kumar%2C+D.+Kumar%2C+and+J.+Singh%2C+Numerical+computation+of+fractional+Black%E2%80%93Scholes+equation+arising+in+financial+market%2C+%7B%5Cit+Egyptian+Journal+of+Basic+and+Applied+Sciences%7D%2C+%7B%5Cbf+1%283-4%29%7D+%282014%29+177-183.&btnG=

Lee D, Kim J, Jung K (2021) Improving object detection quality by incorporating global contexts via self-attention *Electronics*, **10(1)**, 90. https://www.researchgate.net/publication/348257021_Improving_Object_Detection_Quality_by_Incorporating_Global_Contexts_via_Self-Attention/figures?lo=1

Lou Q, Meng X, Karniadakis GM (2021) Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation, *Journal of Computational Physics*, **447** 110676. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Q.+Lou%2C+X.+Meng%2C++G.M.+Karniadakis%2C+Physics-informed+neural+networks+for+solving+forward+and+inverse+flow+problems+via+the+Boltzmann-BGK+formulation%2C+%7B%5Cit+Journal+of+Computational+Physics%7D%2C+%7B%5Cbf+447%7D+%282021%29+%7B110676%7D.&btnG=

Mandelbrot BB, van Ness JW (1968) Fractional Brownian motions, fractional noises and applications, *SIAM Review* **10** 422-437. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=B.B.+Mandelbrot+and+J.W.+van+Ness%2C+Fractional+Brownian+motions%2C+fractional+noises+and+applications%2C+%7B%5Cit+SIAM+Review%7D+%7B%5Cbf+10%7D+%281968%29+422-437.&btnG=

Mainardi F (2022) *Fractional calculus and waves in linear viscoelasticity: an introduction to mathematical models*, World Scientific. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=F.+Mainardi%2C++%7B%5Cit+Fractional+calculus+and+waves+in+linear+viscoelasticity%3A+an+introduction+to+mathematical+models%7D%2C+World+Scientific%2C+%282022%29.&btnG=

Nikan O, Zakieh A, Machado JAT (2024) Localized kernel - based meshless method for pricing financial options underlying fractal transmission system, *Mathematical Methods in the Applied Sciences 47*, **5** 3247-3260. https://onlinelibrary.wiley.com/doi/10.1002/mma.7968

Nuugulu SM, Gideon F, Patidar KC (2021) A robust numerical scheme for a time-fractional Black-Scholes partial differential equation describing stock exchange dynamics, *Chaos, Solitons & Fractals*, **145** 110753. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=S.M.+Nuugulu%2C+F.+Gideon%2C+and+K.C.+Patidar%2C+A+robust+numerical+scheme+for+a+time-fractional+Black-Scholes+partial+differential+equation+describing+stock+exchange+dynamics%2C+%7B%5Cit+Chaos%2C+Solitons+%24%5C%26%24+Fractals%7D%2C+%7B%5Cbf+145%7D+%282021%29+110753.&btnG=

Nuugulu SM, Gideon F, Patidar KC (2023) An Efficient Numerical Method for Pricing Double-Barrier Options on an Underlying Stock Governed by a Fractal Stochastic Process, *Fractal and Fractional*, **7(5)** 389. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=S.M.+Nuugulu%2C+F.+Gideon+and+K.C.+Patidar%2C++An+Efficient+Numerical+Method+for+Pricing+Double-Barrier+Options+on+an+Underlying+Stock+Governed+by+a+Fractal+Stochastic+Process%2C+%7B%5Cit+Fractal+and+Fractional%7D%2C+%7B%5Cbf+7%285%29%7D+%282023%29+389.&btnG=

Ostaszewicz AJ (2012) The Hurst parameter and option pricing with fractional Brownian motion (Doctoral dissertation, University of Pretoria). https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=A.J.+Ostaszewicz%2C+The+Hurst+parameter+and+option+pricing+with+fractional+Brownian+motion+%28Doctoral+dissertation%2C+University+of+Pretoria%29%2C+%282012%29.&btnG=

Owoyemi AE, Sumiati I, Rusyaman E, Sukono S (2020) Laplace decomposition method for solving fractional Black-Scholes European option pricing equation, *International Journal of Quantitative Research and Modeling*, **1(4)** 194-207. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=+A.E.+Owoyemi%2C+I.+Sumiati%2C+E.+Rusyaman+and+S.+Sukono%2C+Laplace+decomposition+method+for+solving+fractional+Black-Scholes+European+option+pricing+equation%2C+%7B%5Cit+International+Journal+of+Quantitative+Research+and+Modeling%7D%2C+%7B%5Cbf+1%284%29%7D+%282020%29+194-207.&btnG=

