Physica D 484 (2025) 134992

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Data driven neural network approaches for pricing options ☆

Samuel M. Nuugulu [a,*], Kailash C. Patidar [b], Divine T. Tarla [b]

[a] Department of Computing, Mathematical & Statistical Sciences, University of Namibia, Private Bag, 13301, Windhoek, Namibia
[b] Department of Mathematics and Applied Mathematics, University of the Western Cape, Private Bag X17, Bellville 7535, South Africa

ARTICLE INFO

Communicated by Victor M. Perez-Garcia

Keywords:
Option pricing
Implied volatility
Neural networks
PINN
Purely data driven approach
Heston stochastic volatility model
Partial differential equations
Numerical methods

ABSTRACT

This paper presents two data driven approaches, the purely data driven (PDD) and physics informed neural network (PINN) approach for solving asset pricing problems. The PDD approach relies purely on available data and does not require any governing partial differential equation (PDE) to solve a pricing problem. On the other hand, under the PINN approach, the pricing is done by solving a governing PDE. Both models are calibrated to observed market prices, and their implied volatilities are compared to those derived from market data and the classical Black–Scholes model. The absolute errors and maximum absolute errors metrics relative to observed implied volatilities and prices and the prices obtained from the classical Black–Scholes model were used in measuring the goodness-of-fit of the two proposed techniques. Several hyperparameter tuning techniques were employed to optimize the performance of the two methods. In addition, we analyze the probability density functions (PDFs) derived from each method and verify that they are valid by demonstrating positivity and proper normalization. Theoretical results, including propositions and theorems, are presented to establish conditions under which the PINN, trained using the Adam optimizer and initialized via the Xavier method, converges to an optimal solution, i.e., a set of trainable parameters that minimize the loss function. In further extensions, the PINN approach was applied to pricing European put options under a Heston stochastic volatility model (HSVM) model. While both methods exhibit competitive performance when calibrated, our empirical findings indicate that the PINN approach yields superior accuracy and stability.

1. Introduction

Several assumptions governing classical asset pricing models in financial markets are almost always defaulted. For example, under the Black–Scholes (BS) model, the volatility is assumed to be a constant and the asset price returns is assumed to follow a Gaussian process. These assumptions are in most cases violated, and as such, several revised models has been proposed. Most of the revised models are of a nonlinear nature and that in itself presents some mathematical complexity in developing tractable solutions to such models. The challenge of developing efficient and robust numerical solutions to nonlinear PDEs arising in finance remain eminent. With the advances in theoretical data science however, the use of deep learning in solving PDEs has become a subject of active research. In the current work, we present a comparative study of two deep neural network based techniques for pricing financial derivatives. The two techniques generally falls under the following categories; the purely data driven approach (PDD) and the partially data driven-physics informed neural network (PINN) approach. According to Raissi et al. [1], PINN are a class of neural networks that are trained to solve supervised learning tasks while respecting any governing law of physics described by the governing partial differential equations. According to Raissi et al. [2], deep learning techniques present more accurate ways of approximating option prices compared to traditional approaches which makes several simplifying assumptions about the underlying processes. The efficiency in deep learning methods is owed to the fact that, the involved neural networks are able to learn complex nonlinear relationships between variables in the models.

Several numerical methods for solve PDEs arising in finance have thus far been proposed, see for example [3–5] just to mention but a few. Syata et al. [4] studied some numerical methods for determining option prices using a risk adjusted pricing methodology. More work on the valuation of derivatives using numerical methods can be seen in [6]. However, with the complications involved in valuing certain derivatives, recent approaches have shifted from numerical methods to machine learning methods such as neural networks. Some of the current work on the use of neural network based methods include but not limited to [1,2,7–10]. In [7] they sovled a Black–Scholes PDE using neural network based method though details on the underlying

☆ This article is part of a Special issue entitled: 'MLDSAIT' published in Physica D.

* Corresponding author.
E-mail addresses: snuugulu@unam.na (S.M. Nuugulu), kpatidar@uwc.ac.za (K.C. Patidar), 3921379@myuwc.ac.za (D.T. Tarla).

https://doi.org/10.1016/j.physd.2025.134992
Received 12 December 2024; Received in revised form 3 June 2025; Accepted 18 October 2025
Available online 11 November 2025

architecture was not presented. Klibanov et al. [9] also presented an approach for solving Black–Scholes where they looked at predicting stock prices one day ahead of time. Some other works on solving PDEs using neural networks (PINNs) can also be found in [8,10]. Other ideas for solving PDEs using neural networks (PINNs) are discussed in [1,2], whereby the governing PDEs is incorporated into the neural network's loss function. Under this approach, the loss function is used for penalizing for any deviations from the governing physical laws, thereby enforcing physical consistency in the network predictions.

From the origin and literature of neural networks as stated in [1,11], neural networks can be used to solve for the solution to almost any problem conceived by the human brain. However there may be computational challenges due to the model's complexity in certain instances. Using available data or discretization techniques, one can find a way to get a point-wise approximation to the parameters of a complex equations and still borrow the idea of PINN to train the model.

Unlike traditional solution methods such as finite difference that depends purely on the discretization of the governing PDEs the PINN approach uses both data and the governing physical equation, see [10]. 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, PINNs are nonlinear models in nature and as such, can adequately explain the nonlinear market dynamics which may not be feasible to locate by studying the convergence and consistency properties numerical solutions governing PDEs. The PINN models adaptively learn the underlying dynamics without the need to explicitly discretize the domain, which is a significant advantage over traditional numerical techniques, which can some times be computationally expensive, see for example [1,10] and references therein. Although the PINN approach has recently emerged as a promising approach for solving PDEs, it does suffer from a few drawbacks, such as non-existence of a unique solution, global minimum and also presents some scalability issues.

In addition to the PINN approach, this paper also investigates the use of a purely model free approach, the (PDD) approach in valuing financial derivatives. With proper calibrations, the PDD approach is able to learn complex patterns within the data without any mathematical complexity often experienced with solving the governing PDEs. Similar to the PINN approach, the PDD is also limited in the sense that, some of the features that may be required for model building need to be approximated from the data, thereby introducing some biases. Furthermore, one need to determine which of the variables are driving the target variables more closely, the approach would therefore require fine-tuning through a trial and error technique.

The main contributions of this paper can be summarized as follows: it presents two systematic frameworks for pricing options, by directly processing and training market data based on appropriately selected features, second, we calibrate both models to observed market prices and implied volatilities, and evaluate their performance using error metrics such as absolute error and maximum absolute error, with comparisons drawn against the classical Black–Scholes model. Third, we provide a theoretical foundation for the convergence of the PINN approach by establishing conditions under which the network, trained using the Adam optimizer and initialized via the Xavier method, converges to a loss minimizing solution. Finally, we extended the PINN techniques to solving the BSMPDE and HSVM. The approaches herein averts the computational and mathematical complexities posed by using traditional numerical discritization techniques. Furthermore, to illustrate the effectiveness of PDD and PINN approaches, we conducted some numerical experiments showcasing the potentials of the two approaches in capturing complex financial dynamics and generating accurate option price predictions.

The rest of the paper is organized as follows. Section 2 presents some preliminaries on the neural network architecture proposed under the PDD approach. Section 3, presents the extension of the PINN approach to solving BSMPDE, while Section 4, present the extension of PINN approach to solving a HSVM. In Section 5, some numerical simulations, results and discussions are presented. Finally, in Section 6, we present concluding remarks as well as the scope for future research.

2. Mathematical preliminaries and methodology

In this section, the mathematical preliminaries of the PDD approach using a fully connected feedforward artificial neural network architecture are presented.

2.1. The PDD approach

The idea of a PDD approach for pricing options involves training a neural network model on a sufficiently large historical market dataset. The idea is for the model to learn the relationship between various market factors and the predicted option price. This approach does not rely on any governing PDEs as is the case in under the PINN approach. The PDD approach learn directly the mapping between the input factors in the data and the target variable from the data. According to literature, the accuracy of a neural network model predictions primarily depends on the quality of the training data and complexity of the neural network architecture, see for example [11,12].

During the training phase, the neural network is designed to grasp the intricate, nonlinear relationships between input variables and the target. The process involves reducing the loss function, which is a functional that quantifies the disparity between the predicted option prices and the desired/actual option price as per the training data.

Below is a general overview of the process involved in predicting option prices using a neural network under a PDD approach.

2.2. Network architecture under the PDD approach

For conceptual purpose of the PDD, consider a fully connected feedforward neural network with 5 input, 4 hidden layers and 1 output layer where the 4 hidden layers are made up of 10 neurons each. We denote the number of weights w in layer h by

$$w_h = n_h n_{h-1}, \tag{2.1}$$

where n_h is the number of neurons in layer h. On the other hand, we let the number of biases (b_h) in layer h to be equal to n_h.

According to [13], the number of trainable parameters in a neural network is computed as the sum of all weights and biases in the network. In our chosen optimal network, the weights were initialized using the Xavier initialization method, which is designed to maintain the variance of activations across layers, thereby mitigating issues such as vanishing or exploding gradients during training. Mathematically, the Xavier initialization sets the weights w of each layer as random variables drawn from a uniform distribution

$$U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right),$$

where n_{in} and n_{out} are the number of neurons in the input and output of the layer, respectively. Biases were initialized to zero to avoid introducing any initial directional bias into the predictions.

During training, the weights and biases are iteratively updated using the Adam optimization algorithm presented at the end of this section, which combines the advantages of adaptive learning rates and momentum to achieve efficient and stable convergence. The total number of trainable parameters in the network is computed as the sum of the weights and biases across all layers:

$$\begin{aligned} &5(10) + 10 + \sum_{h=2}^{4} \left(n_h n_{h-1} + n_h\right) + (n_{10})(1) + 1 \\ &\quad = 60 + 3\left((10)(10) + 10\right) + 10(1) + 1 = 401. \end{aligned} \tag{2.2}$$

The structure of the network is presented in Fig. 1 due to space limitation we are only presenting herein (for illustration purposes) a simplified version of the network which consist of two hidden layers and four neurons per hidden layer.

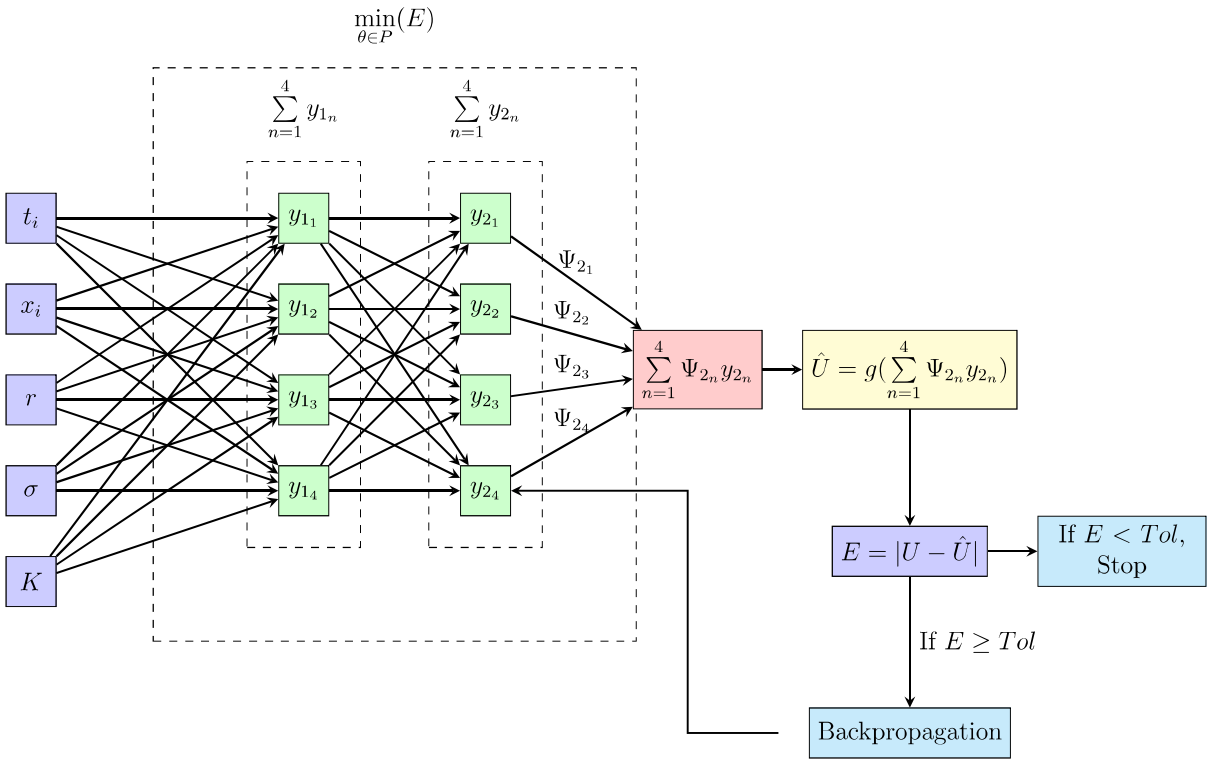


Fig. 1. Structure of feedforward neural network for PDD.

The activation function is taken at each node to be the Sigmoid or the Rectified Linear Unit ($ReLU$) function defined by

$$ReLU(y_{h_j}) = \max(y_{h_j}, 0), \tag{2.3}$$

whereby for each hidden layer,

$$y_h = \sum_{j=1}^{n} \left(tw_{h_j} + S\eta_{h_j} + \sigma\omega_{h_j} + r\rho_{h_j} + K\phi_{h_j} + \beta_{h_j} \right), \tag{2.4}$$

for $j = 1, 2, \ldots, n$, and n is the number of neurons. Generally, y_{h_j}, for $h = 1, 2 \ldots, m$, represent the output at each node in the neural network. The m and n, represent the number of hidden layers and neurons in each of hidden layers respectively. While w_{h_j} represent the weights of the temporal input variable in the hth hidden layer. The η_{h_j} on the other hand represent the weights of the spatial input variable in the h^{th} hidden layer. For a given European put option problem, ω_{h_j}, ρ_{h_j} and ϕ_{h_j} are the respective weights for volatility (σ), interest rate (r) and strike price's K inputs into the hth hidden layer. While β_h are the corresponding biases of the hth hidden layer.

Suppose $g(y)$ represent the desired output, then

$$g(y) = \begin{cases} \max(y, 0) & \text{if } y \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

Let X be the vector of inputs into the network, W_h the weight matrix of layer h with dimensions $n \times n$ for hidden layers, $n \times 1$ for the output layer, b_h the bias vector of layer h with dimension $n \times 1$ for hidden layers and scalar for the output layer. Furthermore, suppose Z_h is a linear transformation of the output of layer h given by

$$Z_h = W_h A_{h-1} + b_h, \tag{2.6}$$

where A_h is the activation output of layer h (after applying activation function on Z_h). Similar to Cervera et al. [8], we can modify the computation of the neural network as follow

$$\begin{aligned} \mathcal{N}(X, P) = {} & g_h \left(W_h g_{h-1} \left(W_{h-1} g_{h-2} \left(W_{h-2} \ldots W_h g_1 \left(W_1 X + b_1 \right) + b_2 \right) \right. \right. \\ & \left. \left. + \cdots + b_{h-1} \right) + b_h \right), \\ = {} & g_h \left(W_h A_h + b_h \right) \circ g_h \left(W_{h-1} A_{h-1} + b_{h-1} \right) \\ & \circ \ldots \circ g_1 \left(W_1 X + b_1 \right), \end{aligned} \tag{2.7}$$

such that, for the first hidden layer, we have the linear combination

$$Z_1 = W_1 X + b_1,$$

where the output at first hidden layer is transformed by $A_1 = g(Z_1)$.

For the hth hidden layer, we have

$$Z_h = W_h A_{h-1} + b_h,$$

where $A_h = g(Z_h)$. Finally, for the output layer we have

$$Z_{h+1} = W_{h+1} A_h + b_{h+1}, \tag{2.8}$$

and

$$\hat{U} = L(X, P) = A_{h+1} = g(Z_{h+1}). \tag{2.9}$$

The network computation errors are differentiated with respect to all the trainable parameters using the back propagation algorithm. The aim is to find a combination of $(W_h, W_{h-1}, \ldots, W_1, b_h, \ldots, b_1)$ which minimizes the mean squared error under the Euclidean norm given by

$$L \equiv \frac{1}{N_s N_t} \sum_{i=0}^{N_s} \sum_{j=0}^{N_t} \|U(S_i, t_j) - \hat{U}(S_i, t_j, P)\|_2^2, \tag{2.10}$$

where P is the set of parameters that needs to be optimized. Each trainable parameter is then updated using Adam optimization algorithm with learning rate l_r.

Proposition 2.1. *Let L be the loss function defined by Eq. (2.10) and let P be the set of trainable parameters. Assume that the derivative of the loss with respect to each trainable parameter exist and s bounded. Then we can compute the derivative of L with respect to each $\theta \in P$ so that the update of these parameters according to Adam optimization is as follows (see [14,15])*

$$\theta_{t_{j+1}} = \theta_{t_j} - l_r \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}, \tag{2.11}$$

where $m_{t_j} = \varphi_{t_j} m_{t_{j-1}} + (1-\varphi_{t_j})\dfrac{\partial L}{\partial \theta}$, $v_{t_j} = \rho_{t_j} v_{t_{j-1}} + (1-\rho_{t_j})\left(\dfrac{\partial L}{\partial \theta}\right)^2$, $\theta \in P$ *and* $\hat{m}_{t_j} = \dfrac{m_{t_j}}{1-\varphi^t_{t_j}}$ *and* $\hat{v}_{t_j} = \dfrac{v_{t_j}}{1-\rho^t_{t_j}}$ *are the bias-corrected first and second moments, respectively.*

Proof. To realize this, we need to evaluate the derivative of the loss with respect to each trainable parameter, i.e., with respect to each entry of each W_h and b_h, for $h = 1, 2, \ldots, m$. In this paper, we shorten our explanation by demonstrating this process via a batched optimization technique whereby the weights and biases at each layer are updated simultaneously.

