university of koblenz
Mathematics / Natural Sciences

A Physics Informed Deep Learning Approach for Pricing Options with Stochastic Volatility and Correlation

Master thesis

Mathematical Institute
University of Koblenz

Submitted in partial fulfillment of
the requirements for the degree
of Master of Science by

Leela Prasad Gurugubelli

Matriculation No.: 221100436

First supervisor:	PD. Dr. Rockenfeller
	Mathematical Institute, University of Koblenz
Second supervisor:	PD. Dr. Long Teng
	Mathematical Institute, Bergische Universität Wuppertal
Submission Date:	January 26, 2024

Declaration

I, Leela Prasad Gurugubelli, hereby affirm that I am the sole author of this thesis and have not received any external assistance, except for the quoted literature and other sources explicitly referenced in this document. I have diligently identified and distinctly listed all literature and sources utilized in the creation of this academic work, both verbatim and in content. This thesis has not been submitted or published in a comparable form in any other examination process. The written version presented here is identical to the electronic version.

G. Leela prasad

Leela Prasad Gurugubelli

Koblenz, January 26, 2024.

Acknowledgement

I would like to acknowledge and express sincere gratitude to my supervisors PD. Dr. Rockenfeller and PD. Dr. Long Teng, for their continuous support during my research, and their patience and motivation. Their guidance helped me through the research and writing of this thesis. I am incredibly grateful for what they offered me and could not have imagined having better supervisors for my Master's thesis.

Further, I thank all my Professors and Lecturers who supported me at different stages of my academics. To all my friends and seniors, a special thanks to all of you for helping me through various stages of my life, personally and academically.

I would like to acknowledge the utilization of ChatGPT, a valuable tool that greatly assisted me in obtaining information and provided assistance in paraphrasing and sentence correction tasks.

Finally, I would like to thank my parents, Venkata Rao and Satyavathi, for supporting me through my decisions and encouraging me to pursue my interests, and my sister Swathi Lakshmi for their support throughout my life. Also, special mention to my brother-in-law Y A Naidu, who has always been an inspiration to me.

Abstract

This thesis explores the application of a deep learning method called PINNs for numerically solving the European call option price in both the Heston model and the extended Heston model with stochastic correlation. We are extending the Heston model by imposing a stochastic correlation driven by a stochastic correlation process between the underlying asset and the stochastic volatility. We are using the reformulated system to generate the pricing PDE with the help of Feynman Kac's theorem, resulting in a non-linear PDE with some mixed derivative terms.

The study applies PINNs to solve the resulting PDEs, focusing on European call option prices across various stocks. This study aims to assess the suitability of the PINNs for the Heston and extended Heston models. The initial part demonstrates how PINNs infer solutions to PDEs, with existing studies predominantly focusing on the Heston model. The effectiveness of PINNs is evaluated, with a specific emphasis on At the Money (ATM) and In the Money (ITM) stocks. To validate this, an online Heston model calculator has been used for European call option prices, which have been computed using the conventional numerical approach of the Finite Difference Method (FDM).

Our study with numerical results are validated that PINNs can solve the Heston model and extended Heston model with different sets of parameters on different stocks. The collective outcomes and experimental data reveal that PINNs offer a promising approach for accurately modeling In the Money (ITM) stocks in the Heston and extended Heston models while highlighting limitations in the case of At the Money (ATM).

Keywords: Heston model, Extended Heston model, Stochastic correlation, Physics Informed Neural Networks (PINNs), Partial Differential Equations (PDEs), European call option, At the Money (ATM), In the Money (ITM), Out of the Money (OTM).

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In the present financial world, the European call option price plays a very important role in shaping investment strategies and risk management. This financial instrument gives the holder the right, but not the obligation, to buy an underlying asset at a predetermined price before a specified maturity date.[12] The European call option prices are influenced by the current stock price, strike price, risk-free interest rate, time to maturity, and volatility. Determining the price of European call options requires sophisticated models, and the Heston model stands out as an enhancement over the classic Black-Scholes [16] framework. The Heston model is implemented by incorporating stochastic volatility to overcome the limitations of the Black Scholes model[18]. This feature enables the Heston model to capture the dynamic and fluctuating nature of market volatility more realistically. By considering phenomena such as volatility smiles and skews, the Heston model provides a refined and precise representation of European call option prices in the present market conditions. As financial markets evolve, the adaptability and improvements offered by the Heston model over the Black-Scholes model contribute significantly to making more informed decisions in the realm of option pricing, reflecting the complexities of today's financial environment.[18]

The Heston model, introduced in the main paper by Steven L Heston [18], is a prominent stochastic volatility model widely employed in equity price modeling. Its primary application is the pricing of options, a type of financial derivative. Unlike other financial assets like equities, an option's value isn't determined by the asset's current value but rather by the changes in the underlying asset's price. The Heston model is an extension of the Black-Scholes model by incorporating stochastic volatility, which is modeled using an Ornstein - Uhelenbeck (OU) process.[25][15] The key feature of the Heston model is its consideration of stochastic volatility as opposed to a constant one, setting it apart from models like the Black-Scholes model that assume constant volatility.

In this thesis, by using time-dependent correlation we are extending the Heston model. The Heston model provides a constant correlation between the two Brownian motions. Here In the extended Heston model, we are trying to implement a different stochastic correlation process between the underlying asset and the volatility. The reason, why we incorporate a stochastic correlation into the Heston model is given as follows firstly, because correlation affects skew of the implied volatility, introducing a non-constant correlation can certainly improve the calibration [40]. Secondly, a couple of papers suggested that the correlation between financial quantities can not be a constant [32] [41]. We can improve the Heston model by adding the stochastic correlation to the model.

To solve the Heston model or other stochastic models Partial Differential Equations (PDEs) there are several numerical methods are available named Finite Difference Method (FDM), Finite Element Method (FEM), and Finite Volume Method (FVM). In the present world Machine learning and Deep learning play a significant role in financial mathematics in terms of solving non-linear problems. Especially solving Non-linear Partial Differential Equations (NPDEs) is getting easier with Machine learning and Deep learning techniques, and one of the deep learning methods is called PINNs.[34] PINNs are the type of universal function approximators that can embed the knowledge of any physical laws that govern a given data set in the learning process and can be described by PDEs.[34]

This thesis studies an implementation of the PINNs deep learning method on the Heston model PDE to calculate the European call option prices. Then we are extending the Heston model by imposing the stochastic correlation between asset price and volatility. We will consider two correlations, one between asset price and volatility and another one between asset price and asset price to volatility. In this Stochastic Differential Equation (SDE) System, we are using 3 Brownian motions for spot, volatility, and correlation processes. We are considering the correlation between spot and volatility as ρ_t, the correlation between spot and spot/volatility as ρ_{cs}, and the correlation between variance and spot/volatility as ρ_t and ρ_{cs}. This means that the spot/volatility correlation is a separate mean-reverting stochastic variable that is itself correlated with the spot.[20] The correlation in the financial market must be modeled in a nonlinear way even randomly by using a mean-reverting stochastic process like modeling volatility or interest rate. This is to say that using a stochastic correlation is more realistic than using a constant correlation. After extending the Heston model the PDE becomes high dimensional and it's difficult to solve.

We extend the Heston model by introducing correlations through bounded Jacobi processes.[25] This improved fit is primarily attributed to the correlation processes we have incorporated into the model. After developing the Stochastic Differential Equation (SDE) system, we derive the PDE of the Heston model and the extended

Heston model using the Feynman-Kac theorem. We solve the derived PDEs using a Deep learning method called PINNs. Through this model, we demonstrate that the PINNs are a better fit to solve the Heston model and extended Heston model to calculate the European call option price at In the Money (ITM) values. For Out of the Money (OTM) values PINNs provide promising results, and for At the Money (ATM) values our proposed PINNs model does not provide accurate results compared to the In the Money (ITM) values.

In this work, we study the deep learning method called PINNs for solving the Heston model and extended Heston model by including the stochastic correlation which is a three-dimensional nonlinear PDE. We explore the PINNs method on the Heston model and extended Heston model, whether it's a suitable method to solve these models or not by comparing it with the European call option prices of the Heston model calculated by using the Finite Difference Method (FDM).

The rest of the thesis is organized as follows. Chapter 2 provides foundational topics to understand Options and different processes in finance. Chapter 3 explains the introduction to the Heston model and derivation of the pricing PDE. In Chapter 4, we introduce PINNs, exploring their applications, the methodology behind solving nonlinear PDEs, and an examination of pricing European call options through this approach. Chapter 5 details the step-by-step process employed to solve the pricing PDE using PINNs. Chapter 6 extends the Heston model by incorporating a stochastic correlation between the underlying asset and stochastic volatility. This extension is followed by the derivation of the pricing PDE using the Feynman-Kac theorem. Moving on to Chapter 7, we implement a series of numerical results, highlighting European call option prices for various strike values of the extended Heston model and the Heston model. In Chapter 8, we discuss in detail, the series of numerical results presented earlier. Finally, Chapter 9 serves as the conclusion, summarizing key findings and insights obtained throughout the thesis.

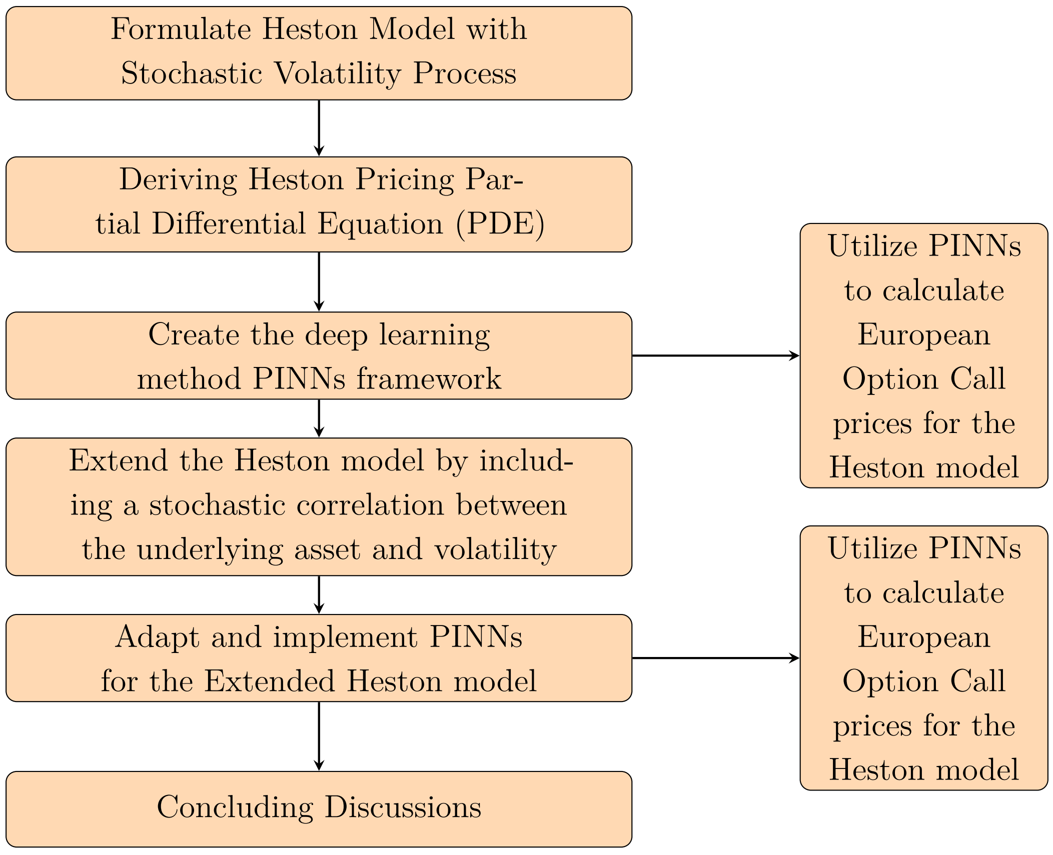

Formulate Heston Model with Stochastic Volatility Process
Deriving Heston Pricing Partial Differential Equation (PDE)
Create the deep learning method PINNs framework
Extend the Heston model by including a stochastic correlation between the underlying asset and volatility
Adapt and implement PINNs for the Extended Heston model
Concluding Discussions
Utilize PINNs to calculate European Option Call prices for the Heston model
Utilize PINNs to calculate European Option Call prices for the Heston model

Figure 1.1: Algorithm for Heston model and Extended model solution using PINNs.

Chapter 2

Preliminary

In this chapter, we introduce the options, different types of options, and some basic definitions of important terms in financial mathematics. There are different models and processes which are important to understand the Heston model discussed here.

2.1 Introduction to Options

Options are defined as it is a type of financial derivative, which means that the option price is not directly based on the underlying asset's price.[12] The valuation of an option depends on the underlying asset price, time expiration, volatility, the risk-free rate of interest, and the strike price of the option. Options are between the buyer and the seller, contracts that give option buyers the right to buy or sell a security at a predetermined price on or before an expiration date. A Call option gives the holder the right to buy a stock and a Put option gives the holder the right to sell a stock.[12]

Strike Price

Options are derivatives, that give the holder or investors the right, but not the obligation, to buy or sell some underlying security at some point in the future at a pre-specified price. That particular pre-specified price is called the Option strike price or exercise price. For call options, the strike price is where the underlying asset can be bought by the option holder. for put options, the strike price is the price at which the underlying asset can be sold.[30]

Call Option

A Call option provides the holder with the right, though not the obligation, to purchase the underlying asset at the predetermined strike price on or before the expiration date. The expiration date marks the point at which the option ceases to be valid. The value of a call option tends to increase as the price of the underlying security rises, given that calls exhibit a positive delta.[12]

For example, Suppose you are interested in a company Euro industries stock, currently priced at €50 per share, and you are expecting an increase in future price. Without directly purchasing the stock, you decided to buy a call option with a strike price of €55 and an expiration date set for three months from now, including an option premium cost of €3 per share. By the expiration time, the stock price surpasses the €55 strike price, and your call option gains value. For example, if the stock price reaches €60, you, as the call option holder, can exercise the option, buying shares at the predetermined €55 strike price, even though the market price is higher. Consequently, your profit amounts to the difference between the market price and the strike price, minus the option premium, resulting in a total profit of €200 if holding 100 shares. On the other hand, if the stock price fails to rise and remains below the €55 strike price, the call option may not be exercised. In this scenario, you are not obligated to buy the stock at the higher strike price, and you may incur a loss equivalent to the option premium paid, which is €3 per share.

