Pricing Rainbow Options Using PINNs

Ayesha Ahmad and adnan khan *

Posted Date: 9 October 2024

doi: 10.20944/preprints202408.2226.v2

Keywords: Qiuantitive finance; option pricing; rainbow options; PINNs; neural networks

 doi:10.20944/preprints202408.2226.v2

Article

Pricing Rainbow Options Using Deep Learnin

Ayesha Ahmad and Adnan Khan *

Lahore University of Management Sciences
* Correspondence: adnan.khan@lums.edu.pk

Abstract: In this study, we consider the valuation of rainbow options using unsupervised machine learning methods. In particular, we consider the pricing of multi-asset (rainbow) European and American options using Physics Informed Neural Networks (PINNS). After developing the PINNS architecture, we benchmark the method by using it to price vanilla and exotic options with one and two underlying assets. We then use the methodology to price a multi-asset European option, followed by the pricing of the multi-asset American option by solving the linear complementarity problem. We compare our results to those obtained using preexisting numerical methods and note excellent agreement. Unlike conventional numerical methods, we note that this methodology does not suffer from the 'curse of dimensionality'. The time complexity of our method is considerably less than that of the conventional techniques. Thus PINNS may offer a faster more efficient solution to the pricing of rainbow options.

Keywords: Qiuantitive finance; option pricing; rainbow options; PINNs; neural networks

1. Introduction

A significant portion of the financial instruments traded today are derivatives, whose values are derived from the performance of underlying assets such as stocks, bonds, indices, commodities, and interest rates. Options are among the most actively traded financial derivatives. An option is a contract that grants its holder the right, but not the obligation, to buy or sell underlying assets (such as stocks) at a predetermined price (*strike price*), on or before a specified date (*expiry*). There are two main kinds of options: call options (which provide the right to buy) and put options (which provide the right to sell). Investors use options for leverage and to hedge against financial risks. However, pricing these instruments can be mathematically challenging.

Options trading has a long history, with standardized options first traded on the Chicago Board Options Exchange (CBOE) on April 26, 1973. The Black-Scholes model [1], developed by Fischer Black and Myron Scholes in 1973, provided a groundbreaking closed-form solution for pricing European options. This work earned Black and Scholes the Nobel Prize in Economics in 1997 and laid the foundation for more complex option pricing models and associated numerical methods. Over the decades, advanced models incorporating realistic assumptions have been developed, though explicit solutions are often not feasible. Consequently, various numerical methods have emerged, including binomial and trinomial trees, Monte Carlo simulations and finite difference and finite element schemes for solving partial differential equations (PDEs).

[2] applied Monte Carlo simulation to the field of financial derivatives, this works by simulating a large number of random paths of the underlying asset's price to estimate the value of any financial derivative. [3] refined the Monte Carlo method for option pricing, particularly focusing on variance reduction techniques to improve computational efficiency. [4] gave a least-squares approach for valuing American Options by Monte-Carlo Simulations.

[5] introduced the binomial tree method; a discrete-time model for option pricing approximates the Black-Scholes model's continuous-time process. [6] developed the trinomial tree model which offered better accuracy than the binomial model by incorporating an additional possible state at each node. [7] extended the binomial model, typically applied to single-asset options, to multi-dimensional options.

Finite difference methods rely on discretizing a function on a grid. [8] extended the application of finite difference methods for solving the Black-Scholes partial differential equation, providing improved accuracy and stability. [9] proposed a fourth-order compact finite difference scheme to tackle

 doi:10.20944/preprints202408.2226.v2

a one-dimensional (1-D) nonlinear Black-Scholes equation, demonstrating unconditional stability. The application of the finite element method in option pricing by [10] offered a robust method for solving option pricing PDEs, particularly useful for American options and exotic derivatives with complex boundaries.

Finite elements were used to price multi-asset American options by [11]. [12] provides a thorough explanation and comparison of numerical methods for pricing financial derivatives, including finite difference methods. [13] presented a superconvergent fitted finite volume method for solving a degenerate nonlinear penalized Black-Scholes equation pertinent to European and American option pricing which was an improvement on conventional finite volume methods. [14] suggested an advanced high-order finite difference method applicable to various option pricing models, encompassing the 1-D nonlinear Black-Scholes equation, Merton's jump-diffusion model, and 2-D Heston's stochastic volatility model. [15] introduced a distinctive finite volume method tailored for solving the Black-Scholes model involving two underlying assets. [16] proposed a radial basis function combined with the partition of unity method for solving American options with stochastic volatility. [17] devised a radial basis function-generated finite difference (RBF-FD) method to solve a stochastic volatility jump model represented as a 2-D PIDE.

The formalization of Artificial Neural Networks originated by [18] as a programming paradigm inspired by biology, enabling computers to learn from observable data. The introduction of the error backpropagation learning algorithm by [19] greatly enhanced the appeal of neural networks (NNs) across diverse research fields. Today, NNs and deep learning are recognized as the most potent tools for addressing numerous challenges in image recognition, speech recognition, and natural language processing. They have also been applied to forecast and categorize economic and financial variables.

In the context of pricing financial derivatives, numerous studies have highlighted the benefits of employing neural networks (NNs) as a primary or supplementary tool. For example, [20] advocated the utilization of learning networks to estimate the value of European options. They asserted that learning networks could reconstruct the Black–Scholes formula by utilizing a two-year training set comprising daily options prices. The resulting network, according to their findings, could then be applied to derive prices and effectively delta-hedge options in out-of-sample scenarios. In their 2000 study, [21] derived a generalized option pricing formula with a structure akin to the Black–Scholes formula using a feed-forward neural network (NN) model. Their findings revealed minimal delta-hedging errors compared to the hedging effectiveness of the Black–Scholes model. [22]'s study highlighted the transformative potential of deep learning in finance, demonstrating how advancements in technology and the accessibility of vast datasets have democratized the application of sophisticated neural network models for option pricing, marking a significant leap forward in the integration of artificial intelligence within the financial sector.

In the study by [23], the researchers redefined the high-dimensional nonlinear Black–Scholes (BS) equation as a set of backward stochastic differential equations (BSDEs) and approximated the solution's gradient using deep neural networks. They illustrated the effectiveness of their deep BSDE method through a demonstration of a 100-dimensional problem. In a recent work by [24], the researchers suggested resolving the one-dimensional Black–Scholes (BS) equation to predict the value of European call options. They achieved this by employing a feed-forward neural network, specifically one with a single hidden layer. Additional sources discussing the utilization of neural networks in option pricing and hedging can be explored in a recent review article by [25].