For each epoch and the entire training set, the gradient of the loss with respect to each trainable parameter vector at each layer is calculated. Note that the mean square error in a neural network training scenario is usually calculated over the entire training set in each epoch. For training over multiple epochs, the loss function is computed repeatedly at the end of each epoch to assess the performance of the model and guide optimization. From Eq. (2.10) we have

$$\left\{\nabla_{b_h} L\right\}_{h=1}^{m} = -\frac{2}{N_s N_t}\sum_{i=0}^{N_s}\sum_{j=0}^{N_t} \|U(S_i,t_j) - \hat{U}(S_i,t_j,P)\|_2 \cdot \left\{\nabla_{b_h}\hat{U}(S_i,t_j,P)\right\}_{h=1}^{m}, \tag{2.12}$$

where b_h is itself a topple of n biases at layer $h \le m$ and $\nabla_{b_h} L$ is the gradient of the loss with respect to b_h. Similarly, for the weights matrix at each layer, the Jacobian of the loss with respect to the weighted matrix at each layer denoted $J_W(L)$ is given by

$$J_W(L) = -\frac{2}{N_s N_t}\sum_{i=0}^{N_s}\sum_{j=0}^{N_t} \|U(S_i,t_j) - \hat{U}(S_i,t_j,P)\|_2 \times \begin{bmatrix} \frac{\partial \hat{U}}{\partial W_{11}} & \frac{\partial \hat{U}}{\partial W_{12}} & \cdots & \frac{\partial \hat{U}}{\partial W_{1n}} \\ \frac{\partial \hat{U}}{\partial W_{21}} & \frac{\partial \hat{U}}{\partial W_{22}} & \cdots & \frac{\partial \hat{U}}{\partial W_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{U}}{\partial W_{m1}} & \frac{\partial \hat{U}}{\partial W_{n2}} & \cdots & \frac{\partial \hat{U}}{\partial W_{mn}} \end{bmatrix}, \tag{2.13}$$

where $-\frac{2}{N_s N_t}\sum_{i=0}^{N_s}\sum_{j=0}^{N_t} \|U(S_i,t_j) - \hat{U}(S_i,t_j,P)\|_2 \frac{\partial \hat{U}}{\partial W_{m,n}}$ is the derivative of the loss with respect to the (m,n) entry of matrix W_h.

More specifically,

$$\nabla_{W_h}\hat{U}(S_i,t_j,P) = \frac{\partial \hat{U}}{\partial W_h} = \left(\frac{\partial \hat{U}}{\partial W_{h,1}} \cdots \frac{\partial \hat{U}}{\partial W_{h,n}}\right),$$

$h = 1, 2, \ldots, m$, is computed as follows: The weights W_h are matrices, and each weight $W_{k,l}$ connects the lth nodes from layer $h-1$ to the kth node in layer h. The gradient of the loss with respect to W_h is computed similarly to the bias gradient. Using the chain rule, we have

$$\begin{aligned} \frac{\partial \hat{U}(S_i,t_j)}{\partial W_h} &= \frac{\partial g_n}{\partial g_{n-1}} \cdots \frac{\partial g_{h+1}}{\partial g_h} \cdot \frac{\partial g_h}{\partial W_h} \\ &= \frac{\partial g_n}{\partial g_{n-1}} \cdots \frac{\partial g_{h+1}}{\partial g_h} \cdot \frac{\partial g_h}{\partial Z_h}\frac{\partial Z_h}{\partial W_h} \\ &= \frac{\partial g_n}{\partial g_{n-1}} \cdots \frac{\partial g_{h+1}}{\partial g_h} \cdot A^T_{h-1} \cdot g'_h(Z_h), \\ \implies \frac{\partial L}{\partial W_h} &= -\frac{2}{N_s N_t}\sum_{i=0}^{N_s}\sum_{j=0}^{N_t} \|U(S_i,t_j) - \hat{U}(S_i,t_j,P)\|_2 \frac{\partial g_n}{\partial g_{n-1}} \cdots \frac{\partial g_{h+1}}{\partial g_h} \\ &\quad \cdot A^T_{h-1} \cdot g'_h(Z_h), \end{aligned}$$

where A_{h-1} is the activation of layer $h-1$, and $g'_h(Z_h)$ is the derivative of the activation function with respect to its input Z_h as presented earlier in this section.

The gradient with respect to b_h can be interpreted as the sum of the errors at each node in layer h, weighted by the derivative of the activation function g_h. The gradient with respect to W_h is the error term multiplied by the activation from the previous layer. In both cases, the backpropagation terms represent the errors that are propagated from the output layer back to layer h, adjusting the gradients as the weights and biases are updated during optimization.

The learning rate l_r determines how fast or slow the model should learn from the data. A smaller learning rate will result in slow convergence but may provide more accurate results, while a larger learning rate will result in faster convergence but may cause the optimization process to oscillate around the minimum or even diverge [11]. Hence, we will tune the parameters appropriately. Detail about the tuning of parameters can be found in [11,12] just to name a few.

The PDD approach is limited by its reliance on the quality and relevance of training data, which can be biased and influenced by market conditions [1]. Additionally, not all financial derivatives have adequate historical data, and companies may modify contracts to optimize their returns, often lacking historical data but may have prevailing model or PDE. Therefore, a partially data-dependent method like PINN, which relies on the governing PDE and available data may be advantageous.

3. Methodology for the PINN approach

Unlike the PDD approach presented in Section 2.2, the PINN generally depends on the governing PDE to make predictions. The partial derivatives in the governing PDE are computed with respect to these input parameters and also with respect to all the trainable parameters (weights and biases) which are initialized and updated as presented in Section 2.2 above. Also, the nature, structure, dimension of the input data may differ from the ones presented under the PDD approach. In the PINN approach however, the trainable parameters are also updated using the back-propagation algorithm described under the PDD approach. Using Neural Networks, we extend the approach for solving BSMPDE to more complicated PDEs arising in finance where the exact solutions do not exist, under which discretization based numerical methods may fail to perform accurately or do so with very high computational costs.

To establish the existence of a solution to a PDE using neural networks, we consider the Kolmogorov and Cybenko theorem [16]:

Theorem 3.1. *For any continuous function $f(x)$ defined on a compact input domain X, there exists a feedforward neural network with a single hidden layer, n neurons and activation function $g(x)$, such that the network's output $L(x)$ satisfies*

$$|g(x) - L(x)| < \epsilon$$

for all $x \in X$ and some $\epsilon > 0$, provided that n is sufficiently large.

Proof. The proof of this theorem is presented in Appendix A.4.

Definition 3.2. Let $P^* = \{\alpha, w, \eta, \beta\}$, be the set of trainable parameters that minimizes $\frac{1}{N_s N_t}\sum_{i=1}^{N_s}\sum_{j=1}^{N_t} \|U - \hat{U}\|_2^2$ at the first layer, with α, w, η, β as defined in Section 2.2, the network computation at layer one can be defined as

$$Net(S,t,P^*) = \sum_{j=1}^{n}\sum_{k=1}^{m}\left(\alpha_{jk} g\left(tw_{jk} + S\eta_{jk} + \beta_k\right)\right), \tag{3.1}$$

and at subsequent layers ($h \ge 2$), the network computation can be obtained from Eq. (2.7).

Let $f(s,t)$ be the residual of the Black–Scholes equation. The total cost during the training process can be defined as

$$f(t,s) = U_t + \frac{1}{2}s^2\sigma^2 U_{ss} + rsU_s - rU, \tag{3.2}$$

The approach used in finding the optimal set of parameters P^* can be found in [1,8] and some references therein. The mean square error is used to solve the minimization problem. That is, we find P^* such that

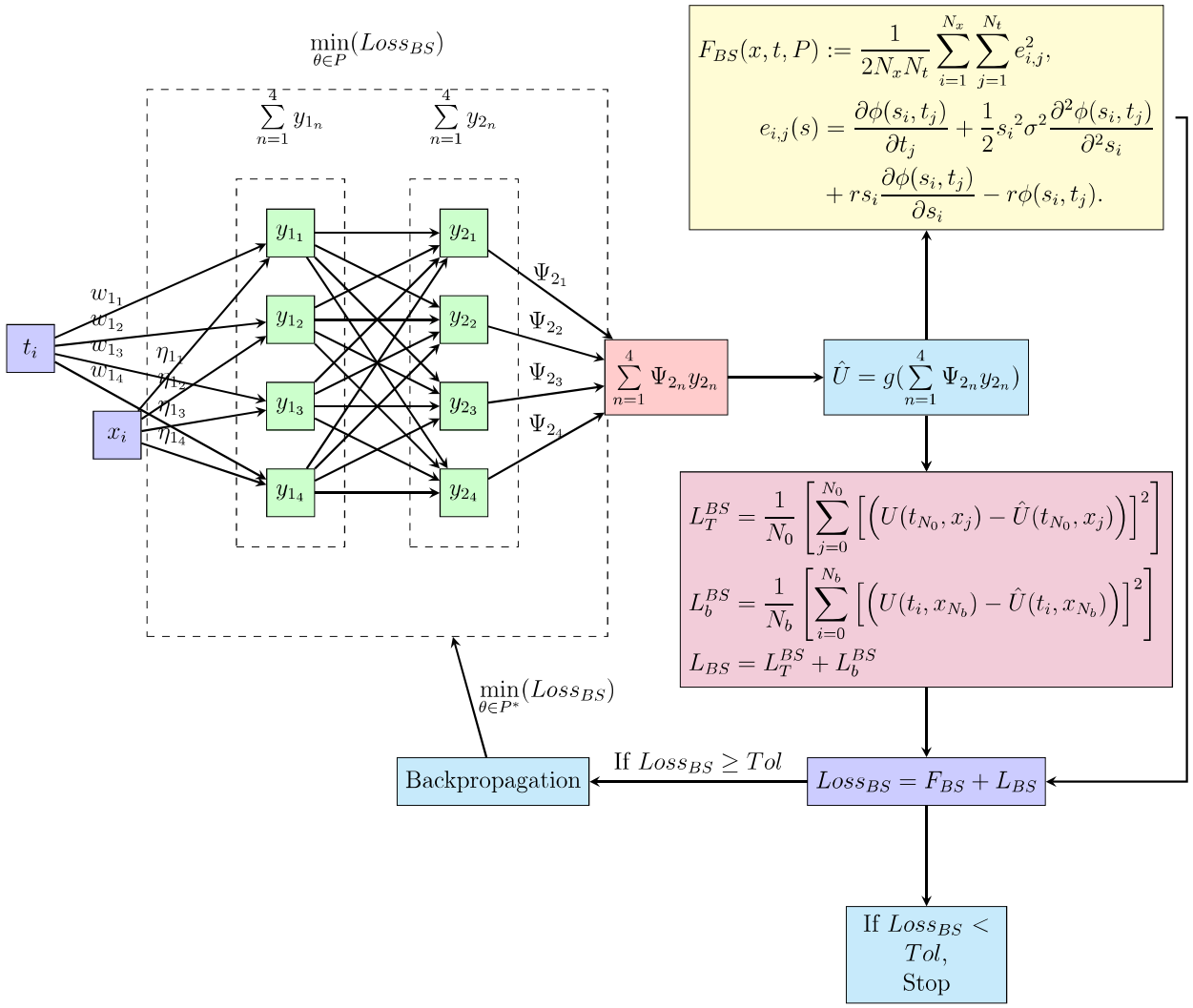


Fig. 2. PINN structure for solving the BSPDE with 2 hidden layers, 4 neurons per hidden layer and $i = 0, 1, 2, \ldots, N; N \in \{N_0, N_b, N_r\}$.

$F(s,t,P^*) = \min F(s,t,P)$ where F represent the cost function defined as

$$F_{BS}(s,t,P) = \frac{1}{N_s N_t}\sum_{i=1}^{N_s}\sum_{j=1}^{N_t} e_{ij}{}^2, \tag{3.3}$$

$$e_{i,j} = \frac{\partial\phi(s_i,t_j)}{\partial t_j} + \frac{1}{2}s_i{}^2\sigma^2\frac{\partial^2\phi(s_i,t_j)}{\partial^2 s_i} + rs_i\frac{\partial\phi(s_i,t_j)}{\partial s_i} - r\phi(s_i,t_j).$$

Here, $i = 1, 2, 3 \ldots, N_r$, and $j = 1, 2, 3 \ldots, N_r$, and N_r is the number of collocation points in the interior of the domain. The boundary and terminal loss under the Black–Scholes model can be computed using the loss at boundary (L_b^{BS}) plus loss at terminal (L_T^{BS}) as

$$\begin{aligned} L_{BS} &:= L_T^{BS} + L_b^{BS} \\ &= \frac{1}{N_0}\sum_{j=0}^{N_0}\left[\left(U(s_j,t_{N_0}) - \hat{U}(s_j,t_{N_0})\right)\right]^2 \\ &+ \frac{1}{N_b}\sum_{i=0}^{N_b}\left[\left(U(s_{N_b},t_i) - \hat{U}(s_{N_b},t_i)\right)\right]^2. \end{aligned} \tag{3.4}$$

The total training loss for training the classical BSPDE using PINN is thus given by

$$Loss_{BS} := F_{BS} + L_{BS}. \tag{3.5}$$

For visual clarity, consider a PINN with 2 inputs, 2 hidden layers and 4 neurons per hidden layer as shown in Fig. 2:

In this section, we have examined the methodology for computing the loss function when solving the classical Black–Scholes PDE using PINN. A flow diagram of the network's architecture is presented. In the next section, we present the loss function when solving the HSVM using PINN.

4. PINN for Heston stochastic volatility model

In this section we extend the PINN approach to the HSVM. The motivation for this model lies in the fact that, the HSVM is widely used in practice due to it ability to capture volatility smiles and returns skewness observed in financial markets. Furthermore, this model is able to generate realistic dynamics for underlying assets and their volatilities, features such as mean reversion and long-term persistence in volatility can easily be modeled.

4.1. Loss function for the HSVM

Let $S(t)$ be the stock price at time t, $v(t)$ be the instantaneous variance or volatility of the stock price, κ the rate of mean reversion, θ the long-term mean of the variance, σ the volatility of the variance process, r the risk-free interest rate, $W_1(t)$ and $W_2(t)$ Brownian motions, ρ the correlation between $W_1(t)$ and $W_2(t)$ satisfying

$$dW_1^{\mathbb{Q}}(t)dW_2^{\mathbb{Q}}(t) = \rho^{\mathbb{P}}dt,$$

where $\mathbb{Q}$ is the risk neutral measure and $\mathbb{P}$ is the real world or probability measure. Therefore, under the HSVM, we assumes the stock price and the instantaneous variance follows the following stochastic processes respectively,

$$dS(t) = rS(t)dt + \sqrt{v(t)}S(t)\mathrm{d}W_1(t), \tag{4.1}$$

$$dv(t) = \kappa(\theta - v(t))dt + \sigma\sqrt{v(t)}\mathrm{d}W_2(t). \tag{4.2}$$

Note that when changing the measure, the correlation coefficient remains fixed, i.e., $\rho^{\mathbb{P}} = \rho^{\mathbb{Q}}$. Similarly under a change of measure, $\sigma^{\mathbb{P}} = \sigma^{\mathbb{Q}}$.

Let $U = U(S, v, t)$, be the price of a European option on a stock with price S, variance v and time t, then the HSVM is given by

$$\frac{\partial U}{\partial t} + rS\frac{\partial U}{\partial S} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + (\kappa(\theta - v))\frac{\partial U}{\partial v} + \frac{1}{2}\sigma^2 v\frac{\partial^2 U}{\partial v^2} + \rho\sigma vS\frac{\partial U}{\partial S\partial v} - rU = 0. \tag{4.3}$$

Let f_{HS} be the residual function defined as

$$f_{HS}(S, v, t) = \frac{\partial U}{\partial t} + rS\frac{\partial U}{\partial S} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + (k(\theta - v))\frac{\partial U}{\partial v} + \frac{1}{2}\sigma^2 v\frac{\partial^2 U}{\partial v^2} + \rho\sigma vS\frac{\partial U}{\partial S\partial v} - rU. \tag{4.4}$$

Let Ω be the set of all trainable parameters. Suppose that $(S, v, t) \in [0, \infty) \times [0, \infty) \times [0, T)$ and that $P^* \subset \Omega$ is the set of optimal parameters that minimizes

$$f_{HS}(S, v, t, P) = \frac{1}{N_s N_v N_t}\sum_{i=1}^{N_s}\sum_{j=1}^{N_v}\sum_{p=1}^{N_t}\left(e_{ijp}^{HS}\right)^2, \tag{4.5}$$

where

$$e_{i,j,p}^{HS} = \frac{\partial U_{ijp}}{\partial t_j} + rS_i\frac{\partial U_{ijp}}{\partial S_i} + \frac{1}{2}v_p {S_i}^2\frac{\partial^2 U_{ijp}}{\partial {S_i}^2} + \left(k(\theta - v_p)\right)\frac{\partial U_{ijp}}{\partial v_p} + \frac{1}{2}\sigma^2 v_p\frac{\partial^2 U_{ijp}}{\partial {v_p}^2} + \rho\sigma v_p S_i\frac{\partial U_{ijp}}{\partial S_i\partial v_p} - rU_{ijp}, \tag{4.6}$$

with N_s, N_v and N_t the number of collocation points in each direction on the computational grid. The loss is computed as a simple sum of the loss at boundary (L_b^{HS}) plus loss at terminal (L_T^{HS}) plus the loss within the interior grid as

$$Loss_{HS} = f_{HS} + L_b^{HS} + L_T^{HS} \tag{4.7}$$

where

$$L_T^{HS} = \frac{1}{N_0}\sum_{j=0}^{N_0}\left[\left(U(s_j, v_0, t_{N_0}) - \hat{U}(s_j, v_0, t_{N_0})\right)\right]^2, \tag{4.8}$$

and

$$L_b^{HS} = \frac{1}{N_b}\sum_{i=0}^{N_b}\left[\left(U(s_{N_b}, v_i, t_i) - \hat{U}(s_{N_b}, v_i, t_i)\right)\right]^2. \tag{4.9}$$

Having established the loss function for solving the HSVM using PINN approach, the subsequent section presents the convergence analysis of the PINN.