Put Option

In contrast to call options, a put option grants the holder the right, without the obligation, to sell the underlying stock at the predetermined strike price on or before the expiration date. Holding a long put is essentially assuming a short position in the underlying security, as the put increases in value when the price of the underlying falls (indicating a negative delta).

For example, consider a scenario where an investor is concerned about potential price declines in Company Euro industries stock, currently trading at €70 per share. The investor purchases a put option with a strike price of €65 and an expiration date two months from now, for €2 per share. If, at the expiration date, the stock price has fallen below €65 (e.g., €60), the put option becomes valuable. The investor can sell shares at the higher strike price, resulting in a profit of the difference between the strike price and the market price, minus the option premium (e.g., (€65 - €60) - €2 = €3 per share). However, if the stock price does not decrease and remains above €65, the put option may not be exercised, and the investor incurs a loss equal to the option premium paid. In summary, a put option serves as a form of insurance, allowing the holder to sell the underlying asset at a predetermined price, thereby mitigating potential losses in a declining market, with the maximum loss limited to the initial cost of the option premium.

European Option

A European option is defined as, firstly European option a type of option contract that can be executed on the expiration date. a European option does not allow the investor to exercise the option early and buy or sell the underlying security, such as a stock, even if its price has changed. Instead, the Call or Put action associated with a European option only occurs on the specific date of option maturity.

American Option

In contrast, an American option is a different variant of an options contract that offers greater flexibility. This type of option can be exercised at any point up to and including the date of expiration. Investors holding American options have the privilege of choosing when to exercise the option based on favorable market conditions. A European option can be exercised only at the expiration date, on the other hand, an American option can be exercised at any time or the expiration date.

Out of the Money (OTM)

The Out of the Money (OTM) call options are the stocks that are lower than the Strike price, these stocks are called Out of the Money call Stocks. In contrast Out of the Money (OTM) put options are the stocks that are higher than the Strike price, these stocks are called the Out of the Money put stocks.

For example, the strike price of a call option is 40, the underlying stock price value is 30, and then the stock is called the Out of the Money (OTM). For the put option, if the stock value is 50, the stock value is higher than the strike price so the stock is called as the Out of the money (OTM).

At the Money (ATM)

The At the Money (ATM) call options are the stocks that are the same as the Strike price, these stocks are called At the Money call Stocks.

For example, the strike price of a call option is 40, the underlying stock price value is 40, and then the stock is called the At of the Money (ATM).

In the Money (ITM)

The In the Money (ITM) call options are the stocks that are higher than the strike price, these stocks are called Out of the Money call Stocks. In contrast In the Money (ITM) put options are the stocks that are lower than the strike Price, these stocks are called the In the Money put stocks.

For example, the strike price of a call option is 40, the underlying stock price value is 50, and then the stock is called the In the Money (ITM). For the put option, if the stock value is 30, the stock value is lower than the strike price so the stock is called the In of the Money (ITM).

2.2 Stochastic Processes and Integrals in Finance

In this section, we are introducing some important processes and different option pricing models in financial mathematics to understand the Heston model better and clearly. We are going to use these models and processes in this thesis.

Ornstein-Uhlenbeck Process

The Ornstein-Uhlenbeck process [13] [14] was proposed by Uhlen-beck and Ornstein in 1930 as an alternative to Brownian motion in a physical modeling context. In physics, the velocity of a moving particle is traditionally defined as the time derivative of its position. However, if the position of a particle is described by Brownian motion, then the time derivative does not exist. The Ornstein-Uhlenbeck process is an attempt to overcome this difficulty by modeling the velocity directly.[28] This process is a stochastic model commonly used in various fields, including finance and physics to describe the random evolution of a continuous-time variable over time.[28]

The Ornstein-Uhlenbeck Processes can be defined as

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t \tag{2.1}$$

where X_t is the process variable at time t, θ is the speed of reversion towards the mean, μ is the mean or long-term average of the process, σ is the volatility, dt represents the differential change in time, and dW_t is a Wiener process or Brownian motion.

Alternatively, we can write X in terms of stochastic integral :

$$X_t = \mu\left(1 - e^{-\theta t}\right) + \sigma e^{-\theta t}\int_0^t e^{\theta s}\, dW_t + X_0 e^{-\theta t}, \quad t \geq 0 \tag{2.2}$$

Cox–Ingersoll–Ross (CIR) model

Cox–Ingersoll–Ross (CIR) model was introduced in 1985 by John C. Cox, Jonathan E. Ingersoll and Stephen A. Ross as an extension of the Vasicek model [42] of financial mathematics, The Cox–Ingersoll–Ross (CIR) model describes the dynamics of interest rate evolution, and it is a type one-factor model or short-rate model as it illustrates the interest rate movements. This can be used in the valuation of interest rate derivatives. [9]

The Cox–Ingersoll–Ross (CIR) process can be defined as

$$dr_t = \alpha(\mu - r_t)\, dt + \sqrt{r_t}\, dW_t, \quad r(0) = r_0 \tag{2.3}$$

where r_t is the process variable at time t, α is the speed of reversion towards the mean, μ is the mean or long-term average of the process, σ is the volatility, dt represents the differential change in time, and dW_t is a Wiener process or Brownian motion.

The unique solution to the CIR model is given by:

$$r_t = r_s + \int_s^t \alpha(\mu - r_u)\, du + \int_s^t \sqrt{r_u}\, dW_u, \quad s < t \tag{2.4}$$

The Bounded Jacobi Process

The Bounded Jacobi process is used to model the stochastic correlation between the Brownian motions. We can extend the Heston model using this process. The Bounded Jacobi process is defined as:

$$dX_t = \theta(\mu - X_t)dt + \sigma\sqrt{1 - \rho_t^2}dW_t \tag{2.5}$$

Ito's Lemma

The Ito lemma formula is an identity used in Ito calculus to find the differential of a time-dependent function of a stochastic process. Let X_t be an Itô process $dX_t = U_t dt + V_t dB_t$. Suppose $g(x) \in C^2(\mathbb{R})$ is a twice continuously differentiable function (in particular, all second partial derivatives are continuous functions). Suppose $g(X_t) \in L^2$, then $Y_t = g(X_t)$ is again an Itô process. [21]

$$dY_t = \frac{\partial g}{\partial t}(X_t)dX_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}(X_t)(dX_t)^2. \tag{2.6}$$

Using the notational convention for $dX_t = U_t dt + V_t dB_t$ and $(dX_t)^2$, we can rewrite the Itô formula as [21]

$$dY_t = \left(\frac{\partial g}{\partial t}(X_t)U_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}\right) dt + \frac{\partial g}{\partial x}(X_t)V_t dB_t. \tag{2.7}$$

Black Scholes Model

The Black Scholes model [16], also known as the Black Scholes Merton model (BSM), is one of the most important concepts in financial theory. The Black-Scholes model is a mathematical model for the dynamics of a financial market containing derivative investment instruments, using various underlying assumptions. The parabolic PDE from the model is called the Black-Scholes model equation and helps to calculate the option price of European-style options. This formula indicates that an option has a specific value based on the risk associated with the underlying asset and its expected return, with the latter being replaced by the risk-neutral rate in the calculations. In simpler terms, it provides a way to estimate the fair price of options by considering market risks and expected returns.[8]

The Black-Scholes formula for the price (C) of a European call option is given by:

$$C = S_0 N(d_1) - K e^{-rT} N(d_2) \tag{2.8}$$

where

$$\begin{aligned} d_1 &= \frac{\ln(S_0/X) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \\ d_2 &= d_1 - \sigma\sqrt{T}, \\ N(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt, \end{aligned}$$

In the above equation, S_0 is the current stock price, K is the option strike price, T is time to expiration (in years), r is the risk-free interest rate, σ is the volatility of the stock price.

The Black-Scholes partial differential equation is given by:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0 \tag{2.9}$$

where V = V(t, S) is the option price as a function of time t

Evolution of Heston model

Many praises have been aptly used to describe Black and Scholes's [16] contribution to option price theory. Despite the evolution of option theory over time, the initial Black-Scholes formula for a European call option continues to be the most successful and extensively applied approach [18]. While the Black-Scholes model has proven to be successful in determining option prices, it is not without its limitations. Black Scholes's effectiveness is notably diminished, particularly when applied to foreign currency options.[18] To overcome these limitations and the presence of uncorrelated volatility about spot returns limits the ability to capture significant skewness effects. In response to this limitation, Steven L. Heston proposed a stochastic volatility model that departs from the Black-Scholes model. This model offers a closed-form solution for pricing European call options, accounting for the correlation between spot and volatility. The Heston model describes the evolution of the volatility of an underlying asset in addition to its price. Its applicability extends beyond equity options, encompassing bond options and currency options as well.[18]

Chapter 3

Heston model

We have already seen the evolution of the Heston model in the previous chapter. In this chapter, we are introducing the Heston model and the derivation of the Heston model PDE.

3.1 Introduction and derivation of Heston Model

Heston stochastic volatility model [18] is a widely used mathematical model in finance to describe the dynamics of an asset price and volatility. It was introduced by Steven L Heston in 1993 and is used to capture the phenomenon of volatility clustering observed in the financial market.[18] The model assumes that the stock price and its volatility follow stochastic processes under the risk-neutral measure. In the risk-neutral measure, a geometric Brownian motion typically describes the stock price process, and the volatility process follows a mean-reverting process. Here's the basic formulation of the Heston stochastic volatility model.

Stochastic price (Under risk-neutral measure):

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^s \tag{3.1}$$

Volatility (Under risk-neutral measure):

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t} dW_t^v \tag{3.2}$$

Correlation:

$$dW_t^s dW_t^v = \rho dt \tag{3.3}$$

Where S_t is the price of the underlying asset at time t, r is risk-free rate, V_t is variance at time t, μ is long-term variance, κ_v is variance mean-reversion speed, σ is volatility of the variance process, dW_t^s, dW_t^v are two correlated Brownian processes with a correlation coefficient.

Brownian motions W_t^s and W_t^v are correlated with a constant $\rho \in [-1, 1]$. Note that under risk-neutral measures, the market price of volatility risk is embedded in the parameters of dV_t. In the above equations $k_v >= 0$, $\mu >= 0$ and $\sigma >= 0$. The parameter k_v controls the mean reversion speed of the volatility. The Heston model possesses a crucial characteristic known as stochastic volatility, enabling it to accurately replicate the implied volatility smile observed in numerous financial markets.[39]

If $2k_v\mu > \sigma^2$ the so-called Feller condition. We explore scenarios where the Feller condition is satisfied, ensuring the guaranteed positivity of $\mu(t)$, and cases where the Feller condition may not be met, potentially leading to the possibility of $\mu(t)$ reaching zero. Achieving the Feller condition can be challenging in practical applications. The consideration of both scenarios allows for a comprehensive analysis of various financial scenarios.[39]

Based on the above Stochastic Differential Equations the corresponding PDE can be derived as

$$\frac{\partial C}{\partial t} + rS_t\frac{\partial C}{\partial S_t} + k_v(\mu - V_t)\frac{\partial C}{\partial V_t} + \frac{1}{2}\left(V_tS_t^2\right)\frac{\partial^2 C}{\partial S_t^2} + \frac{1}{2}\left(\sigma^2 V_t\right)\frac{\partial^2 C}{\partial V_t^2} + \left(\sigma V_t S_t \rho\right)\frac{\partial^2 C}{\partial S_t \partial V_t} - rC = 0 \tag{3.4}$$

Feynman-Kac Theorem

In the 1940s, Richard Feynman and Marc Kac discovered a connection between Partial Differential Equations and a stochastic process. The Feynman–Kac formula resulted, which proves rigorously the real-valued case of Feynman's path integrals. [22]

Suppose that x_t follows the stochastic process :

$$dx_t = \mu(x_t, t)dt + \sigma(x_t, t)dW_t^Q \tag{3.5}$$

where W_t^Q is Brownian motion under the measure Q. Let $V(x_t, t)$ be a differentiable function of x_t and t, and suppose that $V(x_t, t)$ follows the PDE given by [22]

$$\frac{\partial V}{\partial t} + \mu(x_t, t)\frac{\partial V}{\partial x} + \frac{1}{2}\sigma^2(x_t, t)\frac{\partial^2 V}{\partial x^2} - r(x_t, t)V(x_t, t) = 0 \tag{3.6}$$

with the boundary condition $V(X_T, T)$. The theorem asserts that $V(x_t, t)$ has the solution

$$V(x_t, t) = \mathbb{E}^Q\left[e^{-r(T-t)}r(x_u, u)du\Big|\mathcal{F}_t\right]V(X_T, T) \tag{3.7}$$

where the expectation is taken under the measure Q that makes the stochastic term in Equation (3.5) Brownian motion. The generator of the process in (3.5) is defined as the operator [22]

$$A = \mu(x_t, t)\frac{\partial}{\partial x} + \frac{1}{2}\sigma^2(x_t, t)\frac{\partial^2}{\partial x^2} \tag{3.8}$$

so the PDE in $V(x_t, t)$ is written as

$$\frac{\partial V}{\partial t} + AV(x_t, t) - r(x_t, t)V(x_t, t) = 0 \tag{3.9}$$

Heston model Pricing PDE derivation

The Heston model is given by

$$dS_t = rS_t dt + \sqrt{V_t}S_t dW_t^s \tag{3.10}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t}dW_t^v \tag{3.11}$$

where :

$$dW_t^s\, dW_t^v = \rho\, dt \quad \text{[Spot and Volatility]}$$

To check conveniently the affinity, we reformulate the SDE system with respect to the independent Brownian Motions: We first rearrange the SDE system

$$dS_t = rS_t dt + \sqrt{V_t}S_t dW_t^s \tag{3.12}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t}dW_t^v \tag{3.13}$$

Which has a family of correlation matrices

$$C_t = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

performing a Cholesky-decomposition $C_t = LL^T$, It allows us to express the correlation matrix C_t as the product of a lower triangular matrix L and its transpose

$$L[1,1] = \sqrt{C_t[1]} = \sqrt{1} = 1$$

$$L[2,1] = \frac{C_t[2,1]}{L[1,1]} = \frac{\rho}{1} = \rho$$

$$L[2,2] = \sqrt{C_t[2,2] - L[2,1]^2} = \sqrt{1-\rho^2}$$

Where L is the Lower triangular matrix given by

$$L = \begin{pmatrix} 1 & 0 \\ \rho & \sqrt{1-\rho^2} \end{pmatrix}$$

Since our main aim is to impose the asset process and stochastic volatility process, Now the reformulated SDE system becomes

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^s \tag{3.14}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t}\rho dW_t^s + \sigma\sqrt{V_t}\sqrt{1-\rho^2}dW_t^v \tag{3.15}$$

Let's denote the value of the call option with a payoff function maturity T

$$f(S_t) = max(S_T - K, 0) \tag{3.16}$$

To derive the PDE we are using the Feynman-Kac theorem, which follows the

given formula from ito lemma (3.9)

$$\frac{\partial C}{\partial t} + AC(t, S, V) - rC(t, S, V) = 0 \tag{3.17}$$

Here $\frac{\partial C}{\partial t}$ represents the time derivative of the option price, A is the differential operator that involves dynamics of S and V, and r is the risk-free interest rate.