Physics-Informed Neural Networks (PINNs) have shown promising results in solving partial differential equations (PDEs) by incorporating domain knowledge into the training process. Option pricing often involves solving complex PDEs, such as the Black-Scholes equation or more advanced models. Recently [26] used PINNs to price path-dependent options like American options. Also, PINNs have successfully been used to price two-dimensional European and American options by [27] In this work we focus on the pricing of multi-asset 'Rainbow' options (both American and European); such contracts enable investors to speculate on the relative performance of multiple underlying assets.

 doi:10.20944/preprints202408.2226.v2

Rainbow options have gained prominence due to their ability to capture correlations and interactions among various assets, making them a valuable tool for risk management. Rainbow options also play an important role in diversification and complex trading strategies.

The factors that determine the ease or difficulty of pricing multi-asset options include (1) the existence of a closed-form solution (they usually lack closed-form solution), (2) the number of underlying assets i.e. the dimensionality, (3) path dependency, (4) early exercise.

In the absence of closed-form solutions, numerical methods have to be used for multi-asset option pricing. Choosing a suitable numerical scheme involves a combination of speed, accuracy, simplicity, and generality.

The challenge is mainly that with many of the schemes, the computational efforts grow exponentially with the problem's dimensions. Some rainbow option problems have closed-form solutions; like exchange options or options with no path dependency and are relatively easy to price. For three or less dimensions, finite difference methods (FDM) or finite element methods (FEM) are efficient. They cope well with early exercise and many path-dependent features can be incorporated, though usually at the cost of an extra dimension. A more recent method for pricing rainbow options is the collection method. These methods work particularly well for low-dimensional problems. However, for higher dimensions, they become unstable and cannot provide accurate results [28]. For higher dimensions, Monte Carlo simulations are good. Unfortunately, they are not very efficient for American-style early exercise.

There is currently no numerical method that works very well with such a problem. As a result, the problem of efficiently pricing American rainbow option pricing for higher dimensions remains an interesting problem. This makes it a very interesting task to tackle with heuristic methods such as Physics Informed Neural Networks.

This paper is organized as follows. After the introduction and literature review in section 2, PINNs methodology is presented and the mathematical formulation of PINNs for option pricing PDEs is discussed in section 3. In section 4, we demonstrate the effectiveness and robustness of the proposed method for pricing options by comparing its results with existing methods. We benchmark the method by pricing a one-dimensional European option, a one-dimensional Asian option and two-dimensional cash-or-nothing option. In section 5, we use PINNs to calculate option prices for European and American-style rainbow options with four underlying assets and discuss the efficiency and accuracy of the method. In section 6, we summarize our findings and discuss future work.

2. Methodology: Solving Option Pricing PDEs Using PINNs:

For a given differential equation:

$$F(x, y(x), Dy(x), ..., D^m y(x)) = 0$$

in unknown function y along with appropriate boundary and/or initial conditions, PINNs approximate the solution $y(x)$, by using an ANN: $\hat{y}_w(x)$, which is completely described by a set of parameters (weights) W. The loss function of the neural network is constructed by squaring the given differential equation, along with the squared boundary and initial conditions and the weights W are adjusted such that $\hat{y}_w(x)$ minimizes the loss function which is the same as saying that $\hat{y}_w(x)$ solves (to some accuracy) the given PDE while satisfying the given conditions. Here x represents the independent variables in the form of an n-tuple. After figuring out the domain of the independent variables, the points are randomly selected from that domain to act as inputs for our NN.

Vanilla or European Rainbow Option is a financial contract that gives the holder the right to buy/sell (call/put) $'d'$ assets (U_i) for prescribed strike (S) at maturity (T). Since it allows the holder to exercise at a fixed maturity date, its payoff depends upon the price of the underlying stock at expiry only and not the path of underlying. Therefore it is a path-independent option. For the European option with $'d'$ number of underlying the associated PDE is:

 doi:10.20944/preprints202408.2226.v2

$$\frac{\partial \mathscr{O}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}U_iU_j\frac{\partial^2\mathscr{O}}{\partial U_i\partial U_j} - r\sum_{i=1}^{d}\frac{\partial}{\partial U_i}U_i + r\mathscr{O} = 0 \tag{1}$$

where $\mathscr{O}$ is the value of the option, each σ_i is the volatility of underlying stock U_i , ρ_{ij} is the correlation coefficient between stock i & j , r is the interest rate, and $\tau = T - t$ is the time to expiry (T is the expiry, $t \in [0, T]$ is instantaneous time). Equation (1) combined with payoff $g(U, \tau = 0)$ determines the price of the option $\mathscr{O}$

American Rainbow option is a financial contract that gives the holder the right to buy/sell (call/put) $'d'$ assets (U_i) for prescribed strike (S) at any time up to and including maturity (T). This early exercise feature makes the American option valuation problem a free boundary problem i.e. there exists an unknown boundary, that depends upon time. This boundary acts as the decision surface between early exercise and holding the option and it has to be determined as a part of the problem. So for American option valuation, we rewrite the Black Scholes PDE (used to solve European options) to a linear complementary form: which implicitly includes the free boundary condition into the PDE.

Multi-asset American option in the Linear Complementary Problem (LCP) approach forms a multi-dimensional partial differential complementary problem (PDCP) [29];

$$(\frac{\partial \mathscr{O}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}U_iU_j\frac{\partial^2\mathscr{O}}{\partial U_i\partial U_j} - r\sum_{i=1}^{d}\frac{\partial \mathscr{O}}{\partial U_i}U_i + r\mathscr{O}).(\mathscr{O}(U,\tau) - g(U,\tau)) = 0 \tag{2}$$

However for this problem an analytical solution is not readily available. Since we want to convert black Scholes PDE into a Physics Informed Neural Network problem, first, we scale the input (stock) and adjust our PDE accordingly. **Scaling** is done to improve convergence, speed and stability during training. It prevents features with large magnitudes from dominating the optimization process, ensuring fair contributions from all features; as pointed out by [30]. For scaling we substitute:

$x_i = U_i/S$ and $\mathscr{V}(x,t) = \mathscr{O}(x,t)/S$ These substitutions give us scaled PDE for the European Option:

$$\frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}x_ix_j\frac{\partial^2\mathscr{V}}{\partial x_i\partial x_j} - r\sum_{i=1}^{d}\frac{\partial \mathscr{V}}{\partial x_i}x_i + r\mathscr{V} = 0 \tag{3}$$

and for American Option:

$$(\frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}x_ix_j\frac{\partial^2\mathscr{V}}{\partial x_i\partial x_j} - r\sum_{i=1}^{d}\frac{\partial \mathscr{V}}{\partial x_i}x_i + r\mathscr{V}).(\mathscr{V}(x,\tau) - g(x,\tau)) = 0 \tag{4}$$

To convert the above black Scholes pde for **European option** with multiple underlying into a Physics Informed Neural Network problem, first, we describe our pde in general form as