4.2. Convergence analysis of PINN

In this section, we show that the parameterized domain Ω of the neural network space $\mathcal{N}$, is convex. Furthermore, we show that the set of all trainable parameters is bounded in Ω under Adam optimization, i.e., $\|W_h - W^*\|_{C^k(\Omega)} < \epsilon$ for some $0 < \epsilon < \infty$, based on the assumption that the $\frac{\partial Loss}{\partial \theta}$ where $\theta \in \Omega = \{W_1, W_2, \ldots, W_H, b_1, b_2, \ldots, b_H\}$ is bounded, see, e.g., [17,18] for more assumptions governing the convergence of neural networks. We further show that under the sigmoid activation function, the loss function is bounded and convex at least in some subset of Ω. Finally, since Ω is bounded, by the Bolzano Weierstrass theorem, there exist at least a convergent subsequence and more so, the sequence of minimizers converges to the true solution of the PDE for any arbitrary number of hidden layers (i.e., the neural network space is dense in the set of all continuous functions that define the solution of the option).

Proposition 4.1. *Let $\mathcal{N}$ be a family of feedforward neural networks parameterized by a finite domain Ω. Let Ω be initialized by the Xavier initialization, then Ω is convex.*

Proof. The main steps of the proof are outlined here, with detailed explanations provided in Appendix A.2.

Let Ω represent the set of trainable parameters. Let $w_i, w_j \in \Omega^0$, and $\lambda \in [0, 1]$, we show that $w_\lambda = \lambda w_i + (1 - \lambda)w_j \in \Omega^0$ where Ω^0 is an initialized subset of Ω.

$$w_i \geq -\sqrt{\frac{6}{n_{j-1} + n_j}} \implies \lambda w_i \geq -\lambda\sqrt{\frac{6}{n_{j-1} + n_j}}, \tag{4.10}$$

$$w_j \geq -\sqrt{\frac{6}{n_{j-1} + n_j}} \implies (1 - \lambda)w_j \geq -(1 - \lambda)\sqrt{\frac{6}{n_{j-1} + n_j}}. \tag{4.11}$$

Adding the two equations gives

$$\lambda w_i + (1 - \lambda)w_j = w_\lambda \geq -\lambda\sqrt{\frac{6}{n_{j-1} + n_j}} - (1 - \lambda)\sqrt{\frac{6}{n_{j-1} + n_j}} = -\sqrt{\frac{6}{n_{j-1} + n_j}}. \tag{4.12}$$

Similarly,

$$\lambda w_i + (1 - \lambda)w_j = w_\lambda \leq \lambda\sqrt{\frac{6}{n_{j-1} + n_j}} + (1 - \lambda)\sqrt{\frac{6}{n_{j-1} + n_j}} = \sqrt{\frac{6}{n_{j-1} + n_j}}, \tag{4.13}$$

and

$$-\sqrt{\frac{6}{n_{j-1} + n_j}} \leq w_\lambda \leq \sqrt{\frac{6}{n_{j-1} + n_j}} \implies w_\lambda \in \left[-\sqrt{\frac{6}{n_{j-1} + n_j}}, \sqrt{\frac{6}{n_{j-1} + n_j}}\right]. \tag{4.14}$$

Therefore, $\Omega^0 \subset \Omega$ is convex. □

Proposition 4.2. *When PINN is trained using Adam optimization, assuming that the derivative of the loss function with respect to each trainable parameter is bounded, the set of trainable parameters is bounded, i.e., $\forall \theta \in \Omega, \exists N \leq N_{epoch}\ |, \|\theta - \theta^*\|_{C^k(\Omega)} < \infty$, where θ^* is the optimal value for θ in Ω.*

Proof. See Appendix A.3 for a complete proof of Proposition 4.2. Here we present only the main points in the proof.

For each hyperparameter $\theta_t \in \Omega$, compute the gradients $Loss_\theta$ of the objective function with respect to the parameter θ_t at time t as

$$Loss_{\theta_{t_j}} = \frac{\partial Loss_{t_j}}{\partial \theta_{t_j}},\ j = 1, 2, \ldots, N; N \in \{N_0, N_b, N_r\}. \tag{4.15}$$

Assuming that the loss function $Loss_{\theta_{t_j}}$ is convex in some domain, then $\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}$ is also convex in that domain, and by Theorem A.2,

$$loss_m^v(\theta_{t_j}) - loss_m^v(\theta^*) \leq \left(\theta_{t_j} - \theta^*\right)^T \nabla loss_m^v(\theta_{t_j}). \tag{4.16}$$

Substitution Eq. (A.28) into Eq. (A.27) we have

$$loss_m^v(\theta_{t_j}) - loss_m^v(\theta^*) \leq \frac{\left(\theta_{t_j} - \theta^*\right)^2}{2l_r} - l_r\left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2. \tag{4.17}$$

From Eq. (A.33), $\hat{M}_{t_j}$ and $\sqrt{\hat{V}_{t_j}}$ are bounded and hence $l_r\left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}}+\epsilon}\right) \leq 1$. Therefore,

$$\|\theta_{t_{j+1}} - \theta_{t_j}\|_{W^{2,2}(\Omega)} \leq l_r. \tag{4.18}$$

From Eq. (A.30),

$$\|Loss(\theta_{t_j}) - loss(\theta^*)\|_{W^{2,2}(\Omega)} \leq \frac{l_r}{2} + \frac{l_r}{2} = l_r. \tag{4.19}$$

Summing over the sequence of parameters gives

$$\sum_{j=1}^{T} \|Loss(\theta_{t_j}) - loss(\theta^*)\|_{W^{2,2}(\Omega)} \leq l_r T. \quad \square \tag{4.20}$$

Theorem 4.3. *Let $\mathcal{N}(X,\theta)$ be a neural network space with inputs X and parameters θ. Let g be the sigmoid activation function and suppose the networks architecture has l hidden layers denoted $h_1, h_2, \ldots, h_l$, then the loss function defined as $loss_{t_j} = \frac{1}{2}\|U(X) - \hat{U}(X,\theta_{t_j})\|^2$ for $\hat{U}(X,\theta_{t_j}) \in \mathcal{N}(X,\theta)$ is convex if and only if the linear combination of the weights at the lth hidden layer is negative.*

Proof. Let $g(x) = \frac{1}{1+e^{-x}}$, then

$$g''(x) = (g'(x) - (g(x))^2)(1 - g(x)) = \frac{(e^{-x})^2 - e^{-x}}{(1-e^{-x})^3} > 0, \ \forall x < 0. \tag{4.21}$$

For all $x > 0, g''(x) < 0$ and $g''(x) = 0$ for all $x = 0$. This proves that $g(x)$ is convex for $x < 0$. We proof that the loss function $loss_{t_j} = \frac{1}{2}\|U(X) - \hat{U}(X,\theta_{t_j})\|^2$ is convex if and only if the linear combination of the weights at the lth hidden layer is negative.

$$\begin{aligned}
&\text{loss}_{t_j} = \frac{1}{2} \parallel U(X) - g_l\left(W_l g_{l-1}\left(W_{l-1} g_{l-2}\left(W_{l-2} \ldots W_2 g_1\left(W_1 X + b_1\right)\right.\right.\right.\\
&\left.\left.\left. + b_2\right) + \cdots + b_{l-1}\right) + b_l\right) \parallel^2,\\
&\implies \frac{\partial^2 \text{loss}_{t_j}}{\partial \theta_j^2} > 0, \forall \theta_j \in \left\{\{W_k\}_{k=1}^j, \{b_k\}_{k=1}^j\right\},\\
&\iff \frac{\partial^2 g_l}{\partial \theta_j^2} > 0,\\
&\iff w_l . g_{l-1}(*) < 0.
\end{aligned}$$

Notice that $w_l . g_{l-1}(*)$ is a linear combination of all the weights from layer h_{l-1} to h_l where the scalars are the outputs at each neuron in layer h_{l-1}. Once this is negative, the second derivative of the output activated by the sigmoid activation layer is positive. $\square$

Let dim_{in} be the dimension of the input space of the neural network and dim_{out} the dimension of the output space of the neural network. Let $\|u\|_{C^k(\Omega)}$ denotes the norm of the function u in the space $C^k(\Omega)$, which is the space of functions that have continuous derivatives up to order k on the domain Ω. For a function u in $C^k(\Omega)$, the norm is defined as

$$\|u\|_{C^k(\Omega)} = \sum_{|\gamma| \leq k} \sup_{x \in \Omega} \left|\frac{\partial^{|\gamma|} u(x)}{\partial x^\gamma}\right|, \tag{4.22}$$

where γ is a multi-index and x^γ represents partial derivatives.

In De Ryck et al. [19], the authors should that a family of neural networks with two hidden layers is dense in $C^1(\Omega;\mathbb{R}^2)$. In Doumeche et al. [20], the authors generalized the work of De Ryck et al. to any bounded Lipschitz domain and to any number of hidden layers (≥ 2) and finally to any output dimension d_2. In this paper, we show as an example the extension of the proof of Doumeche et al. in [20] to option pricing. A sequence of PINN trained to predict option prices is densed in $C^\infty\left([0,\infty)(\times 0,\infty)\times 0,T]\times\mathbb{R}^n;\mathbb{R}^{dim_{out}}\right)$, i.e., for any function $U \in C^\infty\left(\overline{[0,\infty)(\times 0,\infty)\times 0,T]\times\mathbb{R}^n};\mathbb{R}^{dim_{out}}\right)$ of functions that maps the options parameters to its option value, there exist a neural network sequence $\left(\hat{U}_p\right)_{p=1}^{N_epoch}$ with $\hat{U}_p \in C^2(\Omega;\mathbb{R}^{dim_{in}})$ under the $C^k(\Omega)$ norm such that $\|U - \hat{U}_p\|_{C^k(\Omega)} = 0$, where $\hat{U}_p$ depends on the set of trainable parameters.

Theorem 4.4. *Let $k \in \mathbb{N}$, $h \geq 2$ and let $\Omega \subset \mathbb{R}^{dim_{in}}$ be a bounded Lipschitz domain for the trainable parameters, then the family of PINN, denoted by $\mathcal{N}$, that are minimizers of the loss function for approximating the solution to the HSVM is dense in the function space $C^2(\Omega;\mathbb{R}^{dim_{out}})$ under the $C^k(\Omega)$ norm.*

Proof. The proof of Theorem 4.4 can be found in Appendix A.5.

The studies conducted by [17–20], and [21] demonstrate that feed-forward neural networks converge over bounded Lipschitz domains. Specifically, Theorem 4.4 establishes that the PINN method reliably converges to the desired solution when applied to the BSPDE and HSVM, provided the network is trained using the Adam optimization algorithm with sigmoid activation functions facilitating smooth gradient flow. Furthermore, in Section 5.1, Table 9, we show that the PINN model converges with a rate between $\frac{1}{2}$ and 1.

4.3. Data generation and processing

This section outlines the general procedure used to generate the data for this study. Specific datasets corresponding to each experiment are presented in Section 5.

An interest rate of 3.97%, as recorded on April 29, 2025, was obtained from the Board of Governors of the Federal Reserve System (US), based on the Market Yield on U.S. Treasury Securities at 1-Year Constant Maturity [DGS1], retrieved via FRED (Federal Reserve Bank of St. Louis): https://fred.stlouisfed.org/series/DGS1. Using Python libraries such as yfinance, we downloaded and normalized AAPL stock prices, taking the closing price on 29, 2025, as the spot price. For theoretical analysis, log returns were computed, and their volatility measured as 2.5138 was used as the theoretical volatility. However, in practical experiments, implied volatility was employed, as it yielded more accurate option price predictions. The time to maturity was set to 30 days.

Since we could not obtain a sufficient large option chain from Yahoo Finance, using the small dataset would have been limited for training PINN, we therefore took the minimum price as x_{min} and the maximum price as x_{man} and then produce a high frequency data by descretizing the domain $[x_{min}, x_{mxn}]$ into a suitably large number of prices. Similar discretization is done for the volatility process. Mathematically we have

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

Using the time to maturity $T = 0.0822$, which corresponds to 30 days, we generate data at the initial, boundary and interior of the time domain 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, then

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

respectively. The initial stock price is given by

$$x_{0_{i+1}} = x_{0_i},\ \forall i : i = 0,1,2,\ldots,N_0. \tag{4.23}$$

Specifically, in order to obtain a clean data that theoretically represent the Heston model, the initial instantaneous variance or initial volatility is taken as 2.5. For each q, $q = 1, 2, \dots, N$, $N \in \{N_b, N_r\}$ we obtained a bivariate normal distribution for Wiener processes given by

$$\begin{bmatrix} W_1(t) \\ W_2(2) \end{bmatrix} \sim L\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right) \sqrt{\Delta t}, \ \Delta t \in \{\Delta t_b, \Delta t_r\}, \tag{4.24}$$

at the boundary and interior grid, respectively. With this construction, the solution component of (4.1)–(4.2) is

$$v_{t+1} = \left| v_t + \kappa(\theta - v_t)\Delta t + \xi \sqrt{v_t} W_2(t) \right|, \tag{4.25}$$

and

$$S_{t+1} = S_0 \exp\left(\left(r - \frac{1}{2} v_t \right) \Delta t + \sqrt{v_t} W_1(t) \right). \tag{4.26}$$

With the vectors $\{S_0, S_b, S_r, v_0, v_b, v_r, t_0, t_b, t_r\}$ generated, the initial and boundary conditions become

$$\begin{aligned} U(x_0, v_0, t_0) &= \max\left(Ke^{-rt_0} - x_0, 0\right), \\ U(x_b, v_0, t_b) &= 0. \end{aligned}$$

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

$$(X_{data})_{N_0 \times 3}([x_{min}, x_{max}] \times [v_{min}, v_{max}] \times [t_{min}, t_{max}]) = \left[S_0, \ v_0, \ t_0\right], \tag{4.27}$$

and

$$(X_{data})_{N_b \times 3}([x_{min}, x_{max}] \times [v_{min}, v_{max}] \times [t_{min}, t_{max}]) = \left[S_b, \ v_b, \ t_b\right], \tag{4.28}$$

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

$$(U_{data})_{N_b \times 2} = \left[U_0, \ U_b\right], \text{where } N_b = N_0. \tag{4.29}$$

Finally, in the interior of the domain,

$$(X_{data})_{N_r \times 2}([x_{min}, x_{max}] \times [v_{min}, v_{max}] \times [t_{min}, t_{max}]) = \left[S_r, \ v_r, \ t_r\right]. \tag{4.30}$$

Note that, N_0 and N_b should be equal to avoid mismatch in dimensions due to concatenation.

To avoid scalability issues, the data is standardized and then normalized as follows. For Q in the domain of S or V, the standardized value of Q is given by

$$Q_s = \frac{Q - \mu_Q}{\sigma_Q}, \tag{4.31}$$

and the normalized value of Q_s is

$$Q_N = \frac{Q_s - Q_{s_{min}}}{Q_{s_{max}} - Q_{s_{min}}}, \tag{4.32}$$

where, μ_Q and σ_Q are the mean and standard deviation of each batch of Q, respectively. Finally, the data is transformed back to the original as follow,

$$Q_s = Q_N(Q_{s_{max}} - Q_{s_{min}}) + Q_{s_{min}}, \tag{4.33}$$

and

$$Q = Q_s \sigma_Q + \mu_Q. \tag{4.34}$$

For more details on data normalization, standardization and scalability issues, see for example, [22,23] and references therein.

To check the efficiency of the PDD and PINN approach compared to the market data and 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 exact solution at each point in time, i.e.,

$$MAE = \max(\|\hat{U}_i - U_i\|), \ i = 0, 1, 2, \dots, N; \ N \in \{N_0, N_b, N_r\}, \tag{4.35}$$

where $\hat{U}$ is the approximate payoff solution and U is the exact solution from observed market data.

5. Practical and theoretical experiments

In this section, we provide the numerical results and discussions for the PDD approach outlined in Section 2, the PINN approach applied to the Black–Scholes equation as described in Section 3, and the PINN approach applied to the HSVM discussed in Section 4. All simulations in this paper were performed on a Windows 10 Pro Version $22H2$, $12th$ Gen Intel(R) Core(TM) $i7-1255U$, 1.70 GHz processor, 16 GB RAM and a 64-bit operating system. We used python version 3.12.7 as the computing software and Tensorflow version 2.19 as the deep learning framework.

5.1. Data pre-processing, simulations and discussions under the PDD approach

In order to train a PDD, the nature of data depends on the nature of the expected output. One could make predictions by passing a complete set of new variables each time a prediction is to be made. In which case, the neural network most be trained on a varying set of all parameters so that the model learns the variation in all input parameters. In this scenario the input features used for the PDD in this study is generated as

$$\begin{aligned} S &\sim \mathcal{U}[150, 250], \\ K &\sim \mathcal{U}[100, 300], \\ t &\sim \mathcal{U}[0.1, 1], \\ r &\sim \mathcal{U}[0.01, 0.05], \\ \sigma &\sim \mathcal{U}[0.1, 0.5], \end{aligned}$$

where $\mathcal{U}$ is an independent uniform distribution over the specified interval, repeated for N samples.