Using the Feynman-Kac theorem [38], the option price $C(t, S, V)$ can be expressed as an expectation under the risk-neutral measure

$$C(t, S, V) = e^{-r(T-t)}\mathbb{E}\left[f(C_T)\middle|S(t) = S, V(t) = V\right] = e^{-r(T-t)}\mathbb{E}\left[f(C_T)\right] \tag{3.18}$$

Here A is the differential operator that involves S and V dynamics. Apply ito Lemma to the option value C(t, S, V) to derive its dynamics. The generator of the process is

$$A = \sum_{i=1}^{n} \mu_i \frac{\partial}{\partial x_i} + 0.5\sum_{i=1}^{n}\sum_{j=1}^{n}(\sigma\sigma^{\mathrm{T}})_{ij}\frac{\partial^2}{\partial x_i \partial x_j} \tag{3.19}$$

now we are going to write our SDE system in matrix form

$$\begin{bmatrix} dS_t \\ dV_t \end{bmatrix} = \begin{bmatrix} rS_t \\ \kappa_v(\mu - V_t) \end{bmatrix} dt \times \begin{bmatrix} \sqrt{V_t}S_t & 0 \\ \sigma\rho\sqrt{V_t} & \sigma\sqrt{V_t}\sqrt{1-(\rho)^2} \end{bmatrix} \begin{bmatrix} dW_t^s \\ dW_t^v \end{bmatrix} \tag{3.20}$$

Using the above matrix form we are going to generate the process of A, now $\sigma\sigma^T$ becomes

$$\begin{bmatrix} \sqrt{V_t}S_t & 0 \\ \sigma\rho\sqrt{V_t} & \sigma\sqrt{V_t}\sqrt{1-(\rho)^2} \end{bmatrix} \times \begin{bmatrix} \sqrt{V_t}S_t & \sigma\rho\sqrt{V_t} \\ 0 & \sigma\sqrt{V_t}\sqrt{1-(\rho)^2} \end{bmatrix} = \begin{bmatrix} V_t(S_t)^2 & \sigma\rho V_t S_t \\ \sigma\rho V_t S_t & (\sigma^2 V_t) \end{bmatrix} \tag{3.21}$$

substitute the values into the equation(3.19)

$$A = rS_t\frac{\partial}{\partial S_t} + \left(k_v(\mu - V_t)\frac{\partial}{\partial V_t}\right) + \frac{1}{2}\left(V_t S_t^2\frac{\partial^2}{\partial S_t^2} + \sigma^2 V_t\frac{\partial^2}{\partial V_t^2} + 2\sigma V_t S_t\rho\frac{\partial^2}{\partial S_t \partial V_t}\right) \tag{3.22}$$

Now the PDE equation for (3.17) C = C(t,S,V,ρ) becomes

$$\frac{\partial C}{\partial t} + rS_t\frac{\partial C}{\partial S_t} + k_v(\mu - V_t)\frac{\partial C}{\partial V_t} + \frac{1}{2}\left(V_t {S_t}^2\right)\frac{\partial^2 C}{\partial S_t^2} + \frac{1}{2}\left(\sigma^2 V_t\right)\frac{\partial^2 C}{\partial V_t^2} + \left(\sigma V_t S_t \rho\right)\frac{\partial^2 C}{\partial S_t \partial V_t} - rC = 0 \tag{3.23}$$

In the above equation C is option price, t is Time, r is risk-free rate, S_t is price of the underlying asset at time t, V_t is variance at time t, ρ is correlation, k_v is variance mean-reversion speed, μ is long-term variance, and σ is volatility of the volatility.

The Heston model was developed to help price options while accounting for variations in the asset's price and volatility. When pricing options, one aspect to consider is market volatility and its effects on asset prices.[18] With a proper selection of parameters, the Heston model appears to be very promising and flexible at defining the option prices.[18]The Heston model stands out in comparison to the other models because the Heston model considers Volatility to be a random process. In Black Scholes model volatility is treated as a constant. The Heston model is capable of pricing European options, it can calculate options for both Call and Put values for the European system. The Heston model is versatile enough to describe stock options, bond options, and currency options.[18]

The Heston model is widely used in financial mathematics, at the same time, the Heston model has also been applied in medical research. Thomas F. Heston mentioned in his paper [19] about the Heston model applications in different medical fields. Particularly they are using ARCH (Autoregressive Conditional

Heteroskedasticity) and GARCH (Generalized ARCH) models. By using these models in cardiology, they can understand better variability in heart rates.[19]

Chapter 4

Physics Informed Neural networks (PINNs)

This chapter introduces the well-known Deep learning method called Physics Informed Neural Networks (PINNs) and its working process. We are going to discuss the reason behind using this particular method to solve PDEs and Nonlinear PDEs, the process of PINNs, and the applications of PINNs. Here we are going to discuss the PINNs methodology in detail.

4.1 What is PINNs?

PINNs are a scientific machine learning technique used to solve problems involving PDEs. Neural networks are widely used to solve problems in a variety of domains including computer vision, language processing, image processing, game theory, etc., The use of machine learning approaches in scientific computing including PDEs is relatively recent. The idea of leveraging prior knowledge of physics in the learning process of a Neural Network was introduced by Raissi.[36]

PINNs are a popular tool for solving PDEs in the present scenario. These innovative networks represent a breakthrough in the field of computational physics and engineering, offering a novel approach to solving complex physical problems while bridging the gap between data-driven machine-learning techniques and the governing equations of the physical world.[34][36]

PINNs are a crucial part of deep learning, designed to incorporate physical laws directly into neural networks. This is particularly useful for solving problems involving nonlinear PDEs. PINNs are excellent at handling PDEs, even when there are complicated boundary conditions, making them well-suited for dealing with high-dimensional problems. These networks use a mix of data-driven and physics-driven elements in their learning process, allowing them to learn from available data while respecting fundamental physical principles. This proves beneficial, especially when using market data to fine-tune the model to match real-world observations accurately.[36]

When it comes to the nonlinear PDEs, we can use the supervised settings that we discussed above. While solving the PDEs we are going to train the model using the mapping function between the input variables and a corresponding unknown output quantity. If we train the model with inappropriate data it will lead to some other results, which are not the proper solutions for the equation. While solving the complex PDEs using the neural networks, training the model is the main and important task. And if we are dealing with synthetic data it's going to be more complex and riskier[3]. We need minimum data to train the model, otherwise, it's going to give wrong results for the particular problems.

To tackle these situations, the neural networks are enhancing their models by giving them additional information that is related to physics. These approaches are called PINNs. PINNs are adding information like physics and engineering systems to the model simulating processes by differential equations. PINNs evaluate the solution at different points in the set of data points called collocation points. At collocation points the solution must follow the PDE equation, at the same time it will follow the initial and boundary conditions which are a must to train the model to predict the solution more accurately. PINNs are so efficient at approximating solutions to Inverse and Forward PDE problems.

4.2 Applications of PINNs

PINNs are one of the most popular topics in computational science. It plays a key role, especially in dealing with PDEs. Solving PDEs using PINNs will provide more accuracy.

PINNs offer a very powerful tool for addressing nonlinear systems. The application of PINNs in computational mechanics proves particularly effective in handling problems characterized by nonlinearity, including challenges like large deformation and material nonlinearities. To get partial differential terms no need to use any traditional methods like spatial discretization schemes and approximation methods, by using Automatic Differentiation in PINNs we can get the partial differential terms easily. PINNs are very good at solving inverse problems.[2]

PINNs contribute to the world in different domains, PINNs have been applied to fluid mechanic systems like simulating fluid flow, providing highly accurate solutions to complex problems in aerospace, environmental science, and hydrodynamics. In structural mechanics, PINNs are used to predict material behavior, and structural response, and contribute to civil engineering at a high level. In the healthcare sector, PINNs aid in medical image analysis, patient diagnostics, and drug discovery, offering potential solutions to critical healthcare challenges. PINNs framework is useful for analyzing the nonlinear buckling behavior of a three-dimensional (3D) FG porous, slender beam resting on a Winkler-Pasternak foundation.[5] PINNs play a crucial role in optimizing the design and operation of renewable energy systems and improving the efficiency and sustainability of wind turbines, solar panels, and other green technologies.[2]

PINNs also solved the Swing equation, which has been simplified to an ODE and helped to develop the power system applications.[37] A whole portion of micro electromagnetic problems are solved with a single PINNs model, which learns the solutions of classes of eigenvalue problems, related to the nucleation field associated with defects in magnetic materials.[23] Using PINNs solved the inverse scattering problems in photonic metamaterials and nano-optics, especially using PINNs to retrieve the effective permittivity properties of various finite-size scattering systems. These systems included interactions between multiple nanostructures and nanoparticles with multiple components. [6] [10]

4.3 PINNs Methodology

The proposed PINNs based framework comprises of 3 phases i.e. creating a PINNs model, training the model, and prediction. During the training phase, the hidden layer parameters of the PINNs are optimized using supervised learning. This training builds the relation between the input parameters data and the output data which is the solution of the equation. While we are training the model, we have to calculate the loss function including the PDE residual, and initial and boundary conditions. The loss function will be optimized and provide the best parameters to use the model. Finally, in the prediction phase, the hidden layers remain unchanged to compute the output quantities (e.g.: Option prices) given for the various input parameters of the model. These all steps will take place in the forward pass. In forward pass, we can test our model to evaluate the results by running multiple times.[35]

$$U_t + \eta[u; \lambda] = 0, x \in \Omega, t \in [0, T] \tag{4.1}$$

Where u(t,x) denotes the latent (hidden) solution, $\eta[.; T]$is a non-linear operator parameterized by lambda, Ω is a subset of R. This setup encapsulates a wide range of problems in mathematical physics including conversational laws, and diffusion processes. Advection reaction diffusion-reaction systems and kinetic equations.[35] In the Raissi [35] paper explained the methodology and process for a different example of Continuous-time models and Discrete-time models. Raissi considered some equations as examples and explained the process of particular equations, in this paper he covered Burgers equation, Navier Strokes equation, and Kortewegde Vries Equation.[35]

Neural Networks (NN) in PINNs

In PINNs neural networks are the main step. We have to initialize the neural networks by their architecture, which includes defining the input parameters, number of hidden layers, number of neurons in each layer, initial weights and biases, and output layer.[7]

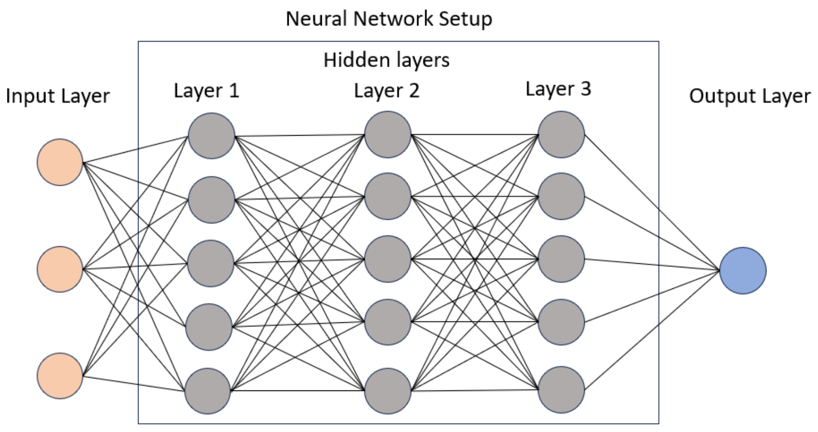


Figure 4.1: Neural Network Setup

In the figure 4.1, We can see the structure of neural networks, which I explained earlier. Neural networks are also known as the Artificial Neural Networks (ANN). Artificial Neural Networks comprise node layers, containing an input layer, one or more hidden layers, and an output layer. Each input layer is associated with the weight and bias. The weighted sum of the particular input will go through the hidden input layer. Hidden layers will give the final output to the output layer through the Activation function. Neural Networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and Artificial Intelligence.[7]

In a neural network, for example, considering just one hidden layer (neuron), the input y_t, a set of inputs given to the model, which can be expressed as a vector of features $y_t = (y_1, ..., y_n)$, is processed by the neuron through weights denoted by $w_i, i = 1, ..., n$. These weights will be distributed to all the inputs, in this step we will allocate the number of weights according to the input values. This processing produces a value z, and the final output c is obtained through some activation function ϕ, i.e., $c = \phi(z)$. [17]

To elaborate further, the value z is the result of processing the n inputs y_i through the corresponding weights w_i.

Mathematically, this is represented as:

$$z = (w_1, ..., w_n) \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} + b = \sum_{i=1}^{n} w_i y_i + b = w^T \cdot y + b \tag{4.2}$$

Here, b is a bias term, and z is essentially the weighted sum of the inputs. In a more compact form, this can be expressed as $z = w^T \cdot y + b$, where w^T denotes the transpose of the weight vector w. The final output c is then obtained by applying the activation function ϕ to z. [17] This is our predicted output from the Neural network.