$\mathscr{D}(\mathscr{P},\mathscr{V}) : \frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}x_ix_j\frac{\partial^2\mathscr{V}}{\partial x_i\partial x_j} - r\sum_{i=1}^{d}\frac{\partial \mathscr{V}}{\partial x_i}x_i + r\mathscr{V} = 0$ in the interior of domain

$\mathscr{B}(\mathscr{P},\mathscr{V}) : g(x_1, x_2, ..., x_d, \tau)$ or any other Boundary condition, on the boundary of the domain

Similarly to convert **the American Option pricing problem** to a **Physics Informed Neural Network** problem, let's describe our PDE in general form as

$$\mathscr{D}(\mathscr{P},\mathscr{V}) : (\frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\sigma_i\sigma_j\rho_{ij}x_ix_j\frac{\partial^2\mathscr{V}}{\partial x_i\partial x_j} - r\sum_{i=1}^{d}\frac{\partial \mathscr{V}}{\partial x_i}x_i + r\mathscr{V})$$

$$.(\mathscr{V}(x_1, x_2, ..., x_d, \tau) - g(x_1, x_2, ..., x_d, \tau)) = 0$$

in the interior of domain

$$\mathscr{B}(\mathscr{P},\mathscr{V}) : g(x_1, x_2, ..., x_d, \tau)$$

 doi:10.20944/preprints202408.2226.v2

or any other boundary condition, on the boundary of the domain.
Here $\mathscr{D}$ and $\mathscr{B}$ are the differential operators on the domain and boundary respectively. In PINNs, the initial conditions are treated the same as the boundary conditions. We select (randomly) a collection of interior and boundary points P_i^I, P_i^B where each point is in an n-tuple of size $n = d + 1$ (for d=number of stocks and extra one dimension to accommodate time) The solution of any differential equation using PINNs involves minimizing a single loss function defined as weighted sum of the L_2 norm of differential equation and boundry conditions:

$$L(W) = \frac{w_I}{|P^I|} \sum_{i=1}^{n_I} \mathscr{D}(\mathscr{P}_i^I, \hat{\mathscr{V}}(W))^2 + \frac{w_B}{|P^B|} \sum_{i=1}^{n_B} \mathscr{B}(\mathscr{P}_i^B, \hat{\mathscr{V}}(W))^2$$

where w_I and w_B are weights and P^I and P^B are the sets of input points.

Minimizing $L(W)$ means we are trying to find $\hat{\mathscr{V}}(W)$ i.e. a set of weights W for which the PDE, $\mathscr{D}(U_i^I, \hat{\mathscr{V}}(W))$ is zero or as close to zero as possible and the boundary conditions are also met, i.e. $\hat{\mathscr{V}}(W)$ solves the PDE.([31])

Algorithm 1 PINN Algorithm for Solving Option Pricing Differential Equations

Step 1: Construct a neural network $\hat{\mathscr{O}}(U; W)$ with parameters W.
Step 2: Define two training sets, P^I for the equation and P^b for the boundary and initial conditions.
Step 3: Specify the loss function $L(W; P)$ by summing the weighted L^2 norm of both the PDE equation and boundary/initial condition residuals.
Step 4: Train the neural network by minimizing the loss function to find the optimal parameters W^*:

$$W^* = \arg\min_W L(W; P)$$

We use deepxde library of python for our work.

Algorithm 2 Usage of DeepXDE for Solving Option Pricing Differential Equations

Step 1: Specify the computational domain using the geometry module.
Step 2: Define the PDE according to the option under consideration using TensorFlow's syntax.
Step 3: Specify the boundary and initial/final conditions according to the nature of the option.
Step 4: Combine the geometry, PDE, and boundary/initial/final conditions into `data.TimePDE`
Step 5: Construct a neural network using the maps module.
Step 6: Define a Model by integrating the PDE problem from Step 4 and the neural network from Step 5.
Step 7: Call `Model.compile` to set the optimization hyperparameters, including the optimizer and learning rate.
Step 8: Call `Model.train` to train the network from random initialization
Step 9: Call `Model.predict` to obtain the PDE solution at various locations.

To check the efficiency of our models we calculate the mean square root of error (RMSE),R_2 and L_2 relative error of our model (P_i) against any preexisting solution method (O_i) and $\bar{O}$ is the mean of the observed values .

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (P_i - O_i)^2}{n}}$$

$$L_2 = \frac{\sum_{i=1}^n (P_i - O_i)^2}{\sum_{i=1}^n (O_i)^2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^n (P_i - O_i)^2}{\sum_{i=1}^n (O_i - \bar{O})^2}$$

like in [32]

 doi:10.20944/preprints202408.2226.v2

3. Benchmarking

We now use the PINNs methodology to solve some well-understood problems. We will compare our results with those in the literature. We have trained our Neural Networks using T4GPU on Google Colab.

3.1. Bench marking with One-dimensional linear case:

First, we will consider solving the standard Black Scholes PDE equation for one underlying stock, obtained by putting $d = 1$ in equation(3):

$$\frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 \mathscr{V}}{\partial x^2} - rx\frac{\partial \mathscr{V}}{\partial x} + r\mathscr{V} = 0 \tag{5}$$

The price of **the European put Option** $(\mathscr{V}_p)$ is obtained by solving the above BSPDE subject to these conditions:

$$\mathscr{V}_p(x,\tau) = S * e^{-r\tau} \qquad at \qquad x = 0$$

$$\mathscr{V}_p(x,\tau) = 0 \qquad at \qquad x = L$$

$$\mathscr{V}_p(x,\tau) = g(x,\tau = 0) = max(S - x(0), 0) \qquad at \qquad \tau = 0$$

where L is some suitably large value obtained by truncating the original $(0,\infty)$ domain to $(0,L]$. We take $L = 5S$. This is a rather simple problem that has an explicit solution. The exact solution of this BSPDE can be found in Ch7 [28]

We solve this model for the following parameters:

$$S = 4, \quad r = 0.03, \quad \sigma = 0.3, \quad T = 1$$

These parameters are the same as the ones used in [27].

We have two inputs (scaled stock price and time to expiry) and one output (scaled option price). We use a fully connected neural network with depth 4 (ie 3 hidden layers) and width 20 i.e. 20 neurons in each layer. For a fully connected feed forward neural network, the weights between two layers are equal to the number of neurons in the first layer multiplied by the number of neurons in the second layer, plus one bias weight per neuron in the second layer, which means our model has 921 weights.We utilize 2000*10 training residual points sampled within the spatio-temporal domain, 200*10 training points sampled on the boundary, and 100*10 residual points for the initial conditions. The activation function used is the hyperbolic tangent (tanh). The training begins with the Adam optimizer for 1000 iterations at a learning rate of 0.001, after which we switch to the L-BFGS optimizer. L-BFGS does not require a learning rate, and the neural network is trained until it converges. The number of training points, iterations, and the structure of the NN are decided upon careful consideration and trying different values until one with the best convergence and least L2 score is figured out.