Secondly we may wish to study the behavior of option prices by keeping other parameters fixed and finding out how the model behaves with a variation in one parameter, in which case we create vectors of the fixed variables as follows.

- Set total number of data points to N and generate the strike price vector $K[N, 1]$, the volatility vector $\sigma[N, 1]$, and the risk free interest rate vector $r[N, 1]$, respectively.
- Create a data frame whose columns are the variables (stock price, time to maturity, the strike price, the interest rate, and the volatility).
- Calculate the option value based on the model under consideration and add the column to the data frame. If market data is available, this column should be the observed option prices.

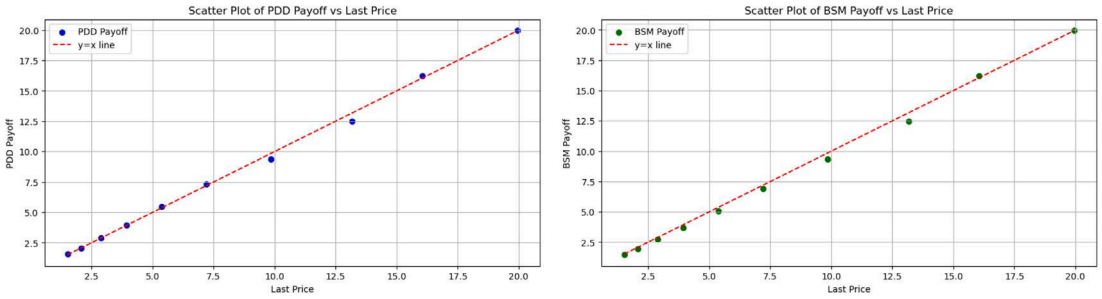
The data set is split into the training and the test set and feature engineering is performed on train set. The split ratio is $8 : 2$. The training set is thus a data frame of dimension $(0.8N \times 6)$ and the test set a data frame of dimension $(0.2N \times 6)$ where the last column is the target variable (i.e., option value). For more details on data processing and feature engineering see, e.g., [11,24].

A plot of the training loss against the number of epochs (i.e., the number of complete passes made during training) is presented in Fig. 4(b). The graph shows that the training loss is converging to zero as the number of complete passes increases as presented in Theorem 4.4.

The overall performance of the neural network is validated by assessing its fit on the test set, and further calibrating the PDD model to market prices ensuring that it generalizes accurately without overfitting.

It is well established in the literature that, when evaluating the effectiveness of option pricing models, researchers commonly employ either the Last prices on an option chain or the average of the bid and ask prices for each trading period as proxies for the true option prices (see, for example, [25,26] and references therein).

In this study, we have used the absolute errors between implied volatilities estimated from the predicted prices and that from the market. We thus explain at this stage how our estimated implied volatilities are obtained.



(a) Scattered plot of the Last price Vs PDD payoff (b) Scattered plot of the Last price Vs BSM payoff

Fig. 3. The scattered plots of the predicted against the last price on an AAPL expiring on the as priced on the .

Remark 5.1. Given implied volatilities σ_i at strikes K_i, $i = 0, 1, 2$, approximate $\sigma(K)$ by

$$\sigma(K) = \sum_{i=0}^{2} \sigma_i \, L_i(K), \tag{5.1}$$

where the Lagrange basis polynomials are

$$L_i(K) = \prod_{\substack{j=0 \\ j \neq i}}^{2} \frac{K - K_j}{K_i - K_j}, \quad i = 0, 1, 2. \tag{5.2}$$

We use a 1-dimensional interpolation to obtain an implied volatility value $\hat{\sigma}$ from a predicted option price $\hat{P}$. The logic is to find the volatility value $\hat{\sigma}$ such that

$$\hat{P} \mapsto \hat{\sigma}, \tag{5.3}$$

using interpolation on the known relation (P_i, σ_i), i.e., market price and market implied volatilities. We use a quadratic interpolation function. Given $n = 3$, neighboring points $(P_0, \sigma_0), (P_1, \sigma_1), (P_2, \sigma_2)$, the quadratic Lagrange interpolant is

$$\sigma(P) = \sigma_0 \cdot \ell_0(P) + \sigma_1 \cdot \ell_1(P) + \sigma_2 \cdot \ell_2(P), \tag{5.4}$$

where the Lagrange basis polynomials $\ell_j(P)$ are defined as

$$\ell_0(P) = \frac{(P - P_1)(P - P_2)}{(P_0 - P_1)(P_0 - P_2)}, \quad \ell_1(P) = \frac{(P - P_0)(P - P_2)}{(P_1 - P_0)(P_1 - P_2)},$$
$$\ell_2(P) = \frac{(P - P_0)(P - P_1)}{(P_2 - P_0)(P_2 - P_1)}.$$

Each $\ell_j(P)$ satisfies $\ell_j(P_i) = \delta_{ij}$ (i.e., it equals 1 at P_j and 0 otherwise). Then, the interpolated implied volatility is

$$\hat{\sigma} = \sigma_0 \cdot \ell_0(\hat{P}) + \sigma_1 \cdot \ell_1(\hat{P}) + \sigma_2 \cdot \ell_2(\hat{P}). \tag{5.5}$$

This gives a smooth quadratic approximation that exactly passes through the known volatility values at the known prices.

The PDF is obtained using Breeden and Litzenberger's result as follows: Given a vector of strike prices $\{K_i\}$, predicted European put option prices, $\hat{P}_i$, and constant spacing $h = K_{i+1} - K_i$, the discrete risk-neutral probability density function (PDF) at each strike is approximated using the five-point stencil as

$$f(K_i) = e^{rt} \cdot \frac{-\hat{P}_{i+2} + 16\hat{P}_{i+1} - 30\hat{P}_i + 16\hat{P}_{i-1} - \hat{P}_{i-2}}{12h^2}. \tag{5.6}$$

To convert this into a discrete probability distribution over strikes, we normalize the computed values to obtained the PDF at each point, i.e.,

$$\text{pdf\_probs}[i] = \frac{f(K_i)}{\sum_j f(K_j)}. \tag{5.7}$$

Overall, we use the Breden and Litzenberger Theorem to estimate the value of PDF for each strike. The Simpson's role is then used to numerically evaluate the area under the curve. For the PDD and the PINN approach applied to the BSMPDE, we indeed got an area of one by removing points of market stress. However, for the HSVM, where all the points were considered, we got an area approximately equal to 1. This is due to the fact that one of the data points at strike 225 was under market tension with a very small market volume. This, however, is normal in most market scenarios except esent the set of trainable parameters. The Tables 6, 7 and Fig. 10. In order to asses the accuracy of PDD approach in predicting European put options, consider Example 5.1 below which is based on a 30 days European put on an AAPL stock as priced on the 29/04/2025.

Example 5.1. Consider $N_{test} = \{50, 100, 200, 400, 800\}$, $N_{train} = 8N, N \in N_{test}$, $S_{min} = 150$, $S_{max} = 250$, $S_0 = 211.21$ $t_{min} = 0$, $t_{max} = 0.0822$, $epochs = 300$, $r = 0.039$, $k \in [100, 300]$, $\sigma \in [0.2, 0.5]$, $l_r = 0.001$, $h_l = \{4\}$, $n = \{128, 84, 31, 1\}$.

The additional test data employed beyond the primary test set to further assess the performance of our model in comparison with market data and the Black–Scholes model is presented in Table 1. In this table, OI denotes the open interest, IV represents the implied volatility, Vol refers to the trading volume, and %chg indicates the percentage change.

Using Example 5.1 and Table 1, the scattered plot of the Last price (proxy to European put prices) and European put prices approximated using the PDD is shown in Fig. 3(a) while Fig. 3(b) shows the scattered plot of the observed Last price against the Put values from the classical BSM. At each point in time, we expect the payoff from the PDD and that from the classical Black–Scholes model to be the same as the observed Last price. Hence a plot of the payoffs from the PDD against the payoffs from the market prices is expected to be a straight line through the origin at 45° to the positive horizontal axis. This line signifies the degree of good fit, i.e., for some solution set Sol^{MP} representing observed market price and Sol^{PDD} representing the payoffs obtained using PDD approach, the plot of ordered pairs

$$Sol^{MP} \times Sol^{PDD} = \{(U^{MP}, U^{PDD}) : U^{MP} \in Sol^{MP} \text{ and } U^{PDD} \in Sol^{PDD}\},$$

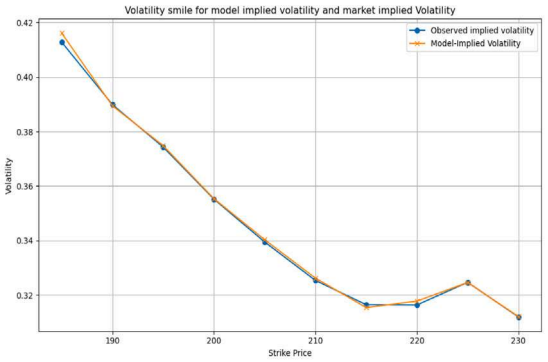
should be close to 45° line as much as possible if the payoff from PDD model is indeed a true proxy to the option value based on the data provided.

Consider 4, we observe the presence of a Kink in the smile at a strike of 225, how ever this is not abnormal. Firstly, lower trading volume and open interest at $K = 225$, can make the implied volatility calculation more susceptible to noise or outlier trades. Secondly, this may be due to a combination of market microstructure effects such as lower liquidity and higher bid–ask spreads, supply/demand imbalances at that strike, and possibly concentrated hedging activity. Such kinks are not uncommon in real-world options markets, especially at strikes with lower liquidity or special significance to market participants.

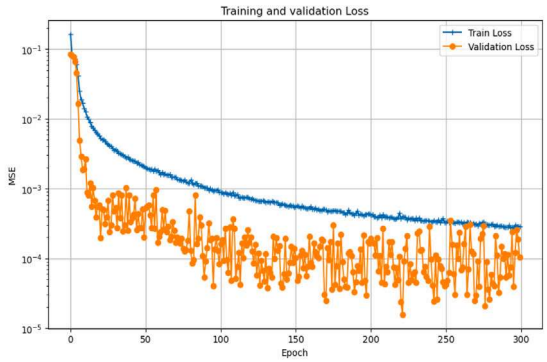
If this kink persists over time or is associated with significant trading activity, it may warrant further investigation into order flow or market

Table 1
Data for put option contracts on AAPL expiring extracted from Yahoo finance.

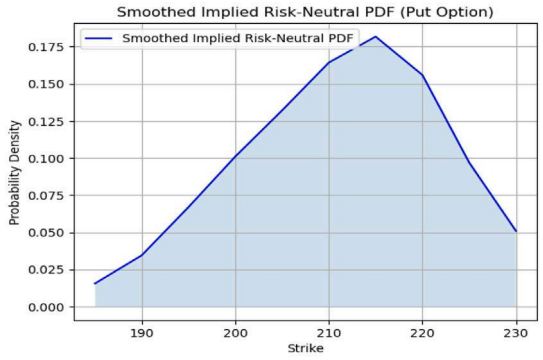
Contract	Date	Strike	Last	Bid	Ask	Change	% Chg	Vol	OI	IV
AAPL250530P00185000	4/29 3:43 PM	185	1.51	1.51	1.71	−0.34	−18.38%	146	537	41.28%
AAPL250530P00190000	4/29 3:54 PM	190	2.07	2.08	2.25	−0.26	−11.16%	362	788	38.99%
AAPL250530P00195000	4/29 3:59 PM	195	2.88	2.89	3.10	−0.44	−13.25%	206	474	37.43%
AAPL250530P00200000	4/29 3:50 PM	200	3.94	4.00	4.15	−0.51	−11.46%	503	1343	35.52%
AAPL250530P00205000	4/29 3:56 PM	205	5.36	5.45	5.60	−0.69	−11.40%	248	462	33.95%
AAPL250530P00210000	4/29 3:54 PM	210	7.21	7.20	7.50	−0.94	−11.53%	135	251	32.54%
AAPL250530P00215000	4/29 2:46 PM	215	9.85	9.65	10.00	−0.15	−1.50%	81	83	31.65%
AAPL250530P00220000	4/29 1:39 PM	220	13.16	11.85	13.20	−1.97	−13.02%	12	58	31.64%
AAPL250530P00225000	4/29 3:07 PM	225	16.05	15.65	17.00	−1.28	−7.39%	62	54	32.47%
AAPL250530P00230000	4/29 2:14 PM	230	19.95	18.90	20.75	−2.01	−9.15%	20	71	31.20%



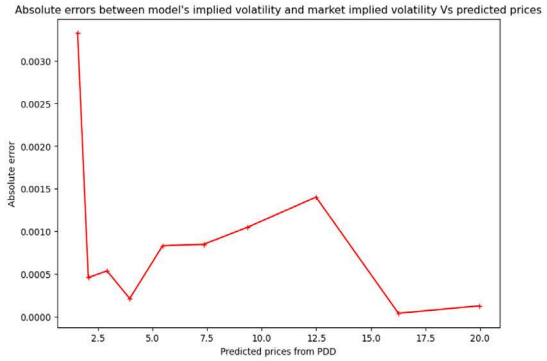
(a) Observed and model's implied volatility Vs strkes



(b) Training and Test loss Vs Epochs



(c) Risk-neutral PDF



(d) Absolute error based on implied volatility

Fig. 4. Plots from PDD approach: (a) Observed and model's implied volatility against the strikes. (b) Training and test loss vs. epochs (c) Risk-neutral PDF. (d) Absolute errors based on implied volatility. The results are obtained based on Example 5.1 calibrated on AAPL expiring on the as priced on the .

Table 2
The absolute errors between the observed implied volatilities and the implied volatilities estimated from the PDD model based on Table 1.

Market_IV	Predicted price	Market price	Model_IV	AE
0.4128	1.564741	1.51	0.416127	3.3269e−03
0.3899	2.056234	2.07	0.389439	4.6080e−04
0.3743	2.914451	2.88	0.374838	5.3784e−04
0.3552	3.953433	3.94	0.355415	2.1493e−04
0.3395	5.460224	5.36	0.340333	8.3345e−04
0.3254	7.345550	7.21	0.326250	8.5013e−04
0.3165	9.356567	9.85	0.315449	1.0506e−03
0.3164	12.487585	13.16	0.317803	1.4029e−03
0.3247	16.257862	16.05	0.324657	4.3071e−05
0.3120	19.968927	19.95	0.312129	1.2949e−04

events specific to that strike. Otherwise, it can often be attributed to normal market frictions and microstructure noise.

From Table 2, the PDD model exhibits strong alignment with market data, showing absolute errors (AE) between 4.3071×10^{-5} and 3.3269×10^{-3}. The largest deviation occurs at Market_IV = 0.4128 (AE = 3.3269×10^{-3}), revealing higher sensitivity to price discrepancies in high volatility regimes, while the smallest error (4.3071×10^{-5}) at $Market\_IV = 0.3247$, demonstrates optimal performance in mid-range volatilities. Non-linear responses are evident, as a 0.4934 price difference (Row 7) produces a smaller AE than a 0.0542 difference (Row 1).

With point accuracy ($AE < 0.0033$), the model proves suitable for pricing and hedging applications, particularly in moderate volatility ranges (0.3120−0.3395). While robust under typical market conditions, calibration refinements are recommended for $Market\_IV > 0.4$, where errors triple compared to mid-volatility levels. This performance profile supports risk management use cases while highlighting edge-case sensitivities requiring further investigation.

Table 3
Put options prices and absolute errors on AAPL expiring based on the option prices.

Strike	Last price	Bid	Ask	IV	BSM payoff	PDD payoff	MAE PDD	MAE BSM
185	1.51	1.51	1.71	41.28%	1.45735479	1.56474078	0.0547	0.0526
190	2.07	2.08	2.25	38.99%	1.94784697	2.05623388	0.0138	0.1222
195	2.88	2.89	3.10	37.43%	2.73081427	2.91445065	0.0345	0.1492
200	3.94	4.00	4.15	35.52%	3.71172095	3.95343256	0.0134	0.2283
205	5.36	5.45	5.60	33.95%	5.08758281	5.46022367	0.1002	0.2724
210	7.21	7.20	7.50	32.54%	6.91224133	7.34555006	0.1356	0.2978
215	9.85	9.65	10.00	31.65%	9.34235300	9.35656738	0.4934	0.5076
220	13.16	11.85	13.20	31.64%	12.47950254	12.48758507	0.6724	0.6805
225	16.05	15.65	17.00	32.47%	16.23267833	16.25786209	0.2079	0.1827
230	19.95	18.90	20.75	31.20%	19.95802422	19.96892738	0.0189	0.0080

Table 4
The accuracy of the PDD approach in %, $tol = \{0.001, 0.01, 01\}$, $N_{test} = \{200, 400, 800, 1600, 2000\}$.

tol	200	400	800	1600	2000
0.001	52.1394	70.2601	77.8185	93.0027.0039	76.2296
0.01	88.9930	97.5891	96.1503	98.5625	90.1519
0.1	100	100	100	100	100

From Table 3 we observed that based on the Absolute Error metric between observed prices and predicted prices, the PDD slightly outperforms the BSM. This shows that the PDD may be more accurate in option pricing than the BSM as it learns the dynamics of market factors from market data.

The process of parameter tuning is very crucial in obtaining an optimal network structure. After trying so may possibilities for the dept and breadth of the neural network under the PDD approach, we settled for an optimal structure is $h_l = 4$ and n = { 128, 64, 31, 1}, in that order. From Tables 2 and 3, we conclude that the PDD approach generalizes well to unseen data when trained on a dataset of size $N = 10000$ with a train-test split ratio of 8 : 2 and the provided data. We obtained this numerical results due to space constrain.

We examined a range of values for r, σ, and K to avoid the necessity of retraining the model whenever new parameter sets are introduced prior to making predictions.