Each artificial neuron may be regarded as a mathematical function that results in an output c by chaining the activation function (nonlinear) on the inputs $y_i, i = 1, \dots, n$:

$$c = \phi(z = \sum_{i=0}^{n} w_i y_i) = \phi(z = w \cdot y + b) \tag{4.3}$$

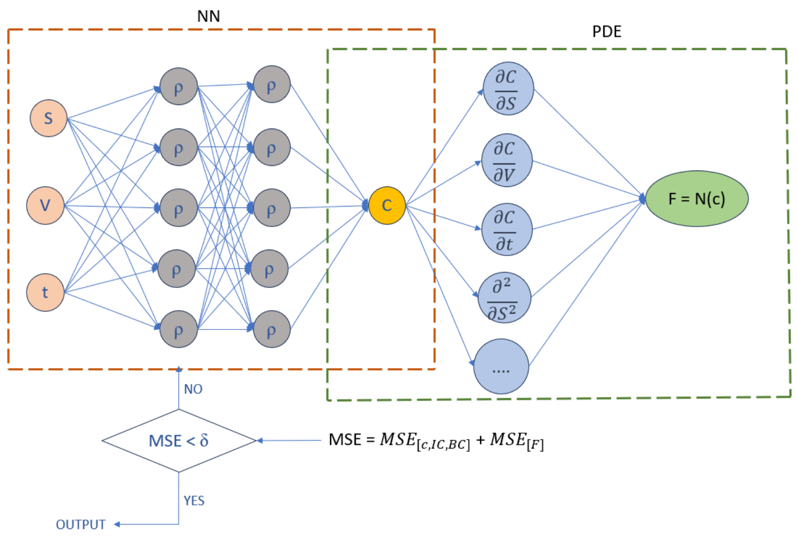

NN
PDE
S
V
t
ρ
C
$\frac{\partial C}{\partial S}$
$\frac{\partial C}{\partial V}$
$\frac{\partial C}{\partial t}$
$\frac{\partial^2}{\partial S^2}$
....
F = N(c)
NO
MSE < δ
MSE = $MSE_{[c,IC,BC]}$ + $MSE_{[F]}$
YES
OUTPUT

Figure 4.2: PINNs Configuration

The provided PINNs setup diagram 4.2 illustrates the architecture of PINNs, The key strength of PINNs lies in their ability to create robust models with reduced data requirements. By combining data-driven learning with physics-guided constraints, PINNs not only optimize model accuracy within the provided dataset but also exhibit a capacity to extrapolate reliably beyond the available data points. This characteristic makes PINNs particularly valuable in scenarios where limited data is available, as they can generate accurate and consistent models even in regions not explicitly covered by the training data.

Feedforward Pass Process in PINNs

One more important step after setting up the neural networks is a feedforward pass. The feedforward pass in a neural network is the process through which input data is processed layer by layer to produce the final output. The process begins with the input layer, where the network receives all the input parameters. Each node of the input represents a feature, and each node is associated with a weight, and these weights represent the strength of the connection. Additionally, it is associated with bias. For each node in the first hidden layer and the next layers, input is a weighted sum of the input parameters multiplying their respective weights and adding the bias. The weighted sum is passed through the activation function. The activated function introduces the non-linearity to the model, it allows the model to learn the complex patterns in the data. The common activation function is in use now like Sigmoid, Hyperbolic Tangent (tanh), and rectified linear unit (Relu).[7]

The above whole process weighted sum, applying an activated function, and propagating the information through each hidden layer is repeated until the data reaches the final output node. Each output of the layer is the input to the next layer. The final layer produces the output. The values given by the output layer are the predicted values for the given input parameters.

A linear function can be thought of as a simple activation function that outputs c $\times$ x for input x with c as a constant. Normally the linear activation function doesn't add non-linearity to the Neural Network setup in PINNs, but the neural networks need to introduce nonlinearity. Otherwise, a neural network produces the output as a linear function of inputs despite having several layers.[11]

Sigmoid Function: To introduce the non-linearity in Neural networks the Logistic Sigmoid activation functions have been used in the early days. In general, a sigmoid function is monotonic and has a first derivative which is bell-shaped. The output of the logistic sigmoid function is saturated for higher and lower inputs, which leads to a vanishing gradient problem.[11]

The sigmoid function is given as:

$$\text{Logistic Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{4.4}$$

Tanh Function: The Tanh function has also been used as the activation function in neural networks. It is similar to the Logistic Sigmoid function while exhibiting the zero-centric property. Generally, the Tanh activation function became preferred over the sigmoid function, because it gives better performance for multi-layer neural networks.[11]

The Tanh activation function is given as :

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.5}$$

Rectified Linear unit Function: There are some key limitations for getting saturated output and an increase in complexity using the activated functions like Sigmoid and Tanh. The Rectified Linear unit (ReLu) activation function has become more popular due to its simplicity and increase in performance. ReLu has been tackling the limitations caused by the other two activation functions.[11]

Rectified Linear Unit(ReLu) activation function is given as :

$$\text{ReLU}(x) = \max(0, x) \tag{4.6}$$

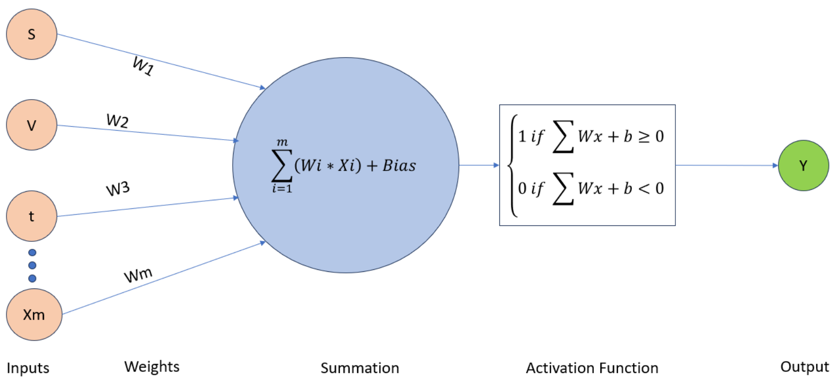


Figure 4.3: Feed Forward Process

Automatic Differentiation in PINNs

In the world of mathematics and computer algebra, automatic differentiation (often referred to as auto-differentiation, autodiff, or AD) is a collection of techniques designed to compute the partial derivative of a function that is defined by a computer program. The fundamental concept behind automatic differentiation lies in exploiting the inherent nature of every computer calculation, regardless of its complexity. Such computations involve a sequence of basic arithmetic operations (like addition, subtraction, multiplication, division) and elementary functions (such as exp, log, sin, cos). By systematically applying the chain rule to these operations, automatic differentiation enables the computation of partial derivatives of any order automatically. This process is not only accurate up to the working precision but also achieves this with, at most, a small constant factor increase in the number of arithmetic operations compared to the original program. [31]

Lets consider a target function $f : \mathbb{R}^n \to \mathbb{R}^m$, the corresponding $m \times n$ Jacobian matrix J is defined by its components:

$$J_{ij} = \frac{\partial f_i}{\partial x_j} \tag{4.7}$$

This matrix contains the partial derivatives of all outputs to their inputs. If f has a one-dimensional output, as in the case when f is an objective function, the Jacobian matrix simplifies to the gradient. In practical scenarios, the focus may be on partial derivatives concerning specific inputs, leading to the computation of a reduced Jacobian matrix, where the respective columns are termed sensitivities.[29]

Now, let's consider f as a composite function: $f(x) = h \cdot g(x) = h(g(x))$, where $x \in \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^k$, and $h : \mathbb{R}^k \to \mathbb{R}^m$. By applying the chain rule and elementary matrix multiplication: [29]

$$J = J_h \cdot g = J_h(g(x)) \cdot J_g(x) \tag{4.8}$$

with the (i, j)-th element defined as:

$$J_{ij} = \frac{\partial f_i}{\partial x_j} = \frac{\partial h_i}{\partial g_1}\frac{\partial g_1}{\partial x_j} + \frac{\partial h_i}{\partial g_2}\frac{\partial g_2}{\partial x_j} + \ldots + \frac{\partial h_i}{\partial g_k}\frac{\partial g_k}{\partial x_j} \tag{4.9}$$

In a more general setting, if our target function f involves a composition of L functions:

$$f = f_L \cdot f_{L-1} \cdot \ldots \cdot f_1 \tag{4.10}$$

then the corresponding Jacobian matrix can be expressed as:

$$J = J_L \cdot J_{L-1} \cdot \ldots \cdot J_1 \tag{4.11}$$

When applying Automatic Differentiation (AD), we express our target function f in a computer program. AD operates in either forward or reverse mode. For a vector $u \in \mathbb{R}^n$, one sweep of forward-mode AD numerically evaluates the action of the Jacobian matrix on u, denoted as $J \cdot u$, according to the following recursive relationship: [29]

$$J \cdot u = J_L \cdot J_{L-1} \cdot \ldots \cdot J_3 \cdot J_2 \cdot J_1 \cdot u = J_L \cdot u_{L-1} \tag{4.12}$$

This process breaks down the action of the Jacobian matrix into simple components, evaluated sequentially.[29]

Loss Computation for PINNs method

Loss computation is the difference between the predicted output of the model to the actual target values. While working on the PDEs the loss will include the PDE residual loss, initial loss, and boundary loss. These losses will be combined and calculated in mean square error (MSE) loss form to reduce the error between the predicted values to the actual target values. Various loss functions exist now, and the use of loss functions depends on the nature of the task, such as the Mean Square Error (MSE) function for regression problems.

The mean square error loss function is mostly used in regression problems to measure the average squared distance between the predicted values and actual values. The input-output set of a particular regression problem is (x_i, y_i) and the model predicts $\hat{y}_i$ as output for input x_i. [35] The Mean Square Error (MSE) formula is given by

$$MSE = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4.13}$$

n	: number of data points in our dataset
Y_i	: actual output for the ith data point
$\hat{Y}_i$	: predicted output for the ith data point

The calculation involves taking the squared difference between each actual and predicted value, summing up these squared differences, and then dividing by the number of data points (n). The squaring ensures that both positive and negative errors contribute to the overall loss. This loss function penalizes larger errors more heavily than smaller errors due to the squaring operation. It is a way of expressing the average magnitude of the errors between predicted and actual values. In practice, minimizing the MSE during training is equivalent to finding model parameters that provide the best fit to the training data in a least squares sense.

In PINNs loss computation will be the sum of the Physical loss of the equation and the data loss of collocation points. So, the Loss function consists of two terms, a data mismatch term, and a Physics-Informed regularization term :

$$\text{LOSS} = L_{\text{Data}} + L_{\text{Physics}} \tag{4.14}$$

Let's consider a Partial Differential Equation (PDE)

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{4.15}$$

U is the solution of the PDE. Let's use the function to calculate the losses in PINNs that we mentioned. Assume that we have discrete data points for x and t, and $u_{\text{NN}}(x_i, t_i)$ represents the neural network prediction at the grid points (x_i, t_i).

$$\text{Data Loss} = \frac{1}{N}\sum_{i=1}^{N} |u_{\text{NN}}(x_i, t_i) - f(x_i, t_i)|^2 \tag{4.16}$$

$$\text{Initial Loss} = \frac{1}{N}\sum_{i=1}^{N} |u_{\text{NN}}(x_i, 0) - f(x_i, 0)|^2 \tag{4.17}$$

$$\text{Boundary Loss} = \frac{1}{M}\sum_{j=1}^{M} |u_{\text{NN}}(0, t_j) - f(0, t_j)|^2 \tag{4.18}$$

$$\text{PDE Residual Loss} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \left|\frac{\partial u_{\text{NN}}}{\partial t}(x_i, t_j) + c\frac{\partial u_{\text{NN}}}{\partial x}(x_i, t_j)\right|^2 \tag{4.19}$$

By considering the above all losses we are going to calculate the Total loss function, which we are implementing in PINNs to reduce the error.

$$\text{Total Loss} = \text{Data Loss} + \text{Initial Loss} + \text{Boundary Loss} + \text{PDE Residual Loss} \tag{4.20}$$

Where L_{Data} measures the mismatch between the neural network predictions and observed data, and L_{Physics} ensures the predictions satisfy the PDE with Initial and boundary conditions. We can observe in the below figure, that after computing the derivatives using Automatic Differentiation, it will concentrate on the Loss computation, as we defined here.

The figure 4.4 shows a clear view of the loss computation process in PINNs, Data Loss coming from the neural networks, and Physics Loss coming from the PDE conditions and residual. The sum of the losses will be considered as the PINN's loss. The loss will follow through all layers and give the final one as the main loss as shown in figure 4.5.

Backward Pass process in PINNs

The algorithm then works backward through the network, starting from the output layer. The goal is to calculate the loss gradient for the weights and biases. Using the

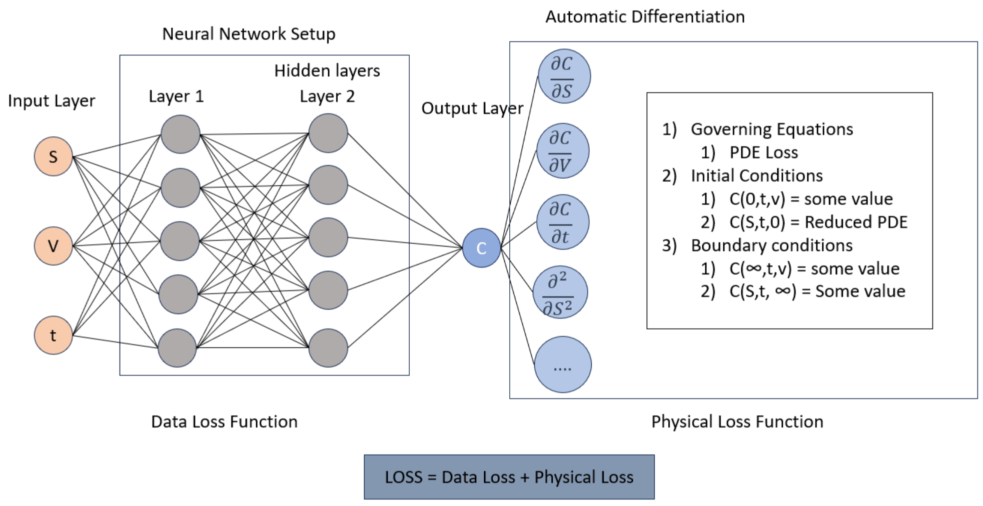


Figure 4.4: Loss Computation Workflow

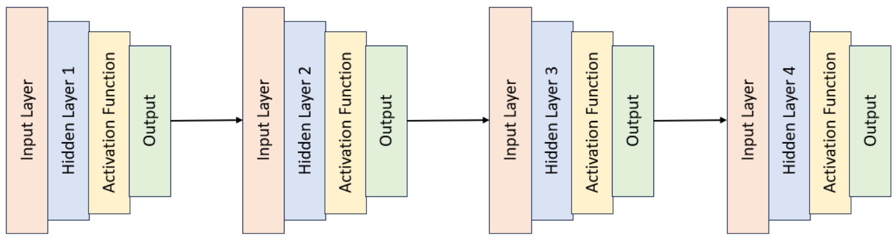


Figure 4.5: Neural Networks Loss

chain rule of calculus, the partial derivatives of the loss for the weights and biases are computed layer by layer. This process involves propagating the error backward through the network. The calculated gradients are used to update the weights and biases in the direction that reduces the loss. This update is typically performed using an optimization algorithm, such as gradient descent, with a learning rate parameter. The above process is iterative for multiple epochs or until convergence. Each iteration refines the network's weights and biases, improving its ability to make accurate predictions on the training data.