The "Adam" optimizer had a train loss of 1.26e-03, and training took 4.4 seconds; The "L-BFGS" optimizer had a train loss of 2.50e-06 and training took 40.64 seconds. So both the optimizers together took the training time of 45.04 seconds

Figure 1 shows the test train loss history

 doi:10.20944/preprints202408.2226.v2

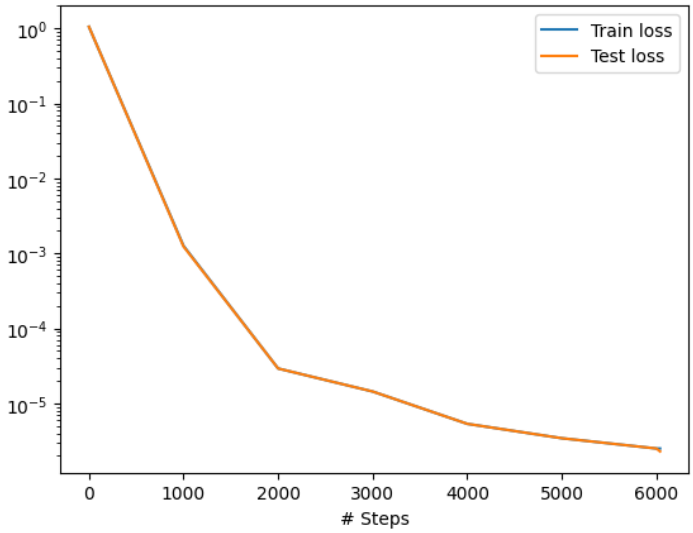


Figure 1. training history for one underlying European put option

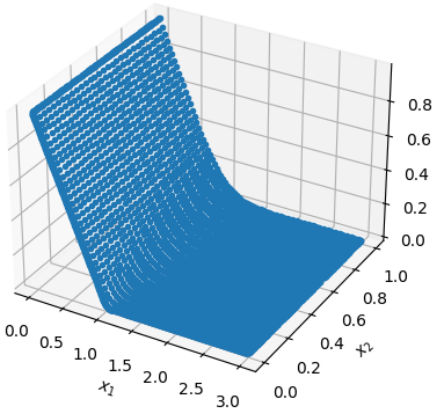
Figure 2 shows the 3d plot of numerical solution over whole time domain $[0, T)$ where $x_1 = U/S$ and $x_2 = \tau$. For error comparison, first, we convert the output of our PINNs which is the scaled price ($\mathscr{V}$), to option price ($\mathscr{O}$) by using the relation:

$$\mathscr{O} = S * \mathscr{V}$$

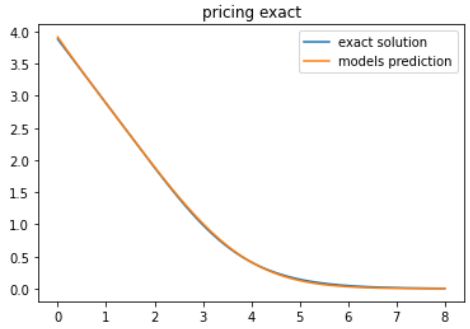
In Figure 2(b), we plot the numerical solution using PINNs against the exact solution calculated using formulas from [29] at $\tau = 1$. In Figure 2(c), we plot the corresponding pointwise error (calculated as the predicted solution - exact solution), over the whole stock space $(0, L]$. In Figure 2(d), we show the error histogram by taking 1000 random points from the domain $(0, L] \times [0, 1]$. Error is calculated using

$$\text{error}_i = \frac{(V_i^{exact}) - (V_i^{nn})}{\overline{(V_i^{exact})}}$$

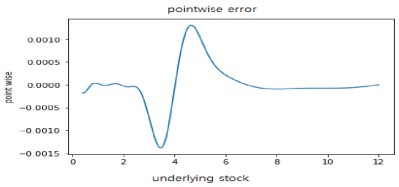
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2024 doi:10.20944/preprints202408.2226.v2



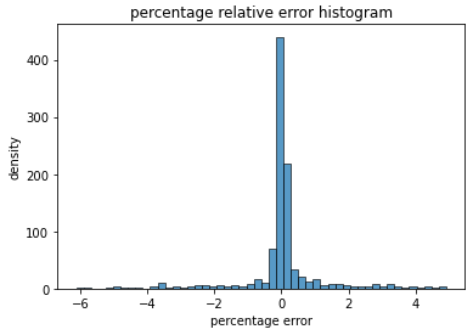
(a) 3D plot of one underlying European put option



(b) Exact solution and numerical solution using PINNs for one underlying European put option



(c) Pointwise error for one underlying European put option



(d) Error histogram for one underlying European put option

Figure 2. Various plots and errors for 1D one underlying European put option

The L_2 relative error is 0.0044 which is very small and quite acceptable.

3.2. Benching Marking with Two-Dimensional Put Option

In this section, we consider the two-asset option pricing problem, which is governed by the following PDE obtained by substituting $d = 2$ in equation(3)

$$\frac{\partial \mathscr{O}}{\partial \tau} - \frac{1}{2}\sigma_1^2 U_1^2 \frac{\partial^2 \mathscr{O}}{\partial^2 U_1} - \frac{1}{2}\sigma_2^2 U_2^2 \frac{\partial^2 \mathscr{O}}{\partial^2 U_2}$$

$$-\rho\sigma_1\sigma_2 U_1 U_2 \frac{\partial^2 \mathscr{O}}{\partial U_1 \partial U_2} - rU_1 \frac{\partial \mathscr{O}}{\partial U_1} - rU_2 \frac{\partial \mathscr{O}}{\partial U} + r\mathscr{O} = 0 \tag{6}$$

$$\text{on} \quad (0,\infty)^2 \times [0,T]$$

where option value $\mathscr{O}(U_1, U_2, \tau)$ depends upon prices of underlying asset $U_i (i = 1, 2)$ and time to maturity τ . The payoff function for cash or nothing put option is:

$$\mathscr{O}(U_1, U_2, \tau) = \begin{cases} C & \text{, if } U_1 < S_1 \& U_2 < S_2 \\ 0 & \text{, otherwise} \end{cases}$$

where $C > 0$ is a fixed amount. For comparison we choose the same model parameters as by [27]

$S_1 = S_2 = 5, \quad T = 1, \quad \sigma_1 = 0.2, \quad \sigma_2 = 0.3, \quad \rho = 0.1, \quad r = 0.1, \quad C = 1$