To further evaluate the accuracy of the model against the BSM, we use the model to generate predictions on the test dataset X_{test}, denoting the predicted values as $\hat{U}$ and the solution from the BSM by U, we iterate through the test set, where N represents the total number of data points. For each point, we calculate the difference $|U - \hat{U}|$. Whenever this difference is less than a predefined tolerance value tol, we increment a counter. Finally, the percentage accuracy of the model is computed as

$$\text{Accuracy } (\%) = \frac{\delta}{N} \times 100\%, \tag{5.8}$$

where δ is the total number of points satisfying

$$\left|U_i - \hat{U}_i\right| < tol, \text{ and } i = 1, 2, \dots N.$$

We present the accuracy of the PDD model for $tol = \{0.0001, 0.001, 0.01, 0.1\}$, i.e., we evaluate

$$\text{Accuracy} = \frac{\sum_{j=1}^{N_{test}} \left|U - \hat{U}\right| < tol}{N} \times 100\%, \tag{5.9}$$

for each value of the tolerance (tol). Based on Table 4 we again conclude that the PDD model is able to capture option prices within a reasonable tolerance.

Table 3 demonstrates the performance of the model in comparison to the true values of Apple stock options observed in real time. It can be noted from this table that as the strike price increases, the option's value generally rises. This is because higher strike prices increases the likelihood that the options will become profitable (i.e., the stock price must decrease more to reach the in-the-money condition). We further observe from Table 3 that the option prices from the PDD model are more consistent with the Last prices of the observed option prices than the prices from the BSM. This suggest that the PDD is slightly a better performing model than the BSM. This observation is further validated in Figs. 3 and 4.

We observe from the data in Table 1 and the results in Table 3, that using implied volatility rather than historical volatility yields more accurate results and aligns with prevailing practice in both academic research and industry as found in [26] and related works.

5.2. Numerical experiments, simulations and discussions for PINN applied to BSMPDE

As defined under Section 5.1, we assume that at time t_j, the stock price dynamics are given by

$$dS(t_j) = \mu S(t_j)dt_j + \sigma S(t_j)dW(t_j), \ j = 1, 2, \dots, N, \tag{5.10}$$

where μ is the drift and σ is the volatility of the asset price. Using the Itô lemma and the Taylor series, we get

$$S_j = S_0 \exp\left(\sigma w_{t_j} + \left(\mu - \frac{\sigma^2}{2}\right)t\right), \ S(t_0) = S_0. \tag{5.11}$$

In this section, instead of estimating bounds for the stock price based on the spot price and discretizing over a limited sample, we adopt an alternative methodology by utilizing historical stock prices of the AAPL index spanning the past 3000 calendar days as the foundational dataset. This approach is motivated by the objective of investigating whether the incorporation of direct historical price data can enhance the predictive accuracy of the model. To this end, the length of the historical window is treated as a variable parameter in the experiments, enabling an assessment of its influence on model performance. This framework also facilitates an examination of potential non-Markovian dependencies between past price trajectories and future price movements.

The graph of the training data distribution is presented in Fig. 5.

In Fig. 5, we observe that, the stock price never go below zero which can be justified from the stock price dynamics formula above. Hence, we ensure that S_0 is close to the minimum possible stock price in the market within the desired time frame but not zero as in reality the price of a stock can hardly ever be zero.

To access the effectiveness of PINN in predicting option prices considering the classical Black–Scholes PDE, consider Example 5.2 below.

Example 5.2. Consider the PINN approach applied to the standard BSMPDE with the following parameters: $l_r = 1(10^{-02}, 10^{-03}, 10^{-04})$, which decay in that order after every 1000 epochs, Normalized prices (S), $t \in [0, 0.0822]$, $r = 0.0397$, $K = 230$, $\sigma = [0.2, 0.5]$, $e = 5000$, $h_l = 4$, $n = \{128, 64, 32, 1\}$, $N = 10000$.

Fig. 6 shows the European put option payoff curves at maturity for various test set sizes as compared to the payoff from the standard BSM model.

From Fig. 6, we observe that the predicted payoff curves closely fit the payoff curves generated from the BSM model with some slight deviations. This is expected as our model will just converge to the BSM

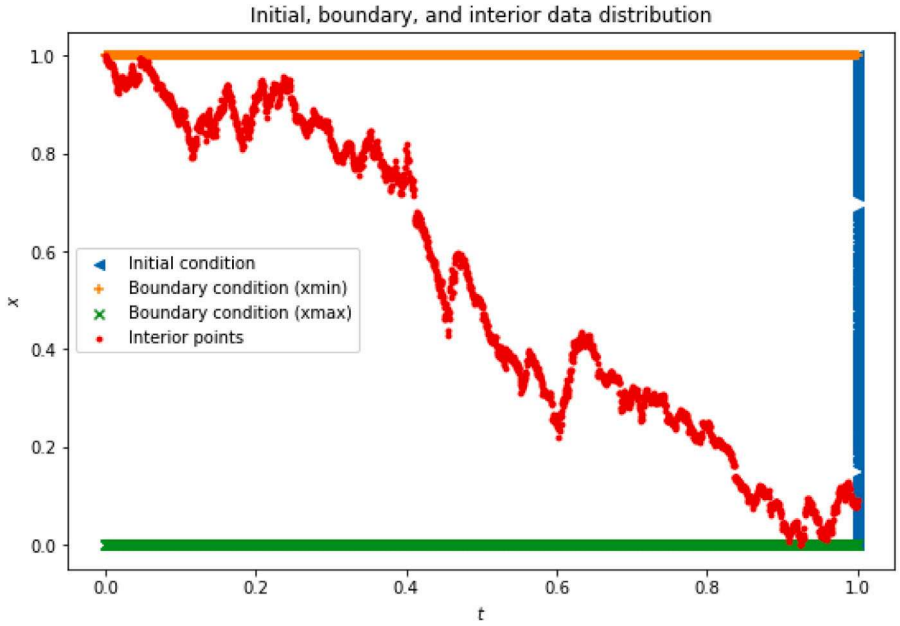
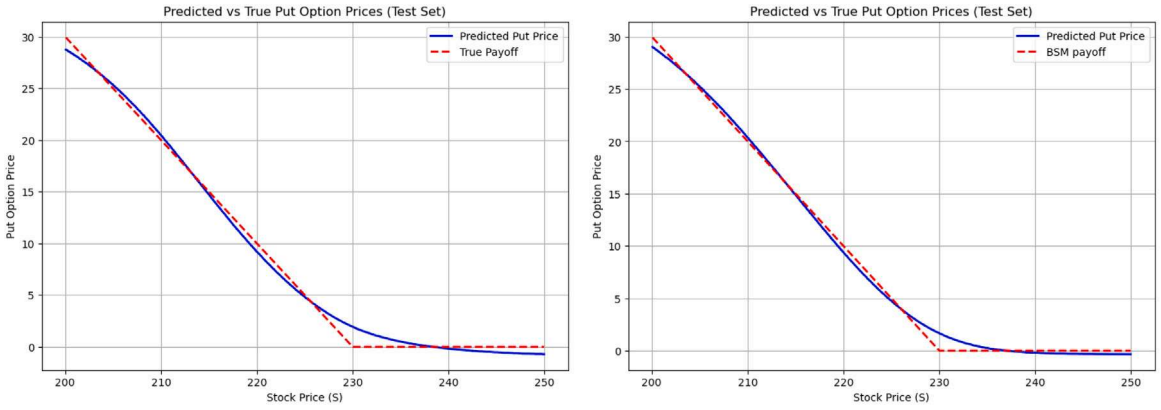
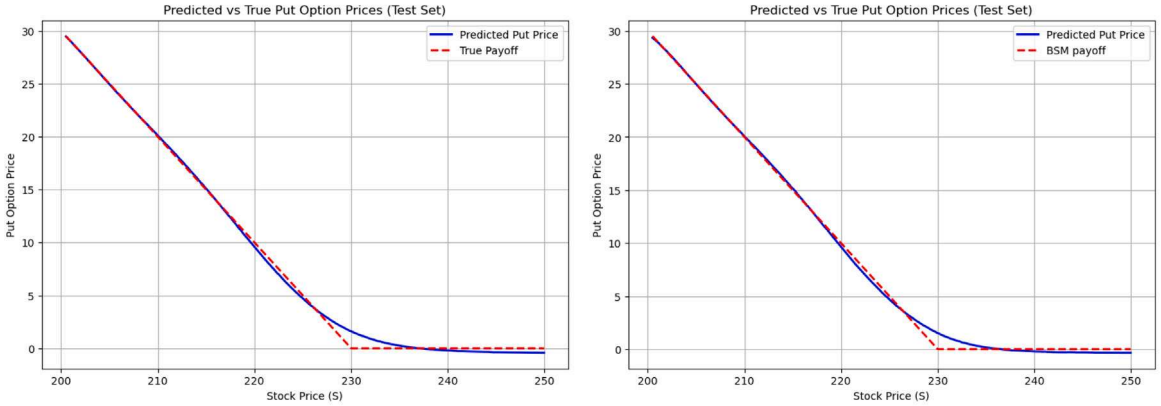


Fig. 5. Normalized data distribution for training PINN obtained from Yahoo Finance on AAPL stock from to.



(a) $N_{test} = 100$

(b) $N_{test} = 200$



(c) $N_{test} = 400$

(d) $N_{test} = 800$

Fig. 6. PINN option prices based on the BSMPDE compared to true prices (in this case we mean prices from BSM) for, $N_test = \{100, 200, 400, 800\}$.

Table 5
The absolute error between payoff from PINN model on BSMPDE and Payoff from BSM at different prices and maturity, $\sigma = 0.3120$, $r = 0.039$, $K = 230$.

S	t	Predicted payoff	Exact payoff	Absolute error
210.53	0.02226	19.4656	19.4694	3.8e−3
211.59	0.06678	18.4065	18.4082	1.7e−3
211.22	0.05137	18.7776	18.7755	2.1e−3
211.84	0.07705	18.1611	18.1633	2.2e−3
210.69	0.02911	19.3015	19.3061	4.6e−3
211.96	0.08219	18.0413	18.0408	5.0e−4
211.06	0.04452	18.9395	18.9388	7.0e−4
211.02	0.04281	18.9798	18.9796	2.0e−4
211.31	0.05479	18.6966	18.6939	2.7e−3
210.78	0.03253	19.2208	19.2245	3.7e−3

should there be a perfect fit thereby eliminating the struggle to propose improved models capable of better capturing option prices.

To access the performance of PINN model applied to the BSMPDE, consider Table 5. In this table, we evaluate the Maximum Absolute Errors between the price predicted with PINN and that predicted with the BSM and we print out the MAE, and the corresponding price and time for which this MAE occurs. The MAE is of order 10^{-1} relative to the BSM prices. This is quite reasonable based on existing literature.

From Tables 3 and 5, it is evident that PINN approach outperforms PDD method when evaluated using the absolute error metric. This superior performance indicates that the PINN framework more effectively captures the underlying market dynamics compared to the PDD approach. A likely explanation for this enhanced accuracy is the PINN's capability to integrate both empirical market data and the fundamental financial principles embedded within the BSMPDE.

Practically, this suggests that PINN-based models hold significant promise for improving option pricing and risk management tasks, particularly in environments where market conditions exhibit complex behaviors that challenge classical numerical methods or limited data is available. Furthermore, the ability of PINNs to incorporate domain knowledge while remaining data-driven may reduce the need for frequent recalibration, thereby enhancing model robustness and computational efficiency. We extend this framework to incorporate additional market factors, such as stochastic volatility via the HSVM to further validate and improve predictive performance and practical applicability (see Fig. 7).

From Table 6, it is evident that, based on the MAE on the IV defined as the absolute difference between the observed IV and the IV derived from the PINN and BSM prices, the IV predicted from PINN prices demonstrates superior accuracy compared to those obtained from the PDD and BSM models, as measured by its closer alignment with the market's observed IV.

5.3. Numerical results and discussions for the PINN applied to the HSVM

To asses the effectiveness of the HSVM in pricing European put options under the PINN approach, we consider Example 5.3 below.

Example 5.3. Let V_t and S_t be the value of the European Options and stock price respectively. We generate values of V_t and S_t as outlined in Section 4. Consider the parameters: $l_r = 1$(e-02, e-03, e-04), with decay happening after every 1000 epochs, $S \in [150, 250]$, $t \in [0, T]$, $v \in [0.1, 0.5]$, $k \in [185, 230]$, $\sigma = [0.1, 0.45]$, $e = 3000$, $N_{train} = 10000$, $h_l = 4$, $n = \{128, 64, 32, 1\}$, $N_b = N_0 = 500$, $\kappa = 1.5$, $\theta = 0.1$, $v_0 = 0.25$, $\rho = -0.7$.

In the Heston model, the variance process v_t follows the CIR (Cox–Ingersoll–Ross) process

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t}dW_t^v.$$

To ensure that the variance process v_t stays strictly positive (i.e., no absorption at zero), the Feller condition must be satisfied, i.e.,

$$2\kappa\theta > \sigma^2.$$

Let us check this for the full range of $\sigma \in [0.1, 0.45]$, with $\kappa = 3$ and $\theta = 0.2$. We have

$$2\kappa\theta = 2 \cdot 3 \cdot 0.2 = 1.2,$$

$\min \sigma^2 = 0.1^2 = 0.01$, and $\max \sigma^2 = 0.45^2 = 0.2025$. Thereforee, the Feller's condition is satisfied across the entire range of σ because for

$$\sigma^2 \in [0.01, 0.2025] < 2\kappa\theta = 1.2.$$

A plot of the corrected random paths is shown in Fig. 8.

This path is computationally achieved using the following formula in python

$$B = np.random.multivariate\_normal(\mu, \Sigma, size = N),$$

$$plt.plot(B.cumsum(axis = 0)).$$

The HSVM is define on the assumption that $W_1(t)$ and $W_2(t)$ are correlated.

Using Example 5.3, we generate in Fig. 9, the graph of the training loss against the epochs, the tail density of S_t under the HSVM and the geometric Brownian motion (GBM), the plot of the predicted option prices under the HSVM and the plot of the collocation points under the Heston model.

Fig. 9(c) illustrates the terminal price density distributions generated by the Heston stochastic volatility model with varying correlation parameter ρ values compared to the standard Geometric Brownian Motion (GBM). The results reveal several significant implications for European put option pricing.

The negative correlation case ($\rho = -0.7$) exhibits a more pronounced left tail and higher central peak compared to both GBM and positive correlation scenarios, consistent with the well-documented leverage effect where price declines correspond with volatility increases. This asymmetry results in higher theoretical prices for out-of-the-money put options, especially for strikes significantly below the current price level.

Conversely, the positive correlation scenario ($\rho = 0.7$) shows a slightly enhanced right tail density relative to GBM while maintaining comparable left tail characteristics. This pattern implies that positive price volatility correlation moderates downside risk pricing but potentially increases the cost of far out-of-the-money call options.

The differences between these distributions highlight the limitations of the standard Black–Scholes framework, which relies on GBM's symmetric normal distribution assumption. The Heston model's ability to capture skewness and excess kurtosis through its correlation parameter provides a more realistic representation of market dynamics, particularly for pricing options in environments where volatility clustering and asymmetric responses to market movements are prevalent.

The smoothed implied risk-neutral PDF derived from market prices of put options reveals several important characteristics as can be found in Figs. 9 and 10.

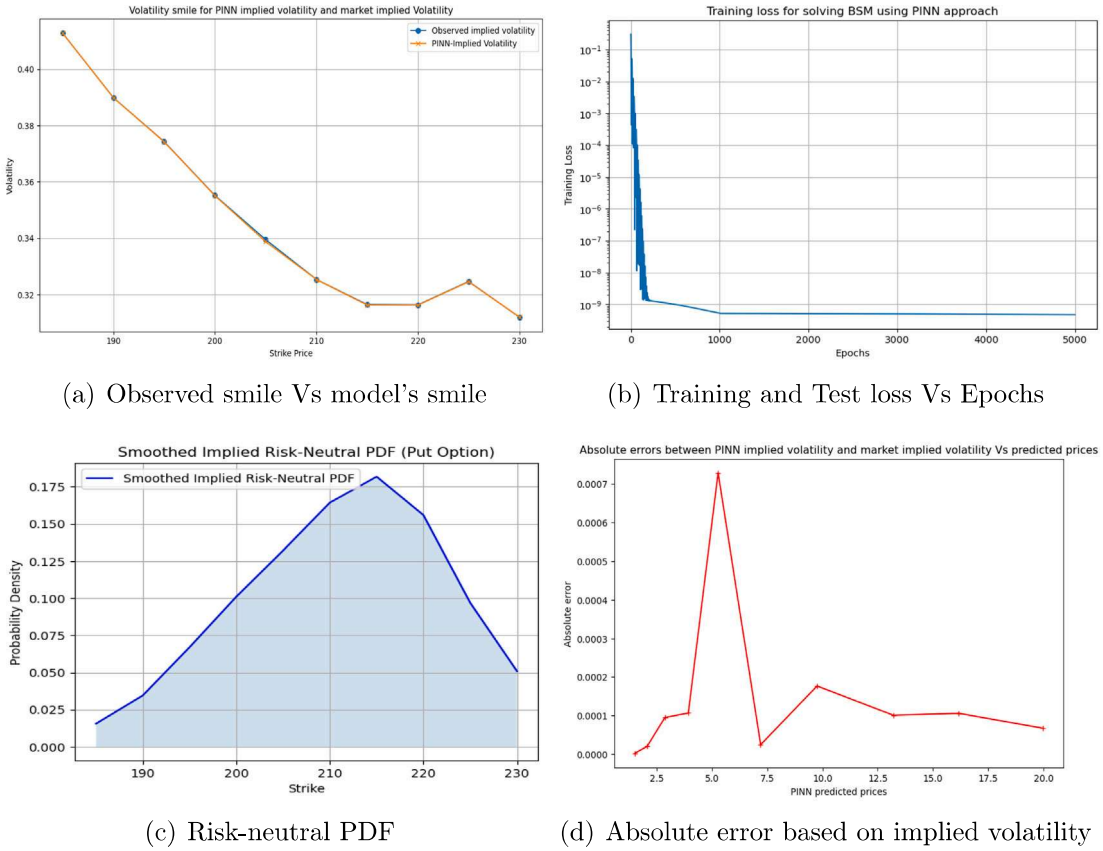
From Fig. 10, the distribution peaks around the 215 strike price, indicating the market's central expectation for the underlying asset. This non-symmetric bell shape differs significantly from the theoretical lognormal distribution assumed in traditional Black–Scholes pricing.