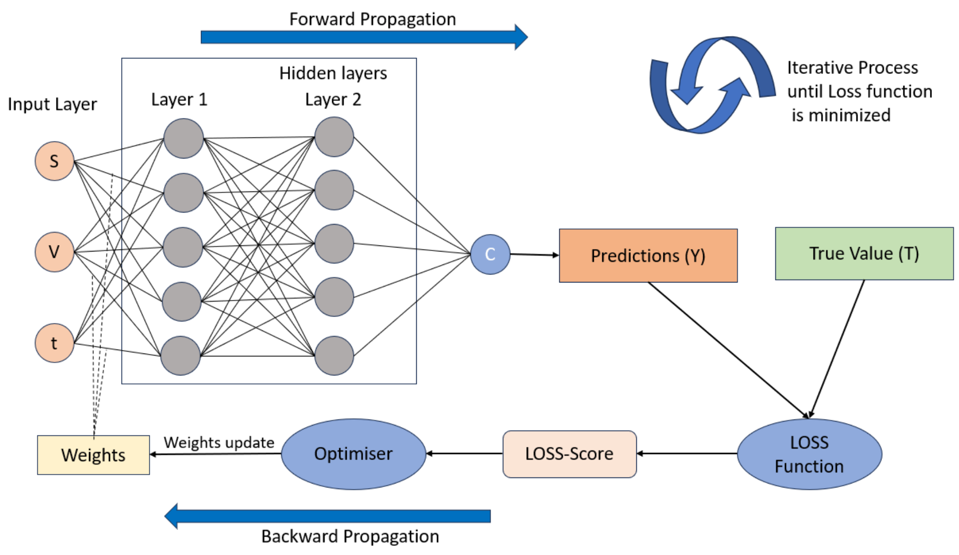


Figure 4.6: Backward Propagation Illustration

Applying Gradient Descent optimization in PINNs

Gradient descent is an optimization algorithm used to find the weights that minimize the cost function. Minimizing the cost function means reaching the minimum point of the cost function. Therefore, gradient descent aims to find a weight corresponding to the cost function's minimum point. The learning rate is a tuning parameter that determines the step size at each iteration of gradient descent. It determines the speed at which we move down the slope. The learning rate controls the step size, preventing the algorithm from overshooting the minimum or converging too slowly.

$$\text{New parameter} = \text{Old parameter - Learning rate * Gradient} \tag{4.21}$$

Let's consider a cost function, we are calculating the error between the True values and the predicted values and squaring the error so that the difference should not be negative.

$$MSE = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.22}$$

Here y_i represents the actual y values, and $\hat{y}_i$ represents the predicted value from the neural networks. The predicted value is calculated by using the formula $y = mx + b$ where m is the slope of that line and b is the intercept. So now the above function becomes

$$MSE = \frac{1}{2}\sum_{i=1}^{N}(y_i - (mx_i + b))^2 \tag{4.23}$$

As we know the gradient descent algorithm is an iterative process, it will take random values for m and b, until it finds the local minima or global minima. First, it will consider m = 0 and b = 0, then the error will be too high, in the second approach it will consider other values for m and b, and the error function will give fewer values compared to the first. Like this, it will iterate till it finds the local minima or global minima. Then the values of m and b which give the global minima values, will be used in the formula to predict the values (y = mx + b).

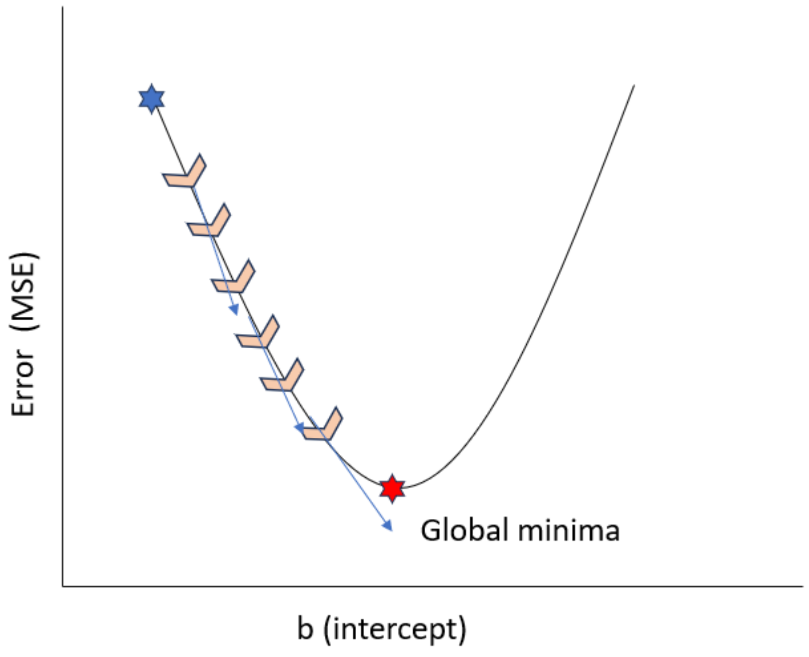


Figure 4.7: Gradient Descent Visualization

From figure 4.7 we can observe that, the local minima at the red point, the initial error value is very high at the blue point, by using the step sizes it will try to reach from the blue point to the redpoint, by checking different values. the step size keeps reducing each time we take a step and thus finally the gradient descent converges to the local/global minima.

For this, we have to calculate the tangent values at each point so that we will get to know which direction we have to go. Since in the example we are considering two values m and b, so we have to calculate two partial derivatives of the error function 4.23.

$$\frac{\partial}{\partial m}(MSE) = \frac{2}{n}\sum_{i=1}^{N} -x_i(y_i - (mx_i + b)) \tag{4.24}$$

$$\frac{\partial}{\partial b}(MSE) = \frac{2}{n}\sum_{i=1}^{N} -(y_i - (mx_i + b)) \tag{4.25}$$

As discussed above now the learning rate will come into play. The learning parameter is hyper parameter which will determine the step size in the process. The optimum value for the learning rate will lead to converge faster. if the learning rate is too high or too low, the model will find difficulties in getting global minima. So it is necessary to find the optimal value of the learning rate. As discussed if we calculate the derivative and find the optimal learning rate, then we can calculate our new parameters for m and b.

$$m = m - \text{Learning rate} * \frac{\partial}{\partial m}(MSE) = m - \alpha * \frac{\partial}{\partial m}(MSE) \tag{4.26}$$

$$b = b - \text{Learning rate} * \frac{\partial}{\partial b}(MSE) = b - \alpha * \frac{\partial}{\partial b}(MSE) \tag{4.27}$$

Now these new values are used to calculate the new error loss, these steps will iterate until the m and b do not significantly reduce the error function.

Iterative Process for PINNs

If we repeat the above steps for a predefined number of iterations (epochs) or until the convergence, the loss becomes sufficiently small. Each iteration refines the model parameters, progressively reducing the loss and improving the model's performance. The iterative nature allows the model to progressively refine its parameters, moving towards values that optimize its predictive performance. After iterative for a certain number of times, we can predict the values using the test data.

Chapter 5

Heston model pricing PDE with PINNs

In this chapter, we are discussing the solving process of the Heston model PDE using Physics Informed Neural Networks (PINNs). The process of solving the Heston model PDE to calculate European call option price and the process methodology will be discussed here.

5.1 Overview of the Heston Pricing Model

As we have already seen the Heston model PDE in Chapter 3. This PDE is widely used in financial mathematics for valuing options prices and other derivatives. The Heston model and its associated PDE find various applications in finance and risk management.

The Heston model is defined as :

Stochastic price (Under risk-neutral measure):

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^s \tag{5.1}$$

Volatility (Under risk-neutral measure):

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t} dW_t^v \tag{5.2}$$

Correlation:

$$dW_t^s dW_t^v = \rho dt \tag{5.3}$$

The Heston model PDE is :

$$\frac{\partial C}{\partial t} + rS_t\frac{\partial C}{\partial S_t} + k_v(\mu - V_t)\frac{\partial C}{\partial V_t} + \frac{1}{2}\left(V_t S_t^2\right)\frac{\partial^2 C}{\partial S_t^2} + \frac{1}{2}\left(\sigma^2 V_t\right)\frac{\partial^2 C}{\partial V_t^2} + \left(\sigma V_t S_t \rho_t\right)\frac{\partial^2 C}{\partial S_t \partial V_t} - rC = 0 \tag{5.4}$$

The above Heston PDE consists of 3 components namely Diffusion terms, Convection terms, and Interest rate terms. Diffusion terms capture the diffusion of the option price to the underlying asset price S_t and stochastic volatility V_t. Incorporate geometric Brownian motion and correlation between asset price and volatility. When it comes to the Convection terms, reflect the convection terms related to the drift of the option price for stochastic volatility, and it represents the mean-reverting behavior of the volatility. Finally, the Interest rate term accounts for the impact of the risk-free interest rate (r).

5.2 PINNs to solve Heston model PDE

To solve the Heston model PDE there are several numerical methods, the Finite Elements method (FEM), Finite Volume method (FVM), ADI schemes, and Numerical integration methods. Utilizing numerical methods for solving PDEs offers effective solutions to the equations. In the current landscape, Machine learning and Deep learning techniques simplify the process, providing accessible and quick implementation options. When it comes to Machine learning techniques neural networks are the best fit to solve the PDEs. Solving PDEs with neural networks has been previously considered in some papers such as [26][33]. However, these papers provide the neural network solution with a pre-fixed mesh and only one hidden layer in the model. In recent years there are several new techniques are emerged from Machine and Deep learning to solve high-dimensional PDEs. Here we are concentrating more on a new Deep learning technique called the PINNs method by Maizar Raissi [36] recently developed by George Karniadarkis and his collaborators at Brown University, since it is quite powerful and efficient in solving various PDEs such as Navier-Stokes equations, integrodifferential equations, fractional differential equations, and stochastic equations. [43]

5.3 Formation of the Problem

In this thesis, we are using a different approach a deep learning method called PINNs implemented by Maizer Raissi [34] to calculate the European call option price of the Heston model PDE. The PINNs method employed the neural networks with the law of physics to solve the PDEs and nonlinear PDEs. Using the advances in the present financial world Raissi employed the automatic differentiation [4] approach to the PINNs to simplify the process of derivative terms. PINNs are one of the best deep learning methods and underused methods to solve PDEs and nonlinear PDEs. We already implemented the PDE for the Heston model in Chapter 3, and we are going to extend the Heston model by incorporating the stochastic correlation between the underlying asset to the volatility process. We are solving the Heston model PDE using PINNs and then we are implementing the PINNs model to solve extended Heston model PDE. From the predicted PINNs option prices, we are validating whether are PINNs the best method to solve the Heston model for the option prices or not.

The general aim of this work is to solve the Heston model PDE using PINNs to predict European call option prices and to prove that PINNs are one of the best methods to solve the Heston model PDEs with proper initial and boundary conditions. Now let us take the Heston model PDE that we derived earlier 5.4.

Parameters to solve Heston model PDE

We are considering the parameters to solve the Heston model as Strike Price (K) is 40, Interest rate (r) is 0.05, Annual Volatility (σ) is 0.25, Mean reversion speed (k_v) is 0.1, Long term average (μ_v) is 0.01 and Correlation (ρ) is -0.167.

For the training data, we considered stocks ranging from 0 to 500, volatility ranging from 0 to 3, and time ranging from 0 to 1 (in years). This broad range will enable the model to capture diverse market conditions and dynamics during the training process. For this experiment, we are considering the above parameters, and we are going to work on the experiment with another parameter dataset. Training the PINNs model is one of the important steps to get accurate results, when we are working on synthetic data choosing the best parameters will lead to better results.

Initial and Boundary Conditions

A European call option with a strike price of K, and maturity time of T, is subjected to the following conditions. While solving the Heston model PDE, we are considering these initial and boundary conditions. Choosing the Initial and boundary conditions is very important to solve the PDE correctly. otherwise, it will lead to inaccurate results.

$$C(S, V, T) = \max(S - K, 0)$$

$$C(0, V, t) = 0$$

$$\frac{\partial C}{\partial t}(\infty, V, t) = 1$$

$$rS\frac{\partial C}{\partial S}(S,0,t) + k_v\mu\frac{\partial C}{\partial V}(S,0,t) - rC(S,0,t) + \frac{\partial C}{\partial t}(S,0,t) = 0$$

$$C(S,\infty,t) = S$$

We are incorporating the above-mentioned initial and boundary conditions in PINNs to the particular PDE at different collocation points. we are considering two initial conditions at $S = 0$ and $V = 0$, as an initial condition we are implementing and comparing it with the PINNs predicted solution. In the same way, we are considering boundary conditions at $S = \infty$ and at $V = \infty$. Implementing the above-mentioned conditions at particular boundaries and comparing them with the PINNs predicted solution to calculate the loss. We are calculating the PDE residual loss at collocation points and loss at initial and boundary points for the Heston PDE.

PINNs Objective Function for Heston model

The PINNs main aim is to solve the PDEs and while training the process of Neural Networks in PINNs model incorporate the physical principles. The PINNs objective function mainly consists of two parts: data-driven loss and physics-informed loss. Generally, PINNs objective function is often formulated as the sum of these two losses.