The physical domain is truncated to a bound rectangle: $0 \leq U_1 \leq 10$ and $0 \leq U_2 \leq 10$

 doi:10.20944/preprints202408.2226.v2

Dirichlet boundary conditions $\mathcal{O} = 0$ are imposed on $U_1 = x_{\max}$ and $U_2 = y_{\max}$. On $U_2 = 0$, the boundary condition is chosen as the one-dimensional European binary put option on U_1, which is

$$V(U_1, 0, \tau) = Ce^{-r\tau}N(-d_1), \tag{7}$$

where

$$d_1 = \frac{\log(U_1/K_1) + (r - \sigma_1^2/2)\tau}{\sigma\sqrt{\tau}},$$

and $N(z)$ is the cumulative distribution function. By the same argument, the boundary condition on $U_1 = 0$ is given by

$$V(0, U_2, \tau) = Ce^{-r\tau}N(-d_2), \tag{8}$$

where

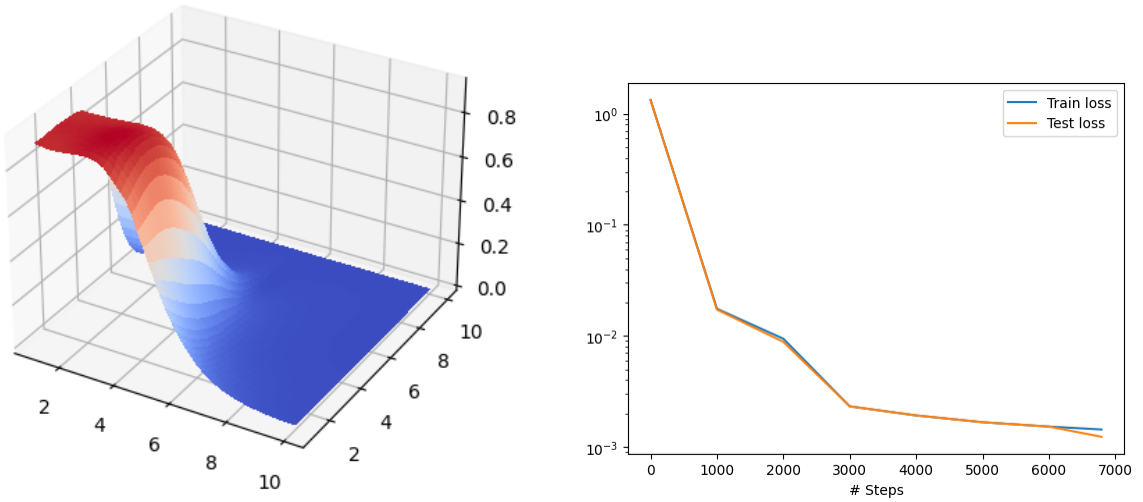
$$d_2 = \frac{\log(U_2/K_2) + (r - \sigma_2^2/2)\tau}{\sigma\sqrt{\tau}}.$$

Here We have three inputs (price of stock one, price of stock two and time to expiry) and one output (cash or nothing put option price). We use a fully connected neural network with depth 6 (which means 5 hidden layers) and width 30. There are 3871 weights in this NN.

We utilize 2000*10 training residual points sampled within the spatio-temporal domain, 200*10 training points sampled on the boundary, and 100*10 residual points for the initial conditions. The activation function used is the hyperbolic tangent (tanh). The training begins with the Adam optimizer for 2000 iterations at a learning rate of 0.001, after which we switch to the L-BFGS optimizer. L-BFGS does not require a learning rate, and the neural network is trained until it converges.

The "Adam" optimizer had a train loss of 1.47e-02 , and training took 22.98 seconds; The "L-BFGS" optimizer had a train loss of 1.56e-03 and training took 73.83 seconds. So both the optimizers together took the training time of 96.81 seconds

Figure 3b shows the test train loss history



(a) 3d plot of numerical solution at $\tau = 1$ for Two-underlying call or nothing put option

(b) training history of neural network for Two-underlying call or nothing put option

Figure 3. Plots related to Two-underlying call or nothing put option

The obtained numerical solution plot is shown in Figure 3a. It shows the 3d plot of the numerical solution at final time T or time to expiry zero, the x-axis and y-axis in the plot represent the two underlying stocks. Our results match those presented in , [27]

 doi:10.20944/preprints202408.2226.v2

3.3. Bench Marking with One-Dimensional Path-Dependent Option

Pricing Asian average rate call Options we introduce a new variable $A(t)$

$$A(t) = \frac{1}{t}\int_0^t x(s)ds$$

and solve the system of PDEs

$$\frac{\partial \mathscr{V}}{\partial t} + \frac{1}{2}\sigma^2 x^2 \frac{\partial^2 \mathscr{V}}{\partial x^2} + rx\frac{\partial \mathscr{V}}{\partial x} - r\mathscr{V} = 0 \tag{9}$$

and

$$\frac{dA}{dt} - \frac{x(t) - A(t)}{t} = 0 \tag{10}$$

subject to the terminal condition of option price:

$$\mathscr{O}(t = T) = \max(A - x(T), 0)$$

and initial condition of average

$$A(t) = x(t)$$

boundary conditions for this average rate call options are:

$$\mathscr{V}(x, A, t) = (S - A) \qquad at \qquad x = 0$$

$$\mathscr{V}(x, A, t) = 0 \qquad at \qquad x = L$$

$$\mathscr{V}(x, A, t) = g(x, A) = \max(S - A, 0) \qquad at \qquad t = T$$

where L is some suitably large value by truncating the original $(0, \infty)$ domain to $(0, L]$. We take $L = 5S$.

A singularity exists in the equation for A at $t = 0$ (i.e., today). However, at $t = 0$ the term $\frac{U - A}{t} * \frac{\partial \mathscr{O}}{\partial A}$ also goes to zero leaving us with a BSPDE for the European option which has an exact solution. so we leave out t=0 and take our time domain as $(0, T]$ or $[0.0001, T]$ ([33])

We solve this model problem with parameters:

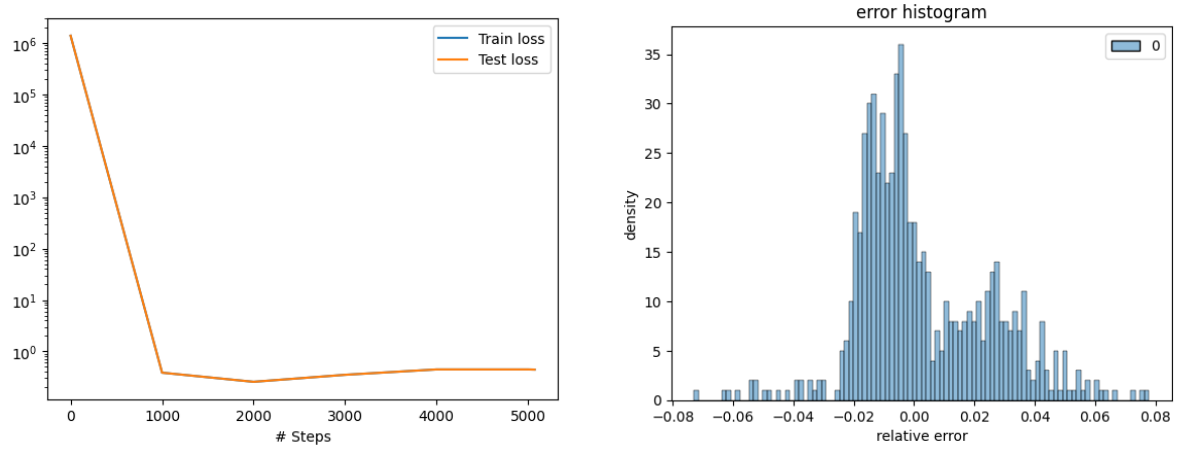
$$S = 1, \quad r = 0.04, \quad \sigma = 0.03, \quad T = 1$$

Note that for an Asian option, we use the actual time (t) instead of introducing τ because t appears explicitly in this pde. We have two inputs (scaled stock price and time to expiry) and two outputs (average price, and option price). We use a fully connected neural network with depth 4 (ie 3 hidden layers) and width 20. Thus there are 942 weights. We utilize 2000*10 training residual points sampled within the spatio-temporal domain, 200*10 training points sampled on the boundary, and 100*10 residual points for the initial conditions. The activation function used is the hyperbolic tangent (tanh). The training begins with the Adam optimizer for 2000 iterations at a learning rate of 0.001, after which we switch to the L-BFGS optimizer. L-BFGS does not require a learning rate, and the neural network is trained until it converges.

The "Adam" optimizer had a train loss of 2.94e-01, and training took 13.90 seconds; The "L-BFGS" optimizer had a train loss of 2.76e-01 and training took 4.64 seconds. So both the optimizers together took training time of 18.54 seconds

For error comparison we randomly generated 1000 data points from the input domain of our NN i.e. $(0, L]x(0, T]$. The first coordinate of the out point represents the initial value of stock(U_i) and the other represents a time to expiry(t). Then we calculated the option value (V_i^{mc}) for those points using the Monte Carlo method with 100000 paths and 256 time steps. We also calculate the values our model predicts (V_i^{nn}) for each point. The relative error is calculated using the formula:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2024 doi:10.20944/preprints202408.2226.v2



(a) training history of 0ne-underlying Asian option

(b) error histogram of 0ne-underlying Asian option

Figure 4. Plots related to 0ne-underlying Asian option

$$\text{error}_i = \frac{(V_i^{mc}) - (V_i^{nn})}{(V_i^{mc})}$$

We plot an error histogram, to evaluate our results [22]. The root-mean-squared error (RMSE) is 0.0041, which may be compared to the fact that the strike prices are all normalized to \$1. Hence the average error is less than ±0.4% of the strike. The average percentage pricing error (error divided by option price) is 0.0220 i.e., almost 2%. The histogram of pricing errors is shown 5(a). We see that the errors are sufficiently small. Further, we estimated a regression of the model values and obtained R2 = 0.9981, i.e. almost 99.8% which is quite acceptable.

4. Pricing of Rainbow Options

We now turn to the problem of pricing rainbow options. We will consider both American and European options in our study. We will compare our results to those obtained using existing methods and show that the method is comparable in accuracy to the existing techniques while being faster at the same time.

4.1. Pricing of European Call Options with Four Underlying Stocks

The scaled PDE for the European call option that will act as the loss function of our PINNs will be:

$$\frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{4}\sigma_i\sigma_j\rho_{ij}x_ix_j\frac{\partial^2\mathscr{V}}{\partial x_i\partial x_j} - r\sum_{i=1}^{4}\frac{\partial \mathscr{V}}{\partial x_i}x_i + r\mathscr{V} = 0 \tag{11}$$

which is obtained by substituting $d = 4$ in equation(3). The initial condition for the max call option is:

$$g(x_1, x_2, x_3, x_4, \tau = 0) = \max(\max(x_1, x_2, x_3, x_4) - 1, 0)$$

Unlike for the one-dimensional or two-dimensional cases, we don't have clear values at the stock boundary but for the multi-dimension call option we can impose this Boundary condition at boundary $x_i = L$

$$\mathscr{V}(x, \tau) = max(x_i - 1, 0) * e^{r*\tau} \qquad as \qquad x_i = L$$

for any $i \in \{1, 2, 3, 4\}$

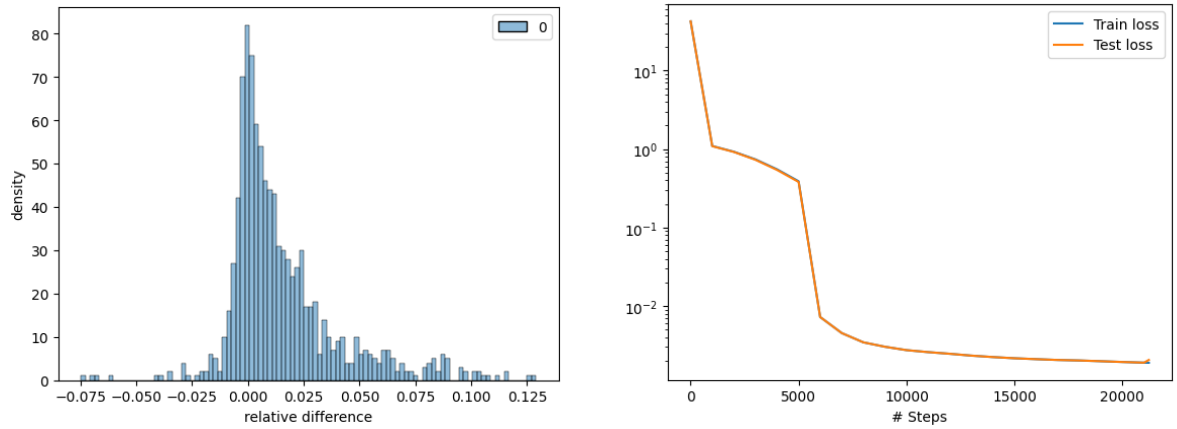
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2024 doi:10.20944/preprints202408.2226.v2

We took all our underlying stocks, U_i to be in the domain $(0, L]$; where L is sufficiently large.($L = 5 * S$). Since our input, x_i is scaled as $x_i = U_i/S$ our x_i is in the domain $(0, 5]$ for each stock. We take these parameters for our option

- Strike Price (S) : \$100 ,
- Maturity (T) : 1 year,
- Risk-free rate (r) : 0.01,
- volatility of stocks (σ_i) : 0.08 ,
- correlation between stocks (ρ_{ij}) : 0.5

To train the network 2000*10 training residual points are sampled within the spatio-temporal domain, 200*10 training points are sampled on the boundary, and 100*10 residual points for the initial conditions. We have five inputs (price of stock one, initial price of stock two, initial price of stock three, price of stock four and time to expiry) and one output (multi-asset max call option price). We choose a fully connected neural network of depth 3 (i.e. 2 hidden layers) and width 20. So there are 561 weights. We start with "tanh" as the activation switch. After defining the neural network we build the model; choosing "adam" as the optimizer and "10−3" learning rate for "5000" iterations where an iteration is the number of times a certain batch is passed via an algorithm. After training with "adam" we train with "L-BFGS" until convergence.