The left side of the distribution (lower strikes) shows a more gradual slope compared to the right side, suggesting market participants are pricing in higher probabilities of downside moves than would be expected under traditional models. This risk-neutral skewness reflects the market's collective risk aversion and willingness to pay premiums for downside protection.

The relatively smooth nature of the distribution indicates consistent pricing across the strike range, though the slightly higher density at the peak (around 0.175) compared to what might be expected

Table 6
Comparison of option prices and implied volatilities when BSMPDE is solved using PINN.

PINN price	BSM price	Market price	PINN IV	BSM IV	Market IV	PINN AE	BSM AE
1.510041	1.4574	1.51	0.412802	0.4173	0.4128	2.00e−05	4.50e−03
2.069339	1.9478	2.07	0.389880	0.3984	0.3899	2.00e−05	8.50e−03
2.874451	2.7308	2.88	0.374205	0.3829	0.3743	9.50e−05	8.60e−03
3.933233	3.7117	3.94	0.355094	0.3666	0.3552	1.06e−04	1.14e−02
5.281022	5.0876	5.36	0.338771	0.3516	0.3395	7.29e−04	1.21e−02
7.205550	6.9122	7.21	0.325375	0.3379	0.3254	2.50e−05	1.25e−02
9.756567	9.3424	9.85	0.316324	0.3376	0.3165	1.76e−04	2.11e−02
13.207585	12.4795	13.16	0.316299	0.3463	0.3164	1.01e−04	2.99e−02
16.157862	16.2327	16.05	0.324594	0.3156	0.3247	1.06e−04	9.10e−03
19.958927	19.9580	19.95	0.312067	0.3115	0.3120	6.70e−05	5.00e−04



(a) Observed smile Vs model's smile

(b) Training and Test loss Vs Epochs

(c) Risk-neutral PDF

(d) Absolute error based on implied volatility

Fig. 7. Plots based on solving BSMPDE using PINN: (a) Observed and model's implied volatility against the strikes. (b) Training vs. Epochs. (c) Risk-neutral PDF. (d) Absolute errors based on implied volatility. The results are obtained based on Example 5.1 calibrated on AAPL expiring on the as priced on the.

Table 7
Minimum density values and area under the density curve for different models.

Model	Minimum density value	Area under curve
Heston ($\rho = 0.7$)	1.19×10^{-5}	0.99981
Heston ($\rho = -0.7$)	6.55×10^{-10}	0.99962
GBM	3.99×10^{-9}	0.99969

from a normal distribution suggests some excess kurtosis in market expectations.

This empirically derived risk-neutral PDF aligns with the findings from the Heston model simulations, particularly with how negative correlation affects the distribution shape. The implied volatility smile embedded in this distribution validates the model's ability to capture market dynamics that standard GBM approaches cannot adequately represent.

The volatility smile plot demonstrates that the Heston model not only captures the empirical features of market implied volatility but also provides a realistic and robust framework for option pricing. The close match between model and market smiles, together with the insights from the implied risk-neutral PDF and tail density analysis, confirm the model's ability to reflect actual market dynamics and the importance of accounting for stochastic volatility and correlation effects in option pricing.

The empirical analysis demonstrates that the Adam optimizer achieves a convergence rate consistent with theoretical expectations for nonconvex stochastic optimization problems. The observed order of convergence $p \approx 0.82$ indicates efficient loss reduction, surpassing the baseline $O(1/\sqrt{k})$ rate but not fully reaching the ideal $O(1/k)$ rate.

5.4. General discussion, some challenges and proposed solutions for the PDD and PINN approaches

This work demonstrates that with clean and sufficient stock price data, the PDD approach is a viable tool for pricing European option contracts without requiring a governing PDE. Similarly, the PINN

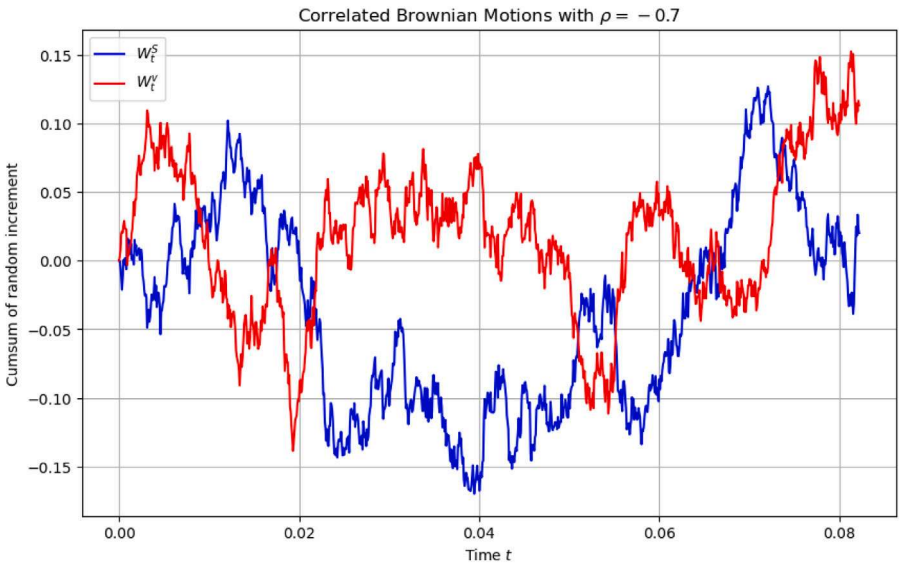


Fig. 8. Correlated random paths with $\rho = -0.7$.

Table 8
Comparison of option prices and implied volatilities with HSVM.

PINN price	BSM price	Market price	Heston price	PINN IV	BSM IV	Market IV	Heston IV	PINN AE	BSM AE	Heston AE
1.510041	1.4574	1.51	1.49	0.412802	0.4173	0.4128	0.4111	2.00e−05	4.50e−03	1.70e−03
2.069339	1.9478	2.07	2.02	0.389880	0.3984	0.3899	0.3882	2.00e−05	8.50e−03	1.70e−03
2.874451	2.7308	2.88	2.83	0.374205	0.3829	0.3743	0.3735	9.50e−05	8.60e−03	8.00e−04
3.933233	3.7117	3.94	3.91	0.355094	0.3666	0.3552	0.3541	1.06e−04	1.14e−02	1.10e−03
5.281022	5.0876	5.36	5.31	0.338771	0.3516	0.3395	0.3380	7.29e−04	1.21e−02	1.50e−03
7.205550	6.9122	7.21	7.14	0.325375	0.3379	0.3254	0.3240	2.50e−05	1.25e−02	1.40e−03
9.756567	9.3424	9.85	9.76	0.316324	0.3376	0.3165	0.3158	1.76e−04	2.11e−02	7.00e−04
13.207585	12.4795	13.16	13.12	0.316299	0.3463	0.3164	0.3159	1.01e−04	2.99e−02	5.00e−04
16.157862	16.2327	16.05	16.10	0.324594	0.3156	0.3247	0.3241	1.06e−04	9.10e−03	6.00e−04
19.958927	19.9580	19.95	19.96	0.312067	0.3115	0.3120	0.3119	6.70e−05	5.00e−04	1.00e−04

Table 9
Estimated order of convergence p_k at each epoch using $p_k = \frac{\log(C/L_k)}{\log k}$, with $C \approx 0.101$, obtained by performing the regression $\log L_k = a + b \log k$, where $b = -p_k$, and $a = \log C$ The loss (L_k) values are obtained from the training loss of PINN optimized using Adam optimization.

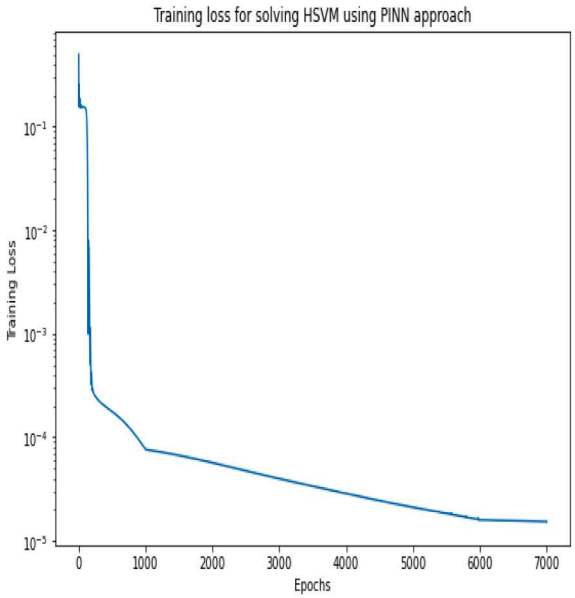
Epoch k	Loss L_k	$\log k$	$\log(C/L_k)$	Order p_k
200	1.3108e−3	5.30	4.34	0.82
400	5.6701e−4	5.99	5.18	0.86
800	3.0366e−4	6.68	5.81	0.87
1600	2.3274e−4	7.38	6.07	0.82
3200	1.3535e−4	8.07	6.62	0.82

framework effectively solves well defined PDEs in finance, such as the HSVM, by leveraging differentiable computations across input variables and trainable parameters. Key contributions include:

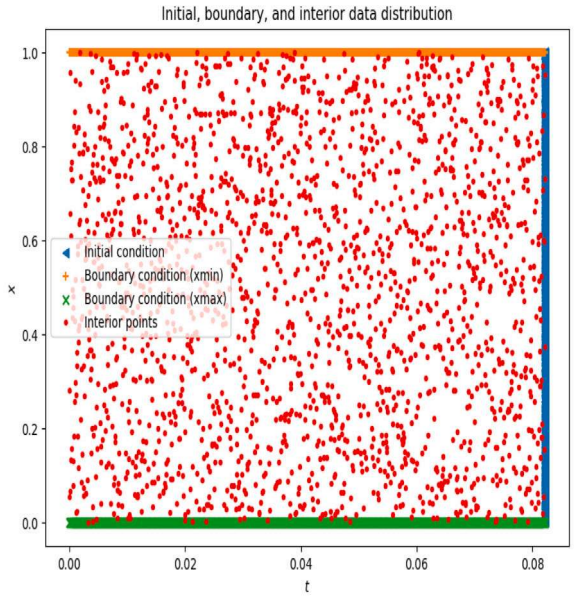
- The PINN approach offers an efficient alternative for pricing options governed by financial PDEs, outperforming traditional numerical methods in terms of simplicity, computational efficiency, and ease of implementation.
- We introduce new techniques of processing data for training PDD models and PINN models when applied to option pricing. Unlike traditional approaches, our approach describe unique feature and offer better adaptability to option pricing and calibration, which leads to improved accuracy, efficiency, scalability.
- Our work provides an extensive empirical evaluation on options data, demonstrating that our approach consistently outperforms existing models like the BSM in pricing options based on the absolute value metric. Furthermore, both the PDD and the PINN approaches are able to correctly capture the volatility smiles in options data with PINN showing slight superiority over the PDD, something that the BSM struggles to capture.
- We offer new theoretical insights into the convergence of PINN, including proofs and analyses, that deepen the understanding of the convergence of PINN and lay the groundwork for future research.
- The proposed approaches (PDD and PINN) are readily applicable to real-world option pricing, as evidenced by practical calibration performed in this examples above highlighting its potential impact financial engineering.

While the PINN methodology delivers reliable results, challenges remain in optimizing its performance. Below, we outline key challenges and solutions:

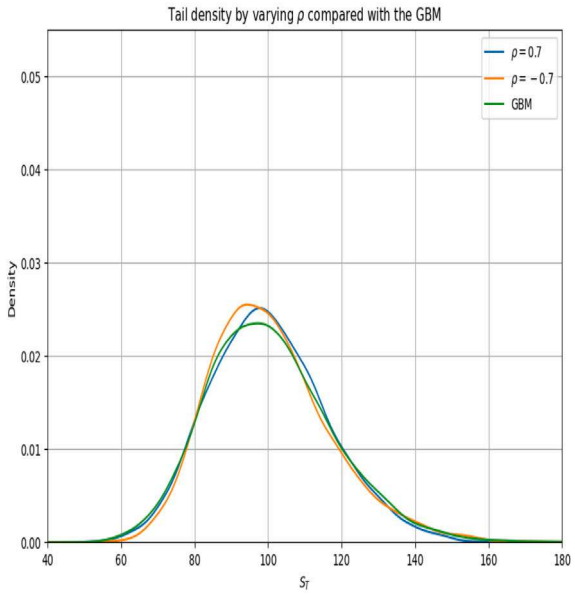
- The PINN requires careful hypeparameter turning to obtained accurate predictions. The learning rate was dynamically adjusted starting from 0.01 and spanning over an appropriate epoch configuration. Hyperparameters, including the number of neurons and hidden layers, were fine-tuned to optimize the network's configuration.
- To address sensitivity to local minima, we used an optimization algorithm combining the strengths of Adaptive Gradient Algorithm and Root Mean Square Propagation. By carefully incorporating initial and boundary conditions, instability in solutions was minimized.
- Feature standardization and normalization were applied during training, with outputs re-transformed to their original scale. Although these transformation and re-transformation is unavoidable which accurate learning is pivotal, it causes some loss of accuracy, Refer to Section 4 and [22] for details.



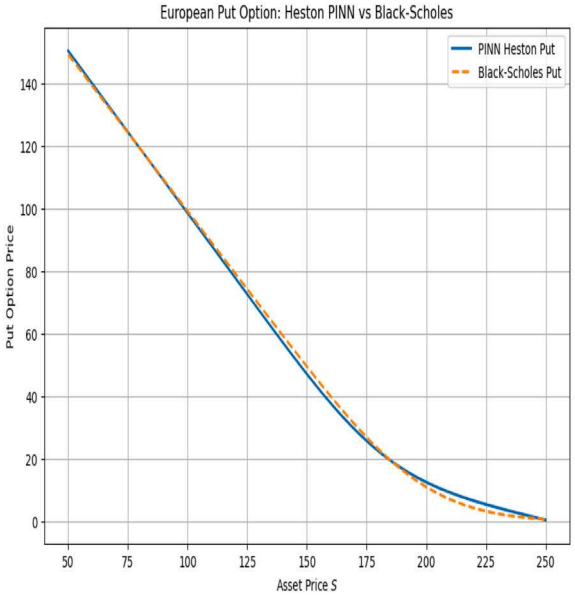
(a) Training loss Vs epoches



(b) Scattered plot of the collocations points under the HSVM



(c) tail Density under HSVM and GBM



(d) 3D plot of the predicted prices

Fig. 9. Plots based on solving Heston stochastic volatility PDE using PINN: (a) Training loss vs. epochs (b) Training data distribution (c) Comparative tail density distribution of the HSVM and GBM, (d) The predicted payoff vs. payoff from BSM, based on Example 5.3.

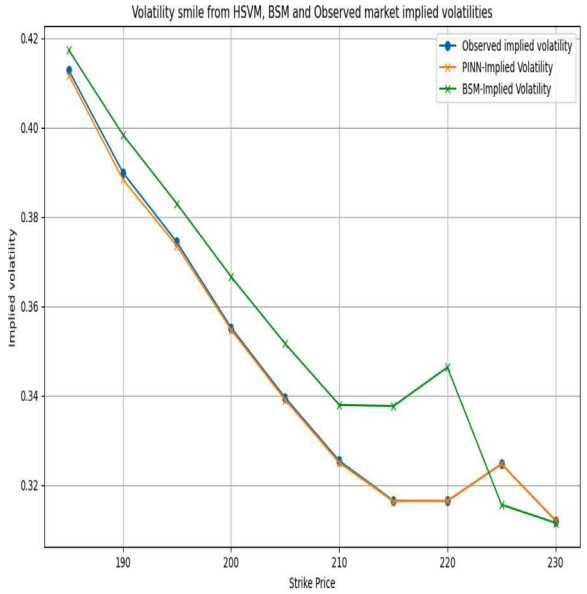
- Network depth and width were varied to identify optimal configurations, ensuring stability and accuracy in solutions.

Through examples and results (Tables 1–9 and corresponding figures), we validated the effectiveness of PDD and PINN approaches for financial option pricing, offering practical insights and advancements for computational finance.

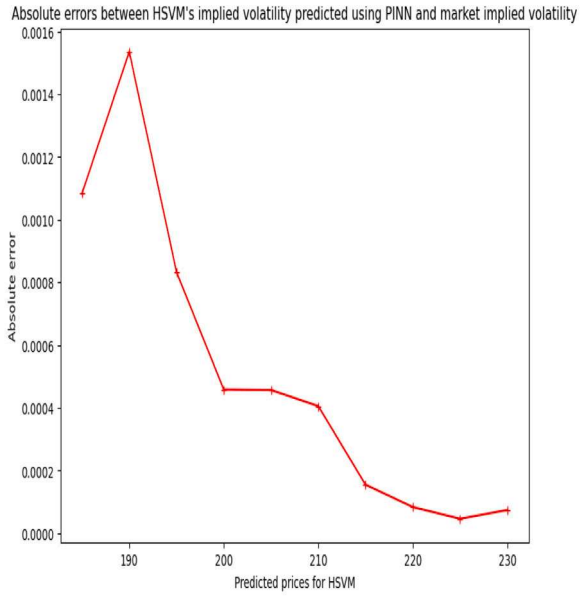
6. Conclusions

In conclusion, this paper presents a robust analysis of Physics-Informed Neural Networks (PINN) and Purely Data-Driven (PDD) approaches for pricing European options. Through comprehensive experiments, we demonstrated that the PINN method can effectively solve the HSVM, provided that the neural network is structured to allow differentiable computations with respect to its input variables and trainable parameters. Our results shows that PINNs offer significant advantages over traditional discretization-based numerical methods by being less complex, computationally efficient, and easier to implement. The PDD approach, while trained on theoretical data, exhibited a high accuracy of up to 95% in predicting real-time option prices, as shown in Table 4.