The objective function of PINNs is :

$$\text{Total Loss} = \text{Data Loss} + \text{Physics Loss} \tag{5.5}$$

Data Loss: The data-driven loss in the PINNs ensures that Neural Networks accurately represent the observed data. Simply it measures the mismatch between the Neural network predictions and observed data. It is often defined using a standard loss function such as mean squared error (MSE) or any other proper metric. [1]

$$\text{Data Loss} = \text{MSE}(\text{Predicted Data}, \text{Actual Observed Data}) \tag{5.6}$$

Here we calculated the Data loss based on all collocation points and calculated the loss in the Mean square error loss function. In the given form N represents the number of points considered for data points.

$$\text{Data Loss} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{PINN}}(s_i, t_i, v_i) - C(s_i, t_i, v_i)|^2 \tag{5.7}$$

Physical Loss: Physical loss in the PINNs ensures the Neural network prediction satisfaction of the governing PDE. It is also often defined using a standard loss function such as mean squared error (MSE) or any other proper metric. [1]

$$\text{Physical Loss} = \text{MSE}(\text{PDE Residuals}, 0) \tag{5.8}$$

When it comes to the Physics loss, we have to consider the PDE Residual loss and initial and boundary losses.

$$\begin{aligned}\text{PDE Residual Loss} = \Bigg(& \frac{\partial C}{\partial t} + rS_t \frac{\partial C}{\partial S_t} + k_v(\mu - V_t)\frac{\partial C}{\partial V_t} \\ & + \frac{1}{2}\left(V_t S_t^2\right)\frac{\partial^2 C}{\partial S_t^2} + \frac{1}{2}\left(\sigma^2 V_t\right)\frac{\partial^2 C}{\partial V_t^2} + (\sigma V_t S_t \rho_t)\frac{\partial^2 C}{\partial S_t \partial V_t} - rC\Bigg)^2\end{aligned} \tag{5.9}$$

$$\text{Initial Stock loss} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{PINN}}(0, t_i, v_i) - C(0, t_i, v_i)|^2 \tag{5.10}$$

$$\text{Initial Volatility loss} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{PINN}}(s_i, 0, v_i) - C(s_i, 0, v_i)|^2 \tag{5.11}$$

$$\text{Boundary Stock loss} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{PINN}}(\infty, t_i, v_i) - C(\infty, t_i, v_i)|^2 \tag{5.12}$$

$$\text{Boundary Volatility loss} = \frac{1}{N}\sum_{i=1}^{N} |C_{\text{PINN}}(s_i, \infty, v_i) - C(s_i, \infty, v_i)|^2 \tag{5.13}$$

Now, When it comes to our Heston PDE, the Loss is going to be the same as the above, but here we are implementing initial conditions for S and V, and boundary conditions for S and V. In the loss functions $C_{\text{PINN}}(s_i, t_i, v_i)$ gives the predicted Call value by PINNs, and $C(s_i, t_i, v_i)$ gives the observed Call value. Using the sum of all losses as mentioned in the equation will help to decrease the error.

The total loss function for our Heston model PDE is :

$$\begin{aligned}\text{Total Loss} = {} & \text{Data Loss} + \text{PDE Residual Loss} \\ & + \text{Initial Stock Loss} + \text{Initial Volatility Loss} \\ & + \text{Boundary Stock Loss} + \text{Boundary Volatility Loss}\end{aligned} \tag{5.14}$$

The above-mentioned losses are considered based on the Heston model PDE initial and boundary conditions. The Data Loss is formulated as the Mean Square Error between observed and predicted data by the neural network. The PDE Residual loss corresponds to the residual loss of the main PDE in the Heston model.

5.4 Implementation

The implementation process of solving the Heston PDE using the above conditions in PINNs. We are going to implement all the conditions on the Heston model PDE, to train our PINNs model. As explained in the process of PINNs in the previous chapter, we are going to create our PINNs setup for the Heston model. After creating the neural network setup, we have to prepare the data to train the model. We are not working on any real data, so we have to prepare our data very carefully. Considering the bad data will lead to a failure model. Once we prepare our data, we have to train the model by giving data at different points like collocation points, initial points, and boundary points. As we discussed in the previous section we already decided our initial and boundary conditions for the Heston model. So, we are implementing these conditions in the PINNs model to train it perfectly.

Data Preparation

While solving the Heston model PDE we worked on synthetic data, we considered stock prices ranging from 0 to 500, volatility range from 0 to 3, and time ranging from 0 to 1 (in years). We separated this data into certain points like collocation points, initial points, and boundary points. All these points are divided to train the model according to the conditions. we considered 1000 collocation points for stocks, volatility, and time. same way considered 200 points as initial points and 200 points as boundary points. In the below figure, we can see, how we distributed the collocation points, initial and boundary points of stocks, and volatility.

Physics Informed Neural Networks (PINNs) Training

We have established the neural network architecture for PINNs, considering the Heston model PDE. The neural network takes stocks, volatility, and time as inputs, with weights and biases considered accordingly. With 5 hidden layers, each consisting of 128 neurons, the network processes these inputs, generating a single output representing the European option call price. This output aligns with the physics laws defined in the PINNs framework, employing Automatic Differentiation (AD) [4] to compute the required derivatives of the option price with respect to the asset price, volatility, and time. The subsequent step involves calculating the PDE residual of the Heston model PDE.

In figure 5.2, we can observe the different losses we are calculating to solve our Heston model PDE. These losses are separated as we mentioned data loss and

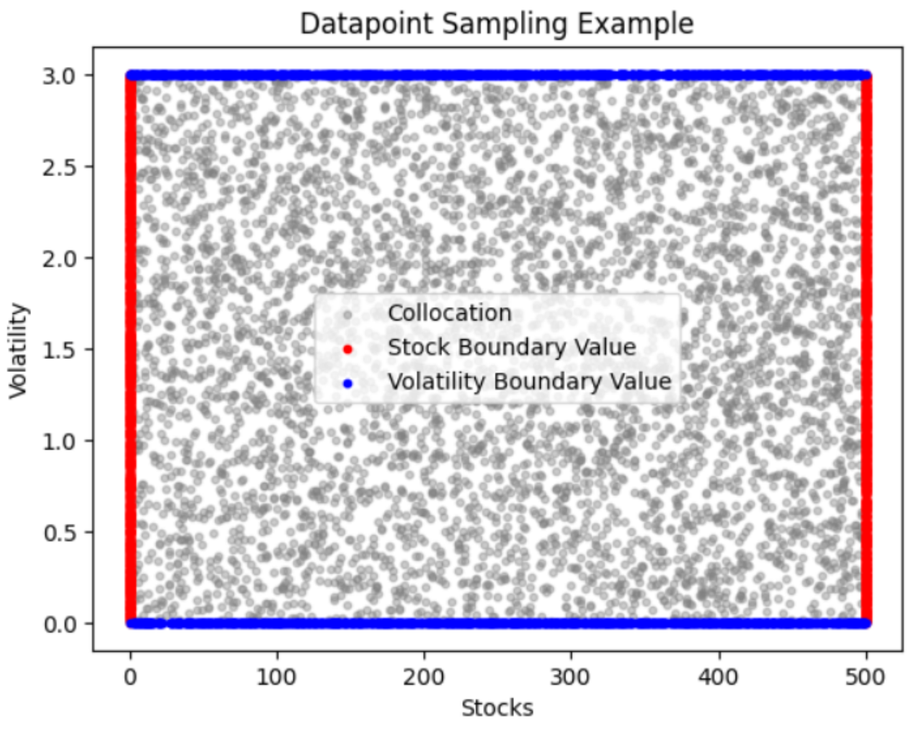


Figure 5.1: Data Sampling

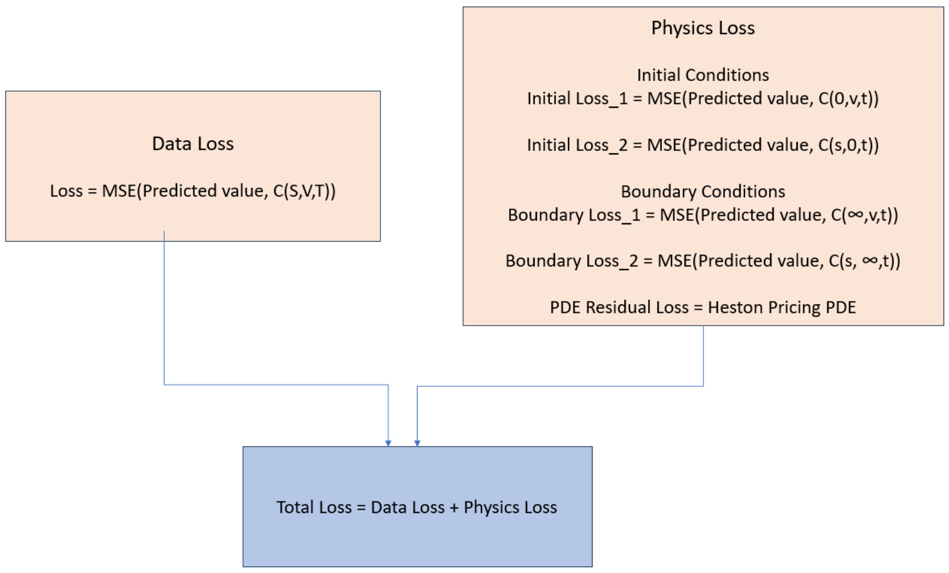

Data Loss
Loss = MSE(Predicted value, C(S,V,T))
Physics Loss
Initial Conditions
Initial Loss_1 = MSE(Predicted value, C(0,v,t))
Initial Loss_2 = MSE(Predicted value, C(s,0,t))
Boundary Conditions
Boundary Loss_1 = MSE(Predicted value, C(∞,v,t))
Boundary Loss_2 = MSE(Predicted value, C(s, ∞,t))
PDE Residual Loss = Heston Pricing PDE
Total Loss = Data Loss + Physics Loss

Figure 5.2: Formula for Total Loss Calculation

physics loss. The data loss left side of the figure is just going to calculate the loss at all collocation points according to the observed data to the predicted data. When we are looking at the right side, we can observe the physics loss which is related to the PDE. In physics loss, we are summing up different losses like PDE residual, Initial losses, and boundary losses. The sum of the individual Mean Square Error losses is considered as the total loss of the model. The main aim of the Loss function is to decrease the total loss of the model, so it will backpropagate and the model will adjust the weights and biases for the model again the process will repeat to decrease the loss function. we can observe the loss function in the figure 5.3.

In this study, we focused on a scenario where the total number of training data (N) is relatively limited, ranging from a few hundred to a few thousand points. Specifically, for our experiment, we utilized 1000 collocation points to train the model. To optimize all previously defined loss functions, we employed L-BFGS, a quasi-Newton, full-batch gradient-based optimization algorithm [27]. The model undergoes training based on the total loss, and through the backpropagation process, adjustments are made to the input, weights, and biases according to the loss. The training continues until the model converges. Our approach demonstrates the capability of achieving high prediction accuracy with a sufficiently expressive neural network architecture, even with a relatively small number of collocation points (N). The training process of the PINNs is illustrated in figure 5.3.

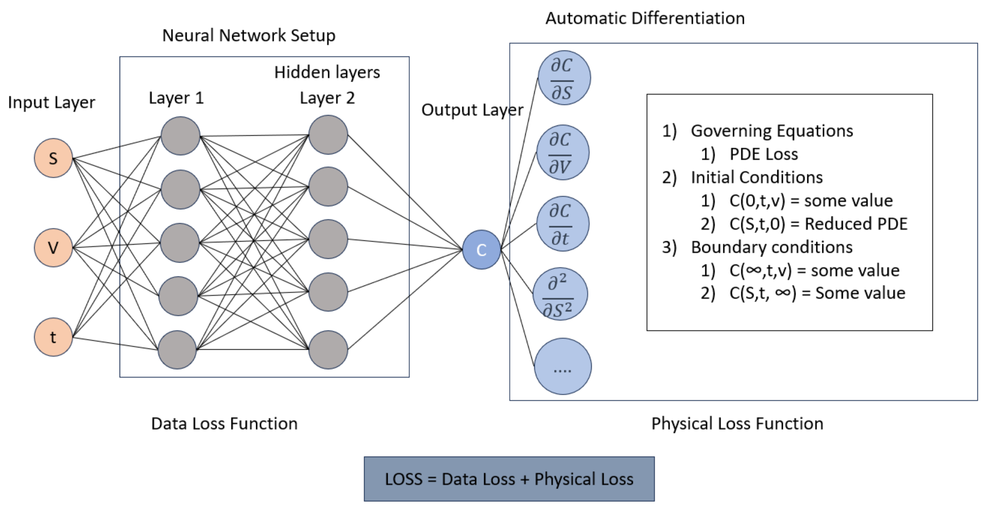


Figure 5.3: Sample process in PINNs Implementation

Prediction We successfully trained our PINNs model for the Heston model PDE. Now we are going to test our model with different data. Now we can consider the three inputs as trained data like stocks ranging from 0 to 500, volatility ranging from 0 to 3, and Time to maturity from 0 to 1. By using any random values from these three inputs now we can predict the European call option price for the Heston model.

5.5 Physics Informed Neural Networks (PINNs) Code Structure

1. Initialize Neural Network for PINNs model

```
class PINN(nn.Module):
def __init__(self, input_dim, hidden_dim, output_dim=1):
    super(PINN, self).__init__()
    self.input_dim = input_dim
    self.hidden_dim = hidden_dim
    self.output_dim = output_dim

    self.hidden_layer = nn.Linear(self.input_dim, self.hidden_dim)
    self.activation = torch.relu
    torch.nn.init.xavier_normal_(self.hidden_layer.weight.data)
def forward(self, x):
    inputs = x
    layer_out = self.activation(self.hidden_layer(inputs))
    out = self.output_layer(layer5_out)
    return out
```

2. Specify the Heston model PDE

```
def pde_loss:
    dC_dt + (r*S1)*dC_dS + (k_v*(mu_v - V1))*dC_dV  +
    (0.5 * (V1 * (S1**2))) * d2C_dS2 +
    (0.5 * (Sigma_v ** 2) * V1)* d2C_dV2 +
    (Sigma_v * S1 * V1* Rho) * d2C_dSdV - r*C
```

3. Define the loss functions (Data Loss, PDE Residual Loss, Initial/Boundary Losses)

```
def Total Loss:
    Data loss + pde_loss + Init_loss_1 + Init_loss_2 +
    Bound_loss_1 + Bound_loss_2
```

4. Specify the optimization algorithm (e.g., L-BFGS)

```
optimizer = optim.Adam(model.parameters(), lr=5e-5)
```

5. Generate collocation points for training data

```
def get_diff_data(n):
X = np.concatenate([np.random.uniform(*t_range, (n, 1)),
                    np.random.uniform(*S_range, (n, 1)),
                    np.random.uniform(*V_range, (n, 1))], axis=1)
y = np.zeros((n, 1))
return X, y
```

6. Training loop:

   a. Forward pass: Evaluate neural network predictions
   b. Compute loss: Calculate the total loss using defined loss functions
   c. Backpropagation: Update weights and biases based on the loss
   d. Optimize: Use the chosen optimization algorithm to minimize the loss
   e. Check convergence: Repeat until the model converges or a predefined criterion is met

```
model = PINN(3, 128, 1)
n_epochs = 13000
optimizer = optim.Adam(model.parameters(), lr=5e-5)
loss_hist = []
for epoch in range(n_epochs):
    optimizer.zero_grad()
    Total loss.backward()
    optimizer.step()

    loss_hist.append(loss.item())
```

7. Post-training:

   a. Evaluate the trained model on test data
   b. Visualize results

```
input_values = torch.tensor([[x,x,x]])
predicted_value = model(input_values.float())
```

8. Comparision:

    a. Compare with true values

    b. Visualize the comparison results

Chapter 6

Extension of Heston model with Stochastic Correlation

In this chapter, we are extending the Heston model by incorporating a stochastic correlation process. Creating a proper Stochastic Differential Equation (SDE) system, and generating a pricing PDE using the Feynman kac theorem.