Pang G, Lu L, Karniadakis GE (2019) fPINN: Fractional physics-informed neural networks, *SIAM Journal on Scientific Computing* **41(4)** 2603-2626. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=G.+Pang%2C+L.+Lu+and+G.E.+Karniadakis%2C++fPINN%3A+Fractional+physics-informed+neural+networks%2C+%7B%5Cit+SIAM+Journal+on+Scientific+Computing%7D+%7B%5Cbf+41%284%29%7D+%282019%29+2603-2626.&btnG=

Pannas E. Long memory and chaotic models of prices on the London Metal Exchange Resources Policy, **27(4)** 235-246. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=E.+Pannas%2C+Long+memory+and+chaotic+models+of+prices+on+the+London+Metal+Exchange+Resources+Policy%2C+%7B%5Cbf+27%284%29%7D+235-246.&btnG=

Podlubny I (2001) Geometric and physical interpretation of fractional integration and fractional differentiation, arXiv preprint math/0110241 . https://arxiv.org/abs/math/0110241

Raissi M, Perdikaris P, Karniadakis GE (2017) Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561

Raissi M, Perdikaris P, Karniadakis EG (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, **378** 686-707. https://www.researchgate.net/publication/328720075_Physics-Informed_Neural_Networks_A_Deep_Learning_Framework_for_Solving_Forward_and_Inverse_Problems_Involving_Nonlinear_Partial_Differential_Equations

Sondermann D (2006) *Introduction to Stochastic Calculus for Finance*, Springer-Verlag, https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=+D.+Sondermann%2C++%7B%5Cit+Introduction+to+Stochastic+Calculus+for+Finance%7D%2C+Springer-Verlag%2C+2006.&btnG=

Syata I, Lesmana DC, Sumarno H (2015) Numerical method for determining option price with risk adjusted pricing methodology (RAPM) volatility model, *Applied Mathematical Sciences*, **9(134)** 6697-6705. https://www.researchgate.net/publication/306313840_Numerical_method_for_determining_option_price_with_risk_adjusted_pricing_methodology_RAPM_volatility_model

Tagliani A, Milev M (2013) Laplace transform and finite difference methods for the Black-Scholes equation, *Applied Mathematics and Computation*, **220** 649-658. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=A.+Tagliani+and+M.+Milev%2C+Laplace+transform+and+finite+difference+methods+for+the+Black%E2%80%93Scholes+equation%2C+%7B%5Cit+Applied+Mathematics+and+Computation%7D%2C+%7B%5Cbf+220%7D+%282013%29+649-658.&btnG=

Tan SH (2018) Towards efficient nonlinear option pricing, Doctoral dissertation, University of Greenwich, https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=S.H.+Tan%2C+Towards+efficient+nonlinear+option+pricing%2C+Doctoral+dissertation%2C+University+of+Greenwich%2C+2018.&btnG=

Ševčovič D, Žitňanská M (2016) Analysis of the nonlinear option pricing model under variable transaction costs, *Asia-Pacific Financial Markets*, **23(2)** 153-174. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=D.+%C5%A0ev%C4%8Dovi%C4%8D+and++M.+%C5%BDit%C5%88ansk%C3%A1%2C+Analysis+of+the+nonlinear+option+pricing+model+under+variable+transaction+costs%2C+%7B%5Cit+Asia-Pacific+Financial+Markets%7D%2C+%7B%5Cbf+23%282%29%7D++%282016%29+153-174.&btnG=

Wilmott P, Howson S, Howison S, Dewynne J (1995) *The Mathematics of Financial Derivatives: A Student Introduction*, Cambridge University Press,. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=P.+Wilmott%2C+S.+Howson%2C++S.+Howison+and+J.+Dewynne%2C++%7B%5Cit+The+Mathematics+of+Financial+Derivatives%3A+A+Student+Introduction%7D%2C+Cambridge+University+Press%2C+1995.&btnG=

You Y, Li J, Reddi S, Hseu J, Kumar S, Bhojanapalli S, Hsieh CJ (2019) Large batch optimization for deep learning: Training bert in 76 minutes, arXiv preprint arXiv:1904.00962