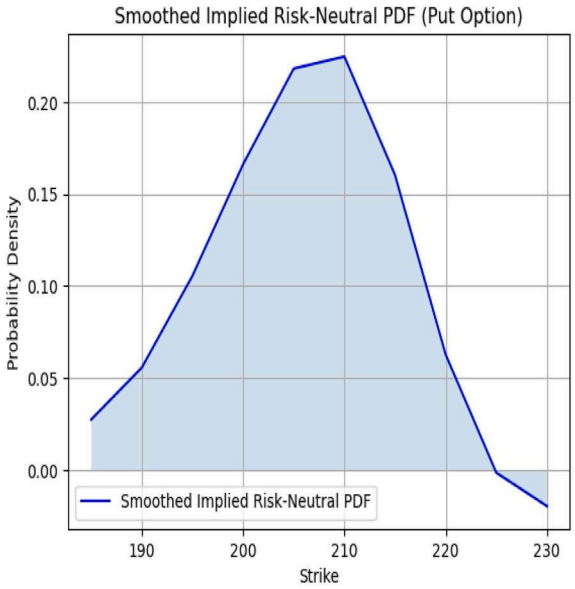
We further demonstrated how the training data can be generated using real market parameters, establishing that both PINN and PDD methods are effective tools for pricing financial options with PINN solutions fitting more closely to the solution from the classical BSM. Specifically, integrating the HSVM into the PINN framework



(a) Volatility smile



(b) Absolute error



(c) PDF from HSVM

Fig. 10. (a) Volatility smile compared for the observed implied volatility, the implied volatility from BSM and the implied volatility for the HSVM, (b) The Absolute Error between the implied volatility for the HSVM and that observed market implied volatility. (c) The PDF.

and training it to value options represents a valuable contribution to computational finance. Moreover, we provided a convergence analysis confirming that the PINN approach, when trained with the sigmoid activation function and the Adam optimizer, guarantees the existence of a local minimum and convergence to it under some specific conditions.

Despite the promising results, challenges such as stiffness, the existence of multiple local minima, scaling issues, and sensitivity to hyperparameters were identified and addressed through various optimization techniques and careful network configuration. Future research will focus on extending these methods to more complex options, aiming to refine and optimize the deep learning models for broader applications in financial option pricing.

CRediT authorship contribution statement

Samuel M. Nuugulu: Validation, Investigation, Data curation, Visualization, Project administration, Conceptualization, Writing – review & editing, Supervision, Formal analysis. **Kailash C. Patidar:** Writing – review & editing, Project administration, Funding acquisition, Data curation, Validation, Methodology, Conceptualization, Supervision, Investigation, Formal analysis. **Divine T. Tarla:** Investigation, Visualization, Methodology, Formal analysis, Writing – original draft, Software, Data curation.

Fundings

Research of KCP was supported by the South African National Research Foundation, South Africa.

Declaration of competing interest

No conflict of interests to declare.

Appendix

This appendix gives some mathematical preliminaries, the layout steps for implementing PINN and the proves of propositions and theorems presented in this paper.

A.1. Mathematical preliminaries

Definition A.1. A set S is convex if the straight line segment connecting any point points in S is entirely in S, i.e.,

$\forall x, y \in S$ and $\lambda \in [0,1],\ \lambda x + (1-\lambda)y \in S.$

A function f is convex if it domain S is convex and

$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$

If $f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y)$, we say that f is strictly convex whenever $x \neq y$ and $\lambda \in [0,1]$. A function f is said to be concave if $-f$ is convex [27].

Theorem A.2. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable in an open domain, then the following are equivalent:*

(i) f *is convex*
(ii) $\forall x, y \in dom(f), f(y) \geq f(x) + \nabla f(x) \cdot (y - x)$
(iii) $\nabla^2 f(x) \geq 0\ \forall x \in dom(f)$

Proof. The proof of this theorem is presented in [28].

Definition A.3. A functional is a map $F : V \to \mathbb{F}$ from a vector space V into a field of scalars $\mathbb{F}$ where $\mathbb{F}$ is typical a field of real or complex numbers [20].

Definition A.4. Let Ω be an open subset of $\mathbb{R}^n$, $\mathcal{L}^p(\Omega)$ the set of p-integrable functions on Ω, i.e.,

$$\mathcal{L}^p(\Omega) = \left\{ u : \Omega \to \mathbb{R} : \left(\int_\Omega (u(x))^p \, dx \right)^{\frac{1}{p}} < \infty \right\}, \tag{A.1}$$

and D^γ the derivative of order $|\gamma| \leq k$. The Sobolev space $W^{k,p}(\Omega)$ is defined as the set of functions that have weak derivatives up to order k and are $p-integrable$, i.e.,

$$W^{k,p}(\Omega) = \{ u \in \mathcal{L}(\Omega) : D^\gamma u \in \mathcal{L}(\Omega)\ \forall \gamma \text{ satisfying } |\gamma| \leq k \}. \text{ [20]} \tag{A.2}$$

We define the norm on $W^{k,p}(\Omega)$ as

$$\|u\|_{W^{k,p}(\Omega)} = \left(\sum_{|\gamma| \leq k} \|D^\gamma u\|^p_{\mathcal{L}(\Omega)} \right)^{\frac{1}{p}}. \tag{A.3}$$

Definition A.5. Let (S, Σ, μ) be a measurable space, let $p, q \in [1, \infty]$ with $\frac{1}{p} + \frac{1}{q} = 1$, then for all measurable real or complex valued functions on S, the Holder's inequality is given by

$\|f.g\|_1 \leq \|f\|_p \|g\|_q.$

Furthermore, if let $p, q \in (1, \infty)$, $f \in \mathcal{L}^p(\mu)$ and $g \in \mathcal{L}^q(\mu)$, the Holder's inequality becomes an equality iff $\|f\|^p$ and $\|g\|^q$ are linearly independent, i.e., there exist constants a, b not all zeros such that (see [29])

$$a\|f\|^p + b\|g\|^q = 0,\ \mu - \text{almost surely}. \tag{A.4}$$

Definition A.6. A Banch space is a normed vector space V over a field $\mathbb{F}$ such that every Cauchy $\{x_n\} \subset V$ converges to some $x \in V$. A sequence $\{x_n\}$ is Cauchy if $\forall \epsilon > 0, \exists N : \forall m, n \geq N, \|x_n - x_m\| \leq \epsilon$. This definition can be found in the numerous textbooks on real analysis, e.g. [30].

Definition A.7. The p-norm of a vector $\mathbf{X}$ in $\mathbb{R}^n$ is defined as $\|\mathbf{X}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$, where $\mathbf{X} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and $p \geq 1$. For specific values of p, this norm has special names: When $p = 1$, it is called the Manhattan norm or ℓ_1-norm given defined as $\|\mathbf{X}\|_1 = \sum_{i=1}^n |x_i|$. When $p = 2$, it is called the Euclidean norm or ℓ_2-norm defined as $\|\mathbf{X}\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}}$. When $p = \infty$, it is called the maximum norm or ℓ_∞ -norm, which is defined as $\|\mathbf{X}\|_\infty = \max_{1 \leq i \leq n} |x_i|$, for details, see e.g., [20,30] and the references therein.

Definition A.8. The closure of a set Ω in a topological space is the smallest closed set that contains Ω. It is denoted by $\overline{\Omega}$ or $\text{cl}(\Omega)$ and can be defined as

$$\overline{\Omega} = \{ x \in X \mid \text{every open set containing } x \text{ intersects } \Omega \}. \tag{A.5}$$

Alternatively, the closure of Ω can be defined as the union of Ω and its limit points

$$\overline{\Omega} = \{ \Omega \cup x \in X \mid \text{every open set containing } x \text{ contains a point of } \Omega \setminus x \}, \tag{A.6}$$

where X is the topological space containing Ω.

Definition A.9. The notation $u|_\Omega$ denotes the restriction of a function u to the set Ω. Specifically, if $u : X \to Y$ is a function and $\Omega \subseteq X$, then the restriction $u|_\Omega : \Omega \to Y$ is defined by:

$$(u|_\Omega)(x) = u(x), \quad \text{for all } x \in \Omega. \tag{A.7}$$

This means that $u|_\Omega$ is the function u considered only on the subset Ω of its original domain.

Definition A.10. The Lipschitz norm of a function $u : V \to W$ between two normed vector spaces $(V, \|\cdot\|_V)$ and $(W, \|\cdot\|_W)$ is defined as the smallest Lipschitz constant L for which the function f satisfies the Lipschitz condition. It is given by:

$$\|u\|_{\text{Lip}} = \sup_{\mathbf{x} \neq \mathbf{y}} \frac{\|u(\mathbf{x}) - u(\mathbf{y})\|_W}{\|\mathbf{x} - \mathbf{y}\|_V}. \tag{A.8}$$

If the function f is Lipschitz continuous, then this norm is finite.

A function u is Lipschitz if $\|u\|_{\text{Lip}} < \infty$. Furthermore, by the mean value theorem, $u \in C^1(V, W) \implies \|u\|_{\text{Lip}} < \|u\|_{C^1(V)}$. The Lipschitz norm $\|u\|_{\text{Lip}}$ measures the steepness or rate of change of the function f and is a key concept in the study of Lipschitz continuous functions.

Definition A.11. The boundary of a set Ω in a topological space X is the set of points that can be approached both from within Ω and from the outside of Ω. It is denoted by $\partial\Omega$ and can be defined as:

$$\partial\Omega = \overline{\Omega} \cap \overline{X \setminus \Omega}, \tag{A.9}$$

where $\overline{\Omega}$ is the closure of Ω and $\overline{X \setminus \Omega}$ is the closure of the complement of Ω.

In other words, the boundary $\partial\Omega$ consists of all points $x \in X$ such that every neighborhood of x contains at least one point in Ω and at least one point in $X \setminus \Omega$ in X.

Definition A.12. The gradient of a vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, where $\mathbf{f} = (f_1, f_2, \dots, f_m)$, is defined as:

$$\nabla \mathbf{f} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}. \tag{A.10}$$

The matrix in Eq. (A.10) is known as the Jacobian matrix of the vector-valued function $\mathbf{f}$ and is often denoted as $J_{\mathbf{f}}(X)$. The gradient points in the direction of steepest increase of $\mathbf{f}$ and its magnitude represent the rate of change of $\mathbf{f}$ in that direction. In optimization problems, the learning rate can be taken as the magnitude of the Jacobian matrix, i.e., $\|J_{\mathbf{f}}(X)\|$ if the Jacobian matrix is a square matrix. Furthermore, the Laplacian of $\mathbf{f}$ is given by

$$\nabla^2 \mathbf{f} = \begin{pmatrix} \frac{\partial^2 f_1}{\partial x_1^2} + \frac{\partial^2 f_1}{\partial x_2^2} + \dots + \frac{\partial^2 f_1}{\partial x_n^2} \\ \frac{\partial^2 f_2}{\partial x_1^2} + \frac{\partial^2 f_2}{\partial x_2^2} + \dots + \frac{\partial^2 f_2}{\partial x_n^2} \\ \vdots \\ \frac{\partial^2 f_m}{\partial x_1^2} + \frac{\partial^2 f_m}{\partial x_2^2} + \dots + \frac{\partial^2 f_m}{\partial x_n^2} \end{pmatrix}. \tag{A.11}$$

Definition A.13. The Hessian of a vector-valued function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, where $\mathbf{f} = (f_1, f_2, \dots, f_m)$, is a tensor of second-order partial derivatives. Each component function f_i has its own Hessian matrix. The Hessian of $\mathbf{f}$ is then defined as a matrix of these Hessian matrices. Let each H_{f_i} (for $i = 1, 2, \dots, m$) is the Hessian matrix of the scalar function f_i, which is given by:

$$H_{f_i} = \begin{pmatrix} \frac{\partial^2 f_i}{\partial x_1^2} & \frac{\partial^2 f_i}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f_i}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f_i}{\partial x_2 \partial x_1} & \frac{\partial^2 f_i}{\partial x_2^2} & \dots & \frac{\partial^2 f_i}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_i}{\partial x_n \partial x_1} & \frac{\partial^2 f_i}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f_i}{\partial x_n^2} \end{pmatrix}. \tag{A.12}$$

The Hessian describes the local curvature of $\mathbf{f}$. If $H_{\mathbf{f}}$ is positive definite, then $\mathbf{f}$ has a local minimum. If the Hessian is negative definite, $\mathbf{f}$ has a local maximum, otherwise $\mathbf{f}$ has saddle points.

A.2. Proof of Proposition 4.1

Let Ω represent the set of trainable parameters. The initial weights $W_1 \in \Omega$ is sampled from a uniform or random distribution as follows

$$W \sim U\left(-\sqrt{\frac{6}{n_{j-1}+n_j}}, \sqrt{\frac{6}{n_{j-1}+n_j}}\right), \text{ or } W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{j-1}+n_j}}\right). \tag{A.13}$$

This is called the Xavier Initialization. Other initialization in literature include the He Initialization which is suited for ReLU activation function given by $W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{j-1}}}\right)$, and the LeCun Initialization suited for the tanh activation and given by $W \sim \mathcal{N}\left(0, \sqrt{\frac{1}{n_{j-1}}}\right)$, where n_j is the number of neurons in the h_j hidden layer.

Using the Xavier initialization, we show that $\Omega^0 = \left[-\sqrt{\frac{6}{n_{j-1}+n_j}}, \sqrt{\frac{6}{n_{j-1}+n_j}}\right]$, is convex. As explained in Definition A.1, let $w_i, w_j \in \Omega^0$, and $\lambda \in [0,1]$, we show that $w_\lambda = \lambda w_i + (1-\lambda) w_j \in \Omega^0$.

$$w_i \geq -\sqrt{\frac{6}{n_{j-1}+n_j}} \implies \lambda w_i \geq -\lambda\sqrt{\frac{6}{n_{j-1}+n_j}}, \tag{A.14}$$

$$w_j \geq -\sqrt{\frac{6}{n_{j-1}+n_j}} \implies (1-\lambda) w_j \geq -(1-\lambda)\sqrt{\frac{6}{n_{j-1}+n_j}}. \tag{A.15}$$

Adding the two equations gives

$$\lambda w_i + (1-\lambda) w_j = w_\lambda \geq -\lambda\sqrt{\frac{6}{n_{j-1}+n_j}} - (1-\lambda)\sqrt{\frac{6}{n_{j-1}+n_j}} = -\sqrt{\frac{6}{n_{j-1}+n_j}}. \tag{A.16}$$

Similarly,

$$w_i \leq \sqrt{\frac{6}{n_{j-1}+n_j}} \implies \lambda w_i \leq \lambda\sqrt{\frac{6}{n_{j-1}+n_j}}, \tag{A.17}$$

$$w_j \leq \sqrt{\frac{6}{n_{j-1}+n_j}} \implies (1-\lambda) w_j \leq (1-\lambda)\sqrt{\frac{6}{n_{j-1}+n_j}}. \tag{A.18}$$

This implies that

$$\lambda w_i + (1-\lambda) w_j = w_\lambda \leq \lambda\sqrt{\frac{6}{n_{j-1}+n_j}} + (1-\lambda)\sqrt{\frac{6}{n_{j-1}+n_j}} = \sqrt{\frac{6}{n_{j-1}+n_j}}, \tag{A.19}$$

and

$$-\sqrt{\frac{6}{n_{j-1}+n_j}} \leq w_\lambda \leq \sqrt{\frac{6}{n_{j-1}+n_j}} \implies w_\lambda \in \left[-\sqrt{\frac{6}{n_{j-1}+n_j}}, \sqrt{\frac{6}{n_{j-1}+n_j}}\right]. \tag{A.20}$$

Therefore, $\Omega^0 \subset \Omega$ is convex.