6.1 Derivation of Extended Heston model PDE

We extend the Heston model by imposing the stochastic correlation between asset price and volatility. we will consider two correlations, one between the asset price and volatility and another one between asset price and asset price to volatility.[24]

The model is given by

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^s \tag{6.1}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t} dW_t^v \tag{6.2}$$

$$d\rho_t = a(t, \rho_t)dt + b(t, \rho_t)dW_t^\rho \tag{6.3}$$

where :

$$dW_t^s\, dW_t^v = \rho_t\, dt \quad \text{[Spot and Volatility]}$$

$$dW_t^s\, dW_t^\rho = \rho_{cs}\, dt \quad \text{[Spot \& Spot/Volatility]}$$

$$dW_t^v\, dW_t^\rho = \rho_t\, \rho_{cs}\, dt \quad \text{[Variance \& Spot/Volatility]}$$

To check conveniently the affinity, we reformulate the SDE system with respect to the independent Brownian Motions: We first rearrange the SDE system

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^s \tag{6.4}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t}dW_t^v \tag{6.5}$$

$$d\rho_t = a(t,\rho_t)dt + b(t,\rho_t)dW_t^\rho \tag{6.6}$$

Which has a family of correlation matrices

$$C_t = \begin{pmatrix} 1 & \rho_t & \rho_{cs} \\ \rho_t & 1 & \rho_t\rho_{cs} \\ \rho_{cs} & \rho_t\rho_{cs} & 1 \end{pmatrix}$$

Performing a Cholesky-decomposition $C_t = LL^T$, It allows us to express the Correlation matrix C_t as the product of a lower triangular matrix L and its Transpose

$$L[1,1] = \sqrt{C_t[1]} = \sqrt{1} = 1$$

$$L[2,1] = \frac{C_t[2,1]}{L[1,1]} = \frac{\rho_t}{1} = \rho_t$$
$$L[2,2] = \sqrt{C_t[2,2] - L[2,1]^2} = \sqrt{1-\rho_t^2}$$

$$L[3,1] = \frac{C_t[3,1]}{L[1,1]} = \frac{\rho_{cs}}{1} = \rho_{cs}$$
$$L[3,2] = \frac{C_t[3,2] - L[3,1]\cdot L[2,1]}{L[2,2]} = \frac{\rho_t\cdot\rho_{cs} - \rho_t\cdot\rho_{cs}}{\sqrt{1-\rho_t^2}} = 0$$
$$L[3,3] = \sqrt{c_t[3,3] - L[3,1]^2 - L[3,2]^2} = \sqrt{1-\rho_{cs}^2}$$

Where L is the Lower Triangular Matrix given by

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \rho_t & \sqrt{1-\rho_t^2} & 0 \\ \rho_{cs} & 0 & \sqrt{1-\rho_{cs}^2} \end{pmatrix}$$

Since our main aim is to impose stochastic correlations to the asset process and stochastic volatility process, Now the reformulated SDE system becomes

$$dS_t = rS_tdt + \sqrt{V_t}S_tdW_t^s \tag{6.7}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t}\rho_t dW_t^s + \sigma\sqrt{V_t}\sqrt{1-\rho_t^2}dW_t^v \tag{6.8}$$

$$d\rho_t = a(t,\rho_t)dt + b(t,\rho_t)\rho_{cs}dW_t^s + b(t,\rho_t)\sqrt{1-\rho_{cs}^2}dW_t^\rho \tag{6.9}$$

We first impose the stochastic correlation driven by the bounded Jacobi process [25] between the asset price and volatility

$$d\rho_t = k_\rho(\mu_\rho - \rho_t)dt + \sigma_\rho\sqrt{1-\rho_t^2}dW_\rho^t \tag{6.10}$$

Rewrite the reformulated system SDE

$$dS_t = rS_t dt + \sqrt{V_t}S_t dW_t^s \tag{6.11}$$

$$dV_t = \kappa_v(\mu - V_t)dt + \sigma\sqrt{V_t}\rho_t dW_t^s + \sigma\sqrt{V_t}\sqrt{1-\rho_t^2}dW_t^v \tag{6.12}$$

$$d\rho_t = k_\rho(\mu_\rho - \rho_t)dt + \sigma_\rho\rho_{cs}\sqrt{1-\rho_t^2}dW_t^s + \sigma_\rho\sqrt{1-\rho_t^2}\sqrt{1-\rho_{cs}^2}dW_t^\rho \tag{6.13}$$

Let's denote the value of the call option with a payoff function at a maturity T

$$f(S_t) = max(S_T - K, 0) \tag{6.14}$$

To derive the PDE we are using the Feynman-Kac theorem [38], which follows the given formula

$$\frac{\partial C}{\partial t} + AC(t,S,V) - rC(t,S,V) = 0 \tag{6.15}$$

A is the differential operator that involves dynamics of S, V, and ρ and r is the risk-free interest rate.

Using the Feynman-Kac theorem, the option price $C(t,S,V,\rho)$ can be expressed as an expectation under the risk-neutral measure

$$C(t,S,V) = e^{-r(T-t)}\mathbb{E}\left[f(C_T)\middle|S(t)=S, V(t)=V, \rho(t)=\rho\right] = e^{-r(T-t)}\mathbb{E}\left[f(C_T)\right] \tag{6.16}$$

Here A is the differential operator that involves S, V, and ρ dynamics. Apply ito Lemma to the option value C(t, S, V, ρ) to derive its dynamics. The generator of the process is

$$A = \sum_{i=1}^{n}\mu_i\frac{\partial}{\partial x_i} + 0.5\sum_{i=1}^{n}\sum_{j=1}^{n}(\sigma\sigma^{\mathrm{T}})_{ij}\frac{\partial^2}{\partial x_i\partial x_j} \tag{6.17}$$

now we are going to write our SDE system in matrix form

$$\begin{bmatrix} dS_t \\ dV_t \\ d\rho_t \end{bmatrix} = \begin{bmatrix} rS_t \\ \kappa_v(\mu - V_t) \\ k_\rho(\mu_\rho - \rho_t) \end{bmatrix} dt \times \begin{bmatrix} \sqrt{V_t}S_t & 0 & 0 \\ \sigma\rho_t\sqrt{V_t} & \sigma\sqrt{V_t}\sqrt{1-(\rho_t)^2} & 0 \\ \sigma_\rho\rho_{cs}\sqrt{1-\rho_t^2} & 0 & \sigma_\rho\sqrt{1-\rho_t^2}\sqrt{1-(\rho_{cs})^2} \end{bmatrix} \begin{bmatrix} dW_t^s \\ dW_t^v \\ dW_t^\rho \end{bmatrix} \tag{6.18}$$

Using the above matrix form now we are going to generate the process of A, now $\sigma\sigma^T$ becomes

$$\begin{bmatrix} \sqrt{V_t}S_t & 0 & 0 \\ \sigma\rho_t\sqrt{V_t} & \sigma\sqrt{V_t}\sqrt{1-(\rho_t)^2} & 0 \\ \sigma_\rho\rho_{cs}\sqrt{1-\rho_t^2} & 0 & \sigma_\rho\sqrt{1-\rho_t^2}\sqrt{1-(\rho_{cs})^2} \end{bmatrix} \times$$
$$\begin{bmatrix} \sqrt{V_t}S_t & \sigma\rho_t\sqrt{V_t} & \sigma_\rho\rho_{cs}\sqrt{1-\rho_t^2} \\ 0 & \sigma\sqrt{V_t}\sqrt{1-(\rho_t)^2} & 0 \\ 0 & 0 & \sigma_\rho\sqrt{1-\rho_t^2}\sqrt{1-(\rho_{cs})^2} \end{bmatrix} \tag{6.19}$$
$$= \begin{bmatrix} V_t(S_t)^2 & \sigma\rho_t V_t S_t & \sigma_\rho\rho_{cs}S_t\sqrt{V_t} \\ \sigma\rho_t V_t S_t & (\sigma^2 V_t) & \sigma_\rho\sigma\rho_t(\rho_{cs})^2\sqrt{V_t} \\ \sigma_\rho\rho_{cs}\sqrt{V_t}S_t\sqrt{1-\rho_t^2} & \sigma_\rho\sigma\rho_t(\rho_{cs})^2\sqrt{V_t}\sqrt{1-\rho_t^2} & (\sigma_\rho)^2(1-\rho_t^2) \end{bmatrix}$$

substitute the values into the equation(6.17)

$$\begin{aligned} A &= rS_t\frac{\partial}{\partial S_t} + \left(k_v(\mu - V_t)\frac{\partial}{\partial V_t} + k_\rho(\mu_\rho - \rho_t)\frac{\partial}{\partial \rho_t}\right) \\ &+ \frac{1}{2}\left(V_t S_t^2\frac{\partial^2}{\partial S_t^2} + \sigma^2 V_t\frac{\partial^2}{\partial V_t^2} + {\sigma_\rho}^2(1-\rho_t^2)\frac{\partial^2}{\partial \rho_t^2} + 2\sigma V_t S_t\rho_t\frac{\partial^2}{\partial S_t\partial V_t}\right. \\ &+ \left. 2\sigma_\rho\rho_{cs}S_t\sqrt{V_t}\sqrt{1-\rho_t^2}\frac{\partial^2 C}{\partial S_t\partial\rho_t} + 2\sigma_\rho\rho_{cs}\rho_t\sigma_v\sqrt{V_t}\sqrt{1-\rho_t^2}\frac{\partial^2}{\partial V_t\partial\rho_t}\right) = 0 \end{aligned} \tag{6.20}$$

Now the PDE equation for (6.15) C = C(t, S, V, ρ)) becomes

$$\begin{aligned} &\frac{\partial C}{\partial t} + rS_t\frac{\partial C}{\partial S_t} + k_v(\mu - V_t)\frac{\partial C}{\partial V_t} + k_\rho(\mu_\rho - \rho_t)\frac{\partial C}{\partial \rho_t} + \frac{1}{2}\left(V_t{S_t}^2\right)\frac{\partial^2 C}{\partial S_t^2} + \frac{1}{2}\left(\sigma^2 V_t\right)\frac{\partial^2 C}{\partial V_t^2} \\ &+ \frac{1}{2}\left({\sigma_\rho}^2(1-\rho_t^2)\right)\frac{\partial^2 C}{\partial \rho_t^2} + (\sigma V_t S_t\rho_t)\frac{\partial^2 C}{\partial S_t\partial V_t} + \left(\sigma_\rho\rho_{cs}S_t\sqrt{V_t}\sqrt{1-\rho_t^2}dt\right)\frac{\partial^2 C}{\partial S_t\partial\rho_t} \\ &+ \left(\sigma_\rho\rho_{cs}\rho_t\sigma_v\sqrt{V_t}\sqrt{1-\rho_t^2}dt\right)\frac{\partial^2 C}{\partial V_t\partial\rho_t} - rC = 0 \end{aligned} \tag{6.21}$$

Where C is the option price, t is time, r is a risk-free rate, S_t is price of the underlying asset at time t, V_t is variance at time t, k_v is variance mean-reversion speed, μ is long-term variance, k_ρ is correlation mean-reversion speed, μ_ρ is a long-term correlation, ρ_t is an instantaneous correlation, $sigma$ is volatility of the asset price process, σ_ρ is the volatility of the correlation process, σ_v is the volatility of the variance process, and ρ_{cs} is the correlation between the asset price and the correlation process.

The extended Heston PDE is high high-dimensional PDE, and by using numerical methods it is difficult to solve. So, we are going the solve the above high dimensional

PDE using the scientific Machine learning method called PINNs. PINNs are an integration of neural networks and physics that can help to learn the model for PDEs and to predict accurate results.

6.2 Extended Heston model PDE with PINNs

The process to solve the extended Heston model PDE is going to be the same as the Heston model. But, in the extended Heston model we are using 4 input parameters, 5 hidden layers, 128 neurons per hidden layer and one output is European call option price. The implementation process will be the same as the normal Heston model.

After defining the Neural Networks, we have to define the Initial and Boundary conditions for the extended Heston model to train the model perfectly. To predict accurate results initial and boundary conditions are essential. Now we will define the conditions for the extended Heston model.

$$C(S,V,T,\rho) = \max(S-K,0) \tag{6.22}$$

$$C(0,v,t,\rho) = 0 \tag{6.23}$$

$$\frac{\partial C}{\partial t}(\infty,v,t,\rho) = 1 \tag{6.24}$$

$$\begin{gathered}\frac{\partial C}{\partial t}(s,0,t,\rho) + rS_t\frac{\partial C}{\partial S_t}(s,0,t,\rho) + k_v\mu\frac{\partial C}{\partial V_t}(s,0,t,\rho) + k_\rho(\mu_\rho-\rho_t)\frac{\partial C}{\partial \rho_t}(s,0,t,\rho)\\ +\frac{1}{2}\left(\sigma_\rho{}^2(1-\rho_t^2)\right)\frac{\partial^2 C}{\partial \rho_t^2}(s,0,t,\rho) - rC(s,0,t,\rho) = 0\end{gathered} \tag{6.25}$$

$$C(s,\infty,t,\rho) = S \tag{6.26}$$

$$\begin{gathered}\frac{\partial C}{\partial t}(s,v,t,-1) + rS_t\frac{\partial C}{\partial S_t}(s,v,t,-1) + k_v(\mu-V_t)\frac{\partial C}{\partial V_t}(s,v,t,-1)\\ +k_\rho(\mu_\rho+1)\frac{\partial C}{\partial \rho_t}(s,v,t,-1) + \frac{1}{2}\left(V_tS_t{}^2\right)\frac{\partial^2 C}{\partial S_t^2}(s,v,t,-1) + \frac{1}{2}\left(\sigma^2V_t\right)\frac{\partial^2 C}{\partial V_t^2}(s,v,t,-1)\\ +\frac{1}{2}\left(\sigma_\rho{}^2\right)\frac{\partial^2 C}{\partial \rho_t^2}(s,v,t,-1) - (\sigma V_tS_t)\frac{\partial^2 C}{\partial S_t\partial V_t}(s,v,t,-1) - rC(s,v,t,1) = 0\end{gathered} \tag{6.27}$$

$$\begin{aligned}
&\frac{\partial C}{\partial t}(s,v,t,1) + rS_t\frac{\partial C}{\partial S_t}(s,v,t,1) + k_v(\mu - V_t)\frac{\partial C}{\partial V_t}(s,v,t,1) \\
&+ k_\rho(\mu_\rho - 1)\frac{\partial C}{\partial \rho_t}(s,v,t,1) + \frac{1}{2}\left(V_t {S_t}^2\right)\frac{\partial^2 C}{\partial S_t^2}(s,v,t,1) + \frac{1}{2}\left(\sigma^2 V_t\right)\frac{\partial^2 C}{\partial V_t^2}(s,v,t,1) \\
&+ \frac{1}{2}\left({\sigma_\rho}^2\right)\frac{\partial^2 C}{\partial \rho_t^2}(s,v,t,1) + \left(\sigma V_t S_t\right)\frac{\partial^2 C}{\partial S_t \partial V_t}(s,v,t,1) - rC(s,v,t,1) = 0
\end{aligned} \tag{6.28}$$

By incorporating the above-mentioned conditions into the PINNs model we can train the extended Heston model PDE perfectly. As we discussed the PINNs implementing the process for the Heston model, same way we can get the European call option prices of the extended Heston model.