The "Adam" optimizer had a train loss of 3.89e-01, and training took 60.77 seconds; The "L-BFGS" optimizer had a train loss of 1.89e-03 and training took 248.28 seconds. So both the optimizers together took the training time of 309.05 seconds. Figure 5b shows the test train loss history



(a) error plot (b) training history of neural network

Figure 5. 4-asset European option

For error comparison we generate option values for 1000 data points using another numerical solution, the Monte Carlo method with 100000 paths and 256 time steps. To evaluate our solution we plot the error histogram. The root-mean-squared error (RMSE) is 0.00965, which is very reasonable considering that the strike prices are all normalized to \$1. Hence the average error is less than ±1% of the strike. The average percentage pricing error (error divided by option price) is 0.0312, i.e., almost 3.1%. The histogram of pricing errors is shown 5a. We see that the errors are very small. Further, we estimated a regression of the model values and obtained an R2 = 0.9856, which again is quite good.

4.1.1. American Option PINNs for Four Underlying

The scaled PDE for the American call option that will act as the loss function of our PINNs will be:

$$\frac{\partial \mathscr{V}}{\partial \tau} - \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{4}\sigma_i\sigma_j\rho_{ij}x_ix_j\frac{\partial^2\mathscr{V}}{\partial x_i\partial x_j} - r\sum_{i=1}^{4}\frac{\partial \mathscr{V}}{\partial x_i}x_i + r\mathscr{V}.(\mathscr{V}(x,\tau) - g(x,\tau) = 0 \tag{12}$$

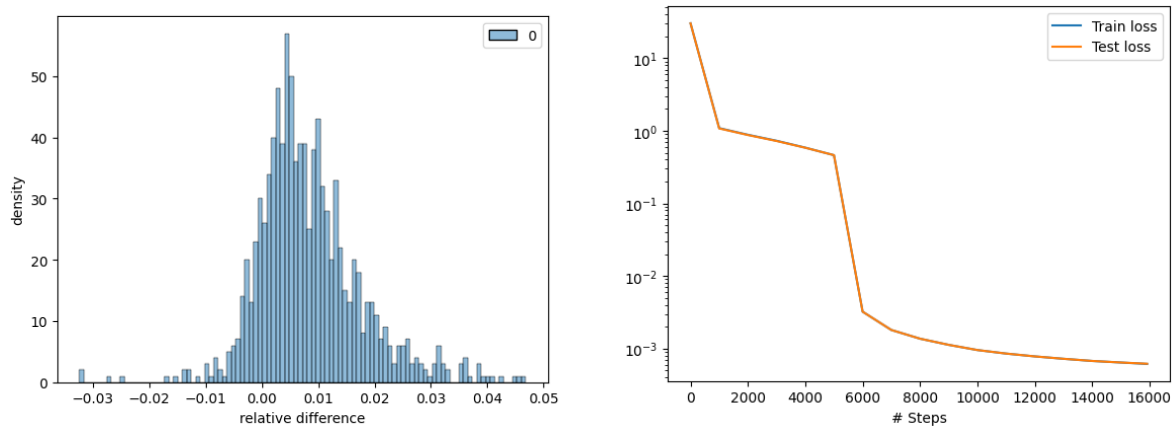
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2024 doi:10.20944/preprints202408.2226.v2

where g is the initial condition

$$g(x_1, x_2, x_3, x_4, \tau = 0) = max(max(x_1, x_2, x_3, x_4) - 1, 0)$$

The same problem parameters as in the European option's case are also used for the American option. We also take the same model parameters and training points as for the European 4D call option.

The "Adam" optimizer had a train loss of 4.62e-01, and training took 60.80 seconds; The "L-BFGS" optimizer had a train loss of 6.12e-04 and training took 159.80 seconds. So both the optimizers together took the training time of 220.60 seconds. Figure 6b shows the test train loss history



(a) error plot (b) training history of neural network

Figure 6. 4-asset American option

Just like we did in the case of the European multi-asset option, for error comparison we generate option values for 1000 data points using Monte Carlo with 100000 paths and 256 time steps. To evaluate our result we plot error histogram. The root mean-squared error (RMSE) is 0.0014, which may be compared to the fact that the strike prices are all normalized to \$1. Hence the average error is less than ±0.15% of the strike. The average percentage pricing error (error divided by option price) is 0.0138, i.e. 1.4% of option price. The histogram of pricing errors is shown in Figure 6a. which shows that the errors are very small. Further, we estimated a regression of the model values and obtained an R2 = 0.9977 , which is very high. Thus we can conclude that the method gives very good results.

For time comparison we saw that determining the European option's (with four underlying stocks) values using the Monte Carlo method with 100000 paths and 256 time steps required 661.8950 seconds. Conversely, training and generating the value of an option using PINNs took 312.8564 seconds, meaning a 52.73% improvement. Determining the American option's (with four underlying stocks) values using the Monte Carlo method with 100000 paths and 256 time steps required 1128.2052 seconds. Conversely, training and generating the value of an option using PINNs took 223.7130 seconds which means an 80.17 improvement. Also employing a trained neural network to calculate the option value always took less than 0.2 seconds which is very fast and can be of great use in case one is racing against time. This substantial enhancement in time efficiency underscores the efficacy of neural networks, Also a very interesting observation is that NN converges faster and better for the American Option than it does for the European option, which points towards the phenomenon that the complexity of the PDE does not affect the speed and performance of PINNs negatively.

5. Conclusion

In this study we considered the pricing of rainbow options (options with multiple underlying assets) using PINNs. We give an overview of the methodology and a literature review in the intro-

 doi:10.20944/preprints202408.2226.v2

duction section. This is followed by benchmarking the method by applying it to some well studied problems. We then apply the method to pricing of rainbow options and discuss our findings.

In section 3 the method is used to price European and American type options with two underlying stocks and Asian options which are strongly path dependent. This is done as a benchmarking exercise as efficient numerical methods exist for these problems. We compare our results to those obtained by traditional numerical schemes and note that PINNs demonstrates strong performance across various scenarios.