Consider an architecture of 4 hidden layers, 10 neurons per hidden layer, 2 inputs variables and one output, then

$$\Omega = \mathbb{R}^{10\times 2} \times \mathbb{R}^{10\times 10} \times \mathbb{R}^{10\times 10} \times \mathbb{R}^{10\times 10} \times \mathbb{R}^{10\times 1} \equiv M_{10\times 30}. \tag{A.21}$$

The bias is usually initialized to zero. As the training progresses, the number of biases correspond to the number of neurons in each hidden layer. Hence ignoring the initial and the biases at the output neuron, then the bias matrix is an element of

$$\mathbb{R}^{10} \times \mathbb{R}^{10} \times \mathbb{R}^{10} \times \mathbb{R}^{10},$$

a total of 40 biases. Including the biases at the output neurons gives 41 parameters for the biases. Again, the domain for the biases is convex. $R^{n\times m}$ is convex, it is well known that the cross product of convex sets is convex, hence the domain of the trainable parameters Ω is convex. □

A.3. Proof of Proposition 4.2

For each hyperparameter $\theta_t \in \Omega$, compute the gradients $Loss_\theta$ of the objective function with respect to the parameter θ_t at time t as

$$Loss_{\theta_{t_j}} = \frac{\partial Loss_{t_j}}{\partial \theta_{t_j}}, \ j = 1, 2, \dots, N; N \in \{N_0, N_b, N_r\}. \tag{A.22}$$

Given that the biased first moment estimate is updated as $m_{t_j} = \varphi_{t_j} m_{t_{j-1}} + (1-\varphi_{t_j}) Loss_{\theta_{t_j}}$, and the biased second moment estimate as $v_{t_j} = \rho_{t_j} v_{t_{j-1}} + (1-\rho_{t_j})(Loss_{\theta_{t_j}})^2$, using Adam optimization, the parameters are updated as

$$\theta_{t_{j+1}} = \theta_{t_j} - l_r \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}, \tag{A.23}$$

where $\hat{m}_{t_j} = \frac{m_{t_j}}{1-\varphi'_{t_j}}$, and $\hat{v}_{t_j} = \frac{v_{t_j}}{1-\rho'_{t_j}}$. From Eq. (A.23), one has

$$\left(\theta_{t_{j+1}} - \theta^*\right)^2 = \left(\theta_{t_j} - \theta^* - l_r \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2, \tag{A.24}$$

$$= \left(\theta_{t_j} - \theta^*\right)^2 - 2\left(\theta_{t_j} - \theta^*\right) l_r \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}, + \left(l_r \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2, \tag{A.25}$$

$$\frac{\left(\theta_{t_j} - \theta^*\right)^2 - \left(\theta_{t_{j+1}} - \theta^*\right)^2}{2l_r} = \left(\theta_{t_j} - \theta^*\right) \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon} + l_r \left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2, \tag{A.26}$$

$$\frac{\left(\theta_{t_j} - \theta^*\right)^2}{2l_r} \geq \left(\theta_{t_j} - \theta^*\right) \frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon} + l_r \left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2. \tag{A.27}$$

Assuming that the loss function $Loss_{\theta_{t_j}}$ is convex in some domain, then $\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}$ is also convex in that domain, and by Theorem A.2,

$$loss_m^v(\theta_{t_j}) - loss_m^v(\theta^*) \leq \left(\theta_{t_j} - \theta^*\right)^T \nabla loss_m^v(\theta_{t_j}). \tag{A.28}$$

Substitution Eq. (A.28) into Eq. (A.27) we have

$$\frac{\left(\theta_{t_j} - \theta^*\right)^2}{2l_r} \geq loss_m^v(\theta_{t_j}) - loss_m^v(\theta^*) + l_r \left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2, \tag{A.29}$$

$$\implies loss_m^v(\theta_{t_j}) - loss_m^v(\theta^*) \leq \frac{\left(\theta_{t_j} - \theta^*\right)^2}{2l_r} - l_r \left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right)^2. \tag{A.30}$$

Moreover, since sigmoid activation function is bounded between $[0, 1]$ and for each pay of inputs (x, t), the exact solution U is constant,

$$Loss_{t_j} = \frac{1}{2} \|U(x,t) - \hat{U}(x,t,\theta_j)\|^2, \tag{A.31}$$

$$= \frac{1}{2} \| U(x,t) - g\left(W_4 g\left(W_3 g\left(W 2 g\left(W_1 Y_1 + b_1\right) + b_2\right) + b_3\right) + b_4\right) \|^2. \tag{A.32}$$

If g is the sigmoid activation function given by $g(y) = \frac{1}{1 + e^{-y}}$, then the range of the output is $[0, 1]$ which is compact and this concludes boundedness of the loss function when considering the sigmoid activation function.

$$\|Loss_{t_j}\|_{W^{2,2}(\Omega)} \leq C < \infty \implies \|Loss_{\theta_{t_j}}\|_{W^{2,2}(\Omega)} \leq C_1, \tag{A.33}$$

From Eq. (A.33), $\hat{M}_{t_j}$ and $\sqrt{\hat{V}_{t_j}}$ are bounded and hence $l_r \left(\frac{\hat{m}_{t_j}}{\sqrt{\hat{v}_{t_j}} + \epsilon}\right) \leq 1$. Therefore,

$$\|\theta_{t_{j+1}} - \theta_{t_j}\|_{W^{2,2}(\Omega)} \leq l_r. \tag{A.34}$$

From Eq. (A.30),

$$\|Loss(\theta_{t_j}) - loss(\theta^*)\|_{W^{2,2}(\Omega)} \leq \frac{l_r}{2} + \frac{l_r}{2} = l_r. \tag{A.35}$$

Summing over the sequence of parameters gives

$$\sum_{j=1}^{T} \|Loss(\theta_{t_j}) - loss(\theta^*)\|_{W^{2,2}(\Omega)} \leq l_r T. \quad \square \tag{A.36}$$

A.4. Proof of Theorem 3.1

The proof of the Kolmogorov extension theorem can be divided into two steps, the construction of a measure on a semi-ring and the extension of the measure to the Borel σ-algebra.

Let (Ω, F) be a measurable space, where Ω is a non-empty set and F is a σ-algebra on Ω. Let P be a family of probability measures on (Ω, F) that is consistent in the sense that for any finite collection $A_1, A_2, \ldots, A_n \in F$, the joint probability measure of these events assigned by any measure in P is equal to the product of the measures of these events assigned by the other measures in P. That is,

$$P(\cap_{i=1}^n A_i) = \prod_{i=1}^n P_i(A_i),$$

where P_i is any measure in P.

Let B be the Borel σ-algebra on the product space Ω^N, where N is a countably infinite index set. Consider the set function μ defined on the cylinder sets of B as follows:

$$\mu(A_1 \times A_2 \times \cdots \times A_n \times \Omega \times \Omega \times \cdots) = \prod_{i=1}^n P(A_i),$$

where $A_1, A_2, \ldots, A_n$ are any events in F. It can be shown that μ is a consistent set function, meaning that it satisfies the same consistency condition as the family P. By the Carathéodory extension theorem [31], μ can be extended to a unique probability measure on the σ-algebra generated by the cylinder sets, which is the Borel σ-algebra B.

It remains to show that the extension of μ to B agrees with the measures in P on the finite-dimensional σ-algebras. This can be done using a monotone class argument, which involves showing that μ is a monotone class of probability measures that contains P, and therefore it must be equal to the σ-algebra generated by P, hence the proof.

Theorem 3.1 also gives an explicit formula for the number of hidden neurons required to achieve a given level of approximation accuracy, which depends on the smoothness of the function $f(x)$. To generate the trial solution, we denote the artificial neural network (ANN) by $\phi(s,t)$ at (s,t), then according to Theorem 3.1, a trial approximate solution to the differential equation (DE) can be given by,

$$\phi(s,t) = \Delta t P(s) + s \Delta t Net(s,t,P^*), \tag{A.37}$$

where $(s,t) \in [0,\infty) \times [0,T)$ and $P^* = \{\alpha, \omega, \eta, \beta\}$, is an unknown set of parameters to be determined such that $(\alpha, \omega, \eta, \beta) \in \mathbb{R}^m$ with m the number of neurons in the hidden layer.

Note that, because of the complications in getting the analytical value of $\phi(s,t)$ under most market conditions, we focus on numerical methods to approximate solutions PDEs which is often less complicated to handle under normal market conditions. $\square$

A.5. Proof of Theorem 4.4

Let $\Omega \subset \mathbb{R}^{dim_{in}}$ be a bounded Lipschitz domain. We use an affine transformation $Z_h : \mathbb{R}^{dim_{in}} \to [0,1]^{dim_{in}}$ given by:

$$Z_h = W_h A_{h-1} + b_h, \tag{A.38}$$

where $W_h \in \mathbb{R}^{dim_{in} \times dim_{in}}$ is an invertible matrix and $b_h \in \mathbb{R}^{dim_{in}}$ is a vector. This maps Ω to the domain $[0,1]^{dim_{in}}$ which is the range for the sigmoid activation function at the previous layer. For a function $g \in C^\infty(\Omega; \mathbb{R})$, use an extension theorem to extend g to a function $\tilde{g} \in W^{K+1,1}([0,1]^{dim_{in}})$. Define $\hat{g}(x) = g(Z_h^{-1}(x))$, thus, $\hat{g}$ is defined on $[0,1]^{dim_{in}}$.

According to De Ryck et al. [19], for any $\epsilon > 0$, there exists a neural network $u \in \mathcal{N}$ (a neural network with 2 layers and sigmoid activation) such that

$$\|u - \hat{g}\|_{W^{K+1,1}([0,1]^{dim_{in}})} < \epsilon. \tag{A.39}$$

Table 10
Training and test loss for different activation functions across various data sizes.

N	ReLU	Sigmoid	Tanh	Swish
1000	3.5264e−6 3.9032e−6	1.5890e−2 1.5213e−2	7.6941e−4 6.7865e−4	5.1325e−5 4.5938e−5
2000	4.2273e−6 1.6372e−6	2.3912e−3 2.3527e−3	2.7304e−4 2.7631e−4	4.2447e−6 4.2749e−6
4000	1.6873e−7 1.1812e−6	2.0923e−4 2.0548e−4	6.1759e−5 3.5371e−5	2.4438e−6 8.3194e−7
8000	1.5367e−7 1.1264e−7	3.0291e−5 2.9467e−5	1.6612e−5 7.0391e−6	2.5479e−6 1.6914e−7

Table 11
Activation functions for training neural networks.

Function	Domain	Codomain	Motivation
$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$x \in \mathbb{R}$	$(-1, 1)$	The tanh function offers outputs centered around zero, which can aid in balanced gradients. However, it has slower convergence than sigmoid.
$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$	$x \in \mathbb{R}$	$(0, 1)$	The sigmoid function was chosen for its fast convergence, which helps reduce training time in a PINN setting. Normalizing prices aligns them with the sigmoid's range, enhancing predictive performance.
$\text{softplus}(x) = \ln(1 + e^x)$	$x \in \mathbb{R}$	$(0, \infty)$	The softplus function provides a smooth, differentiable alternative to ReLU, useful for PINNs. However, it has slower convergence than sigmoid, making it less efficient for the task.
$\text{swish}(x) = x \cdot \frac{1}{1+e^{-x}}$	$x \in \mathbb{R}$	$(-\infty, \infty)$	Swish combines linear and non-linear behavior, offering smooth transitions. It has potential for deeper models but was less efficient than sigmoid in this PINN application.

Since $\tilde{g}$ is an extension of $\hat{g}$ and $\hat{g} = g \circ Z_h^{-1}$, this approximation implies that u approximates $\hat{g}$ and consequently g on Ω. To approximate g on Ω, note that:

$$\|u - g\|_{C^k(\Omega)} \leq \epsilon \cdot \max(1, \|QZ_h\|_{C^k(\mathbb{R}^{dim_{in}})}), \tag{A.40}$$

where $\|QZ_h\|_{C^k(\mathbb{R}^{dim_{in}})}$ accounts for the affine transformation.

Define a function $v : \mathbb{R}^{dim_{in}} \to \mathbb{R}^{dim_{in}}$ using the sigmoid function

$$v(x) = g_h(W_h Z_{h-1} + b_h) \circ g_{h-1}(W_{h-1} Z_{h-2} + b_{h-1}) \circ \ldots \circ g_1(W_1 X + b_1), \tag{A.41}$$

where g_h is the sigmoid function composed h times.

The function v diffeomorphic (i.e., v and v^{-1} and their partial derivatives on C^∞ exist) mapping, so $v(\Omega)$ is a bounded Lipschitz domain.

Consider $g(v^{-1})$, which is a function defined on $v(\Omega)$. By applying the result from the previous case, there exists a sequence $(u_m)_{m \in \mathbb{N}}$ of neural networks with 2 layers such that

$$\|u_m - g(v^{-1})\|_{C^k(v(\Omega))} \to 0 \text{ as } m \to \infty. \tag{A.42}$$

To approximate g on Ω, use the fact that

$$\|u_m(v) - g\|_{C^k(\Omega)} \leq BK \|u_m - g(v^{-1})\|_{C^k(v(\Omega))} \cdot (1 + \|g_h\|_{C^k(\mathbb{R}^{dim_{in}})})^k.$$

Since $\|g_h\|_{C^k(\mathbb{R}^{dim_{in}})} < 1$, this ensures

$$\lim_{m \to \infty} \|u_m(v) - f\|_{C^k(\Omega)} = 0.$$

Thus, neural networks with any number of layers $H > 2$ using the sigmoid activation function can approximate functions in $C^1(\Omega; \mathbb{R}^d)$ arbitrarily well. This completes the proof. □

To determine the most suitable activation function for the PDD approach, we trained the network with different activation functions across various data sizes and analyzed the average training and validation losses. Table 10 summarizes the results. For each N, the upper value represents the average training loss, while the lower value corresponds to the test loss. Based on these results, the ReLU activation function was identified as the most effective choice for the PDD approach.

A similar methodology was applied to determine the optimal activation function for the PINN approach, with the sigmoid activation being selected as the best choice.

For the definitions of some activation functions considered in this paper, see Table 11.

Data availability

Data will be made available on request.

References

[1] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations, 2017, arXiv preprint arXiv:1711.10561.

[2] M. Raissi, P. Perdikaris, E.G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.

[3] S.M. Nuugulu, F. Gideon, K.C. Patidar, A robust numerical solution to a time-fractional Black–Scholes equation, Adv. Difference Equ. 1 (2021) 123.

[4] I. Syata, D.C. Lesmana, H. Sumarno, Numerical method for determining option price with risk adjusted pricing methodology (RAPM) volatility model, Appl. Math. Sci. 9 (134) (2015) 6697–6705.

[5] P. Wilmott, S. Howson, S. Howison, J. Dewynne, The Mathematics of Financial Derivatives: A Student Introduction, Cambridge University Press, 1995.

[6] D. Ševčovič, M. Žitňanská, Analysis of the nonlinear option pricing model under variable transaction costs, Asia-Pacific Financ. Mark. 23 (2) (2016) 153–174.

[7] S. Eskiizmirliler, K. Günel, R. Polat, On the solution of the Black–Scholes equation using feed-forward neural networks, Comput. Econ. 58 (2021) 915–941.

[8] J.A.C. González, Solution of the Black–Scholes equation using artificial neural networks, J. Phys.: Conf. Ser. 1221 (1) (2019) 012044, IOP Publishing.

[9] M.V. Klibanov, K.V. Golubnichiy, A.N. Nikitin, Application of neural network machine learning to solution of Black–Scholes equation, 2021, arXiv preprint arXiv:2111.06642.

[10] S.M. Nuugulu, K.C. Patidar, D.T. Tarla, A physics informed neural network approach for solving time fractional Black–Scholes partial differential equations, Optim. Eng. (2024) 1–30.

[11] A.C. Müller, S. Guido, Introduction To Machine Learning with Python: A Guide for Data Scientists, O'Reilly Media, Inc., 2016.

[12] S. Sandhya, Neural Networks for Applied Sciences and Engineering: From Fundamentals To Complex Pattern Recognition, Auerbach Publications, 2016.

[13] A.C. Charu, Neural Networks and Deep Learning: A Textbook, Spinger, 2018.

[14] D.P. Kingma, B. Jimmy, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[15] D. Lee, J. Kim, K. Jung, Improving object detection quality by incorporating global contexts via self-attention, Electron. 10 (1) (2021) 90.

[16] F. Girosi, T. Poggio, Representation properties of networks: Kolmogorov's theorem is irrelevant, Neural Comput. 1 4 (1989) 465–469.

[17] Z. Allen-Zhu, Y. Li, Z. Song, A convergence theory for deep learning via over-parameterization, in: International Conference on Machine Learning, 2019, pp. 242–252.

[18] B. Fang, D. Klabjan, Convergence analyses of online adam algorithm in convex setting and two-layer relu neural network, 2019, arXiv preprint arXiv:1905.09356.
[19] T. De Ryck, S. Lanthaler, S. Mishra, On the approximation of functions by tanh neural networks, Neural Netw. (2021) 732–750.
[20] N. Doumèche, G. Biau, C. Boyer, Convergence and error analysis of PINNs, 2023, arXiv preprint arXiv:2305.01240.
[21] D. Gazoulis, I. Gkanis, C.G. Makridakis, On the stability and convergence of physics informed neural networks, 2023, arXiv preprint arXiv:2308.05423.
[22] P.J.M. Ali, R.H. Faraj, E. Koya, Data normalization and standardization: a technical report, Mach. Learn. Tech. Rep. 1 (1) (2014) 1–6.
[23] J. Pathak, K.R. Bailey, C.E. Beebe, S. Bethard, D.S. Carrell, P.J. Chen, D. Dligach, C.M. Endle, L.A. Hart, P.J. Haug, S.M. Huff, Normalization and standardization of electronic health records for high-throughput phenotyping: the SHARPn consortium, J. Am. Med. Informatics Assoc. 20 (e2) (2013) 341–348.
[24] Y. Hilpisch, Python for Finance: Analyze Big Financial Data, O'Reilly Media, Inc, 2014.
[25] R. Kokoszczynski, S. Paweł, R. Ślepaczuk, Midquotes Or Transactional Data? the Comparison of Black Model on HF Data, Vol. 15, Economic Sciences Working Paper, University of Warsaw, 2010.
[26] A. Sinha, J. Poonolly, A. Gayathiri, D. Janarthanan, Analysis of option prices using Black Scholes model, J. Emerg. Technol. Innov. Res. 5 (2018) 12–21.
[27] J. Nocedal, S.J. Wright, Numerical Optimization, Springer New York, New York, NY, 1999.
[28] A.A. Ahmadi, ORF 523: Graduate Course in Operations Research and Financial Engineering, Princeton University, 2024, https://aaa.princeton.edu/orf523.
[29] Y. Shin, J. Darbon, G.E. Karniadakis, On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs, 2020, arXiv preprint arXiv:2004.01806.
[30] T. Hytönen, J. Van Neerven, M. Veraar, L. Weis, Analysis in Banach Spaces, Springer, Berlin, 2016.
[31] J. Kupka, The Carathéodory extension theorem for vector valued measures, Proc. Amer. Math. Soc. 72 (1) (1978) 57–61.