We next apply PINNs to the pricing of multi-asset options. Pricing of these instruments pose significant computational challenges, including 'the curse of dimensionality'. We consider European and American options with four underlying stocks and note that PINNs is an efficient method for pricing these instruments.

In summary, leveraging AI and neural networks, which are rapidly evolving fields, in option pricing not only showcases current effectiveness but also opens avenues for leveraging future advancements in AI and PINNs. Once the network is trained, the option pricing process becomes significantly faster than existing algorithms. Moreover, we observed substantial improvements in time efficiency, especially when generating values across multiple stocks and time intervals for the same option.

As future work we would like to apply the methodology to more complex methods and numerically challenging option price models, involving stochastic volatility and transaction cost.

References

1. Black, F.; Scholes, M. The pricing of options and corporate liabilities. *Journal of Political Economy* **1973**, *81*, 637–654. doi:https://doi.org/10.1086/260062.
2. Boyle, P.P. Options: A Monte Carlo approach. *Journal of Financial Economics* **1977**, *4*, 323–338. https://doi.org/10.1016/0304-405X(77)90005-8.
3. Broadie, M.; Glasserman, P. Estimating security price derivatives using simulation. *Management Science* **1996**, *42*, 269–285.
4. Longstaff, F.A.; Schwartz, E.S. Valuing American Options by Simulation: A Simple Least-Squares Approach. *Review of Financial Studies* **2001**, *14*, 113–147. doi:http://dx.doi.org/10.1093/rfs/14.1.113.
5. Cox, J.C.; Ross, S.A.; Rubinstein, M. Option pricing: A simplified approach. *Journal of Financial Economics* **1979**, *7*, 229–263.
6. Boyle, P.P.; Evnine, J.; Gibbs, S. Numerical evaluation of multivariate contingent claims. *Review of Financial Studies* **1989**, *2*, 241–250.
7. Moon, K.S.; Kim, W.J.; Kim, H. Adaptive lattice methods for multi-asset models. *Computers & Mathematics with Applications* **2008**, *56*, 352–366.
8. Courtadon, G. A more accurate finite difference approximation for the valuation of options. *Journal of Financial and Quantitative Analysis* **1983**, *17*, 697–703.
9. Liao, W.; Khaliq, A.Q.M. High-order compact scheme for solving nonlinear Black–Scholes equation with transaction cost. *International Journal of Computer Mathematics* **2009**, *86*, 1009–1023. https://doi.org/10.1080/00207160802209788.
10. Andalaft-Chacur, A.; Ali, M.M.; Salazar, J.G. Real options pricing by the finite element method,. *Computers and Mathematics with Applications* **2011**, *16*, 2863–2873.
11. Kaya, D. Pricing a Multi-Asset American Option in a Parallel Environment by a Finite Element Method Approach. PhD thesis, Uppsala University, Department of Mathematics, 2011.
12. Seydel, R.U. *Tools for Computational Finance*, 5th ed.; Springer, 2012.
13. Wang, S.; Zhang, S.; Fang, Z. A superconvergent fitted finite volume method for Black-Scholes equations governing European and American option valuation: SUPERCONVERGENT FITTED FINITE VOLUME METHOD. *Numerical Methods for Partial Differential Equations* **2014**, *31*, 1190–1208. doi:10.1002/num.21941.
14. Dilloo, M.J.; Tangman, D.Y. A high-order finite difference method for option valuation. *Computers & Mathematics with Applications* **2017**, *74*, 652–670. doi:10.1016/j.camwa.2017.05.006.
15. Koffi, R.S.; Tambue, A. A Fitted L-Multi-Point Flux Approximation Method for Pricing Options. *Computational Economics* **2022**, *60*, 633 – 663.

 doi:10.20944/preprints202408.2226.v2

16. Mollapourasl, R.; Fereshtian, A.; Vanmaele, M. Radial Basis Functions with Partition of Unity Method for American Options with Stochastic Volatility. *Computational Economics* **2019**, *53*, 259–287. doi:10.1007/s10614-017-9739-8.
17. Soleymani, F.; Zhu, S. RBF-FD solution for a financial partial-integro differential equation utilizing the generalized multiquadric function. *Computers & Mathematics with Applications* **2021**, *82*, 161–178.
18. McCulloch, W.S.; Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics* **1943**, *5*, 115–113. . Reprinted in McCulloch 1964, pp. 16–39.
19. Rumelhart, D.; Hinton, G.; Williams, R. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536.
20. Hutchinson, J.M.; Lo, A.W.; Poggio, T. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The journal of Finance* **1994**, *49*, 851–889.
21. Garcia, R.; Gençay, R. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics* **2000**, *94*, 93–115.
22. Culkin, R. Machine Learning in Finance: The Case of Deep Learning for Option Pricing. *computer science* **2017**.
23. Han, J.; Jentzen, A.; Ee, W. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences* **2018**, *115*, 8505–8510.
24. Eskiizmirliler, S.; Günel, K.; Polat, R. On the solution of the black-scholes equation using feed-forward neural networks. *Computational Economics* **2021**, *58*, 915–941.
25. Ruf, J.; Wang, W. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance* **2020**, *24*, 1–45. doi:10.21314/JCF.2020.390.
26. Gatta, F.; Di Cola, V.S.; Giampaolo, F.; Piccialli, F.; Cuomo, S. Meshless methods for American option pricing through Physics-Informed Neural Networks. *Engineering Analysis with Boundary Elements* **2023**, *151*, 68–82. doi:https://doi.org/10.1016/j.enganabound.2023.02.040.
27. Wang, X.; Li, J.; Li, J. A Deep Learning Based Numerical PDE Method for Option Pricing. *Computational Economics* **2023**, *62*, 149–164. doi:10.1007/s10614-022-10279-.
28. Wilmott, P. *Paul Wilmott on Quantitative Finance*, second ed.; John Wiley and Sons, 2006.
29. Wilmott, P.; Dewynne, J.; Howison, S. *Option Pricing: Mathematical Models and Computation*; Oxford Financial Press, 1993.
30. Sola, J.; Sevilla, J. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science* **1997**, *44*, 1464–1468. doi:10.1109/23.589532.
31. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *SIAM Journal on Scientific Computing* **2019**, *41*, A463–A483. doi:10.1137/19M1274067.
32. Toit, J.F.D.; Laubscher, R. Evaluation of Physics-Informed Neural Network Solution Accuracy and Efficiency for Modeling Aortic Transvalvular Blood Flow. *Mathematical and Computational Applications* **2023**, *28*, 62. doi:10.3390/mca28020062.
33. Hozman, J.; Tichý, T. DG method for numerical pricing of multi-asset Asian options—The case of options with floating strike. *Applied Mathematics* **2017**, *62*, 171–195. doi:10.21136/AM.2017.0273-16.