Chapter 7

Numerical Results

In this chapter, we conduct an in-depth exploration of the numerical results obtained by solving the Heston PDE through PINNs to determine European call option prices. Following, we perform a comparative analysis between the results obtained through PINNs and the values from the Heston model, utilizing online calculators. As a concluding step, our examination extends to the application of PINNs on the PDE of the extended Heston model. We then present and discuss the outcomes derived from this application.

7.1 Results for Heston model European Call option price using PINNs

We solved the Heston model PDE using a deep learning method called PINNs to calculate European call option prices as we explained in the previous chapters. The results obtained from PINNs model are firstly compared with the Heston model which is calculated using an online calculator for different stock prices.

In this experiment, we investigated the performance of the Heston model values of PINNs with the Heston model values using an online calculator. In the following sections, we present the numerical results provided by the PINNs methodology. Our analysis begins by conducting comprehensive PINNs experiments, the outcomes of which hold key insights into European Call Option prices. To provide more understanding of the results, we initiate our discussion with the European call option prices obtained from the Heston model using PINNs.

Our PINNs simulation process is performed with 3 Inputs(stocks, time, and volatility), 5 hidden layers, 128 neurons per each hidden layer, and iterated for different epochs (10000, 15000, 20000, 25000). We define the error for the European call of the Heston model using PINNs with the Heston model European call values using an Online calculator.

While solving the pricing PDE using PINNs we used these parameters: strike price (K) is 40, interest rate (r) is 0.05, annual volatility (σ) is 0.001, mean reversion speed (k_v) is 0.1, long term average (μ_v) is 0.01, volatility of volatility (σ_v) is 0.001, correlation (ρ) is -0.167, initial volatility (V_0) is 0.03, time to maturity(T) is 1.

By using the above parameters the PINNs model provided the below European Call prices for different stocks after 10000, 15000, 20000, and 25000 epochs (Iterations).

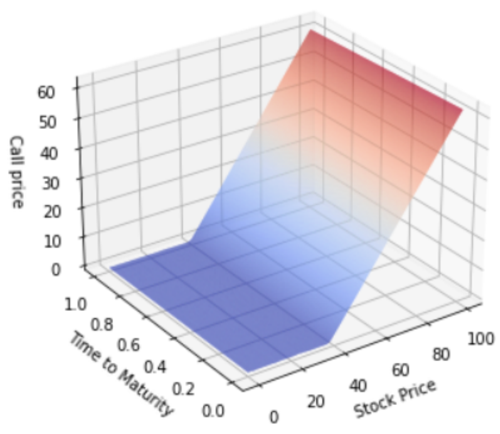
Table 7.1: European Call Option Prices in the Heston Model with PINNs after 10000 Epochs

Stock Price	Heston model Call price
30	0.1953921
38	0.6096941
40	2.1391237
60	21.939014
64	25.940650
68	29.942282
80	41.947170

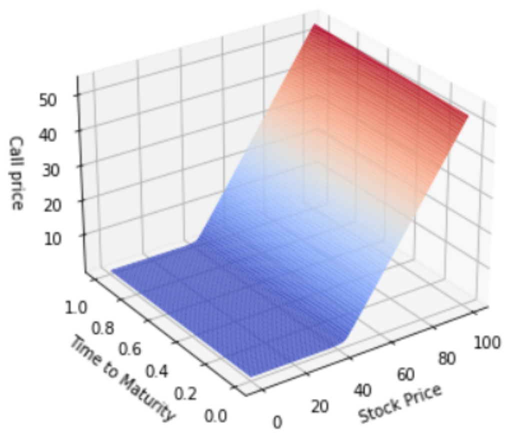
The table 7.1 presents the European call option prices in the Heston model, computed using the PINNs model after 10,000 epochs. The calculations are based on specific parameters that we already mentioned in the above. The table displays option values for select stocks within the range of 0 to 100, which means focusing on Out of the Money (OTM), At the Money (ATM), and In the Money (ITM) stocks.

The PINNs method solves the Heston model PDE using different numbers of epochs. By choosing the best hyperparameters, we can produce accurate results. Whenever we are using different numbers of epochs we are going to get different results with very small deviations, we can compare our PINNs results with the online calculator for the Heston model, and then we observe the point at which accurate results are achieved during a specific number of epochs.

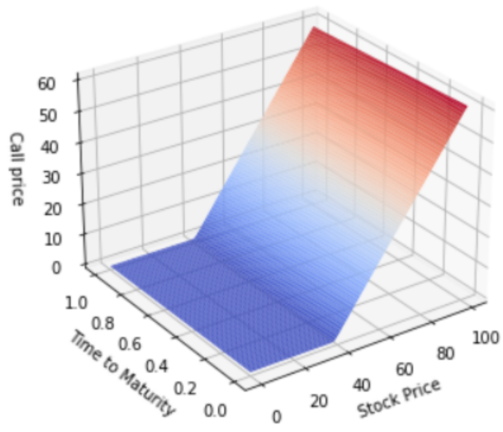
Consider a European call option, when the stock price is getting away from the At the Money (ATM) to In the Money (ITM) the call values start increasing gradually. When the stocks are increasing then European call option values are also increasing.



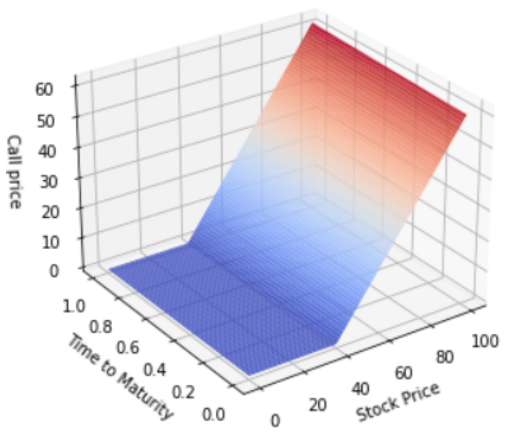
(a) PINNs after 10000 epochs



(b) PINNs after 15000 epochs



(c) PINNs after 20000 epochs



(d) PINNs after 25000 epochs

Figure 7.1: Predicted European Call Option Values in the Heston Model using PINNs after different epochs

Table 7.2: European Call Option Prices in the Heston Model with PINNs after 15000 Epochs

Stock Price	Heston model Call price
30	0.1832670
38	0.4463621
40	1.7994881
60	21.946484
64	25.943253
68	30.939215
80	41.930340

Table 7.3: European Call Option Prices in the Heston Model with PINNs after 20000 Epochs

Stock Price	Heston model Call price
30	0.1953201
38	0.5146402
40	2.0569086
60	22.062744
64	26.064552
68	30.066383
80	42.071835

From the table 7.5 we can see that after 15000 epochs, PINNs predicted European call option price is giving very little error when we are comparing with the true values. So, from the above experiments, we can consider after 15000 epochs PINNs giving the accurate results for the Heston model to calculate European call option prices.

The presented results showcase the Heston model European call option prices predicted by PINNs. The model underwent training across various epochs, including 10000, 15000, 20000, and 25000, resulting in different option prices for the Heston model. The numerical results after several experiments displayed above depict option prices for different stocks, categorized as Out of the Money, At the Money, and In the Money values. The utilization of different epochs served the purpose of identifying optimal hyperparameters, aiming to achieve effective training of the

Table 7.4: European Call Option Prices in the Heston Model with PINNs after 25000 Epochs

Stock Price	Heston model Call price
30	0.2201670
38	0.6418986
40	1.5574714
60	21.994915
64	25.993532
68	29.990173
80	42.122370

PINNs model.

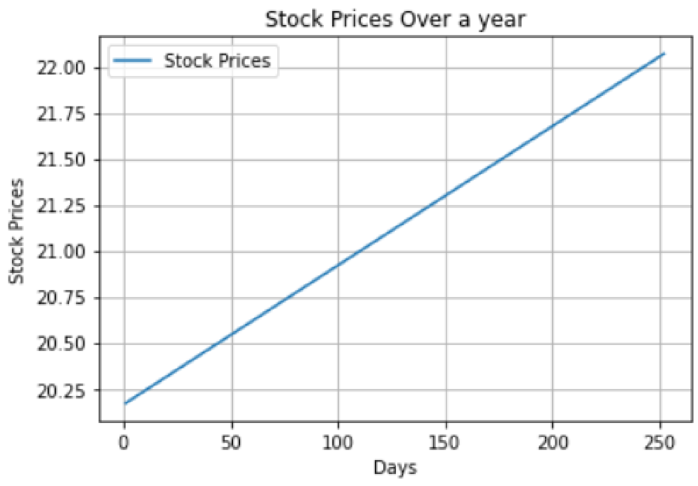


Figure 7.2: European call over a 1 year trading days

The figure 7.2 illustrates the PINNs predicted European call option price of a Heston model of stock value 60. We can see the PINNs predicted option price of stock value (S) = 60, at the strike price (K) = 40, interest rate (r) = 0.05, annual volatility (σ) = 0.001, mean reversion speed (k_v)= 0.1, long term average (μ_v) = 0.01, correlation (ρ) = -0.167, initial volatility (V_0) = 0.03 over the trading days in an overall year, which means 252 trading days. In this figure, we can observe how the option price is increasing day by day.

Table 7.5: PINNs predicted European Call Option Prices at Various Epochs for Different Stock Prices

Stock Price	10000	15000	20000	25000
60	21.939014	21.946484	22.062744	21.994915
61	22.939423	22.945671	23.063202	22.994566
62	23.939837	23.944864	24.063656	23.994226
63	24.940237	24.944054	25.064121	24.993876
64	25.940650	25.943253	26.064552	25.993532
65	26.941061	26.942436	27.065022	26.993181
66	27.941454	27.941631	28.065470	27.992830
67	28.941875	28.940826	29.065935	28.992140
68	29.942282	29.940020	30.066383	29.990173
69	30.942692	30.939215	31.066847	30.988838
70	31.943094	31.938408	32.067270	31.988577
71	32.943516	32.937610	33.067738	32.998295
72	33.943910	33.936800	34.068200	34.012510
73	34.944317	34.935986	35.068660	35.026714
74	35.944730	35.935184	36.069107	36.040924
75	36.945137	36.934380	37.069550	37.055130
76	37.945538	37.933575	38.070007	38.069344
77	38.945953	38.932766	39.070446	39.083546
78	39.946360	39.931957	40.070896	40.097750
79	40.946774	40.931145	41.071377	41.111973
80	41.947170	41.930340	42.071835	42.122370

7.2 Comparision of Heston model with Online Calculator

In the previous section, we predicted the European call option values using PINNs at different numbers of epochs for different stocks. Now we are going to compare the predicted values with the true values calculated using an online calculator https://kluge.in-chemnitz.de/tools/pricer/ is designed by kluge. The online calculator provides the values using the Finite Differences Method (FDM). Taking the online calculator values as a benchmark we are going to calculate the error for the Heston model European call prices using PINNs.

The European call price values using PINNs using the same parameters mentioned

Table 7.6: Predicted European Call Option Values in the Heston Model using PINNs vs True values

Stock Price	PINNS Call price	Online Call price	Absolute relative Error
30	0.1953910	0.2126015	0.0172105
38	0.6096941	2.5570420	1.9473479
40	2.1391237	3.7385004	1.5993767
60	21.946484	21.9602731	0.043735
64	25.943253	25.9535098	0.010256
68	29.940020	29.9515581	0.011538
72	33.936800	33.9510183	0.014218
80	41.930340	41.950836	0.020496

above. We can observe that, there is a big margin of error for stock values of 38 and 40, which stocks are less than the strike price, and a stock price of 40 is called the At the Money (ATM) value, which means the value of the stock price and the strike price is the same then the particular stock prices are called as the At the Money (ATM) values. When it comes to the In the Money (ITM) values, it means the stock price is higher than the strike price, At In the Money (ITM) values the error margin between the European call option price using PINNs to the Online calculator the relative error is much less.

Table 7.7 shows the Heston model European call option values epochs like 10000, 15000, 20000, and 25000 performed by PINNs for In the Money (ITM) values. In the final column, we noted the True values which mean Heston model Call option values calculated by using the Finite Difference Method (FDM) calculator.

The figure 7.3 shows the comparison between the Heston model European call values predicted by using PINNs at different epochs like 10000, 15000, and 25000 to the True Heston model values calculated by using the online calculator which is calculated using the numerical method Finite Difference Method (FDM). Both are calculated using the same parameters. From this, we can observe that At the Money (ATM) stocks are deviating from the Heston true values that we calculated using the Finite Difference method (FDM), PINNs are not producing the expected values at At the Money (ATM) stocks, but PINNs are producing accurate results for In the Money (ITM) stocks.

We considered the online calculator European call option price values as benchmark values because it calculates the European call option price values using the Finite

Table 7.7: Comparative Analysis of European Call Option Prices at Various Epochs Vs Actual Values for Different Stock Prices

Stock Price	10000	15000	20000	25000	Online Calculator
60	21.939014	21.946484	22.062744	21.994915	21.960273
61	22.939423	22.945671	23.063202	22.994566	22.957752
62	23.939837	23.944864	24.063656	23.994226	23.955889
63	24.940237	24.944054	25.064121	24.993876	24.954517
64	25.940650	25.943253	26.064552	25.993532	25.953510
65	26.941061	26.942436	27.065022	26.993181	26.952772
66	27.941454	27.941631	28.065470	27.992830	27.952234
67	28.941875	28.940826	29.065935	28.992140	28.951843
68	29.942282	29.940020	30.066383	29.990173	29.951558
69	30.942692	30.939215	31.066847	30.988838	30.951352
70	31.943094	31.938408	32.067270	31.988577	31.951203
71	32.943516	32.937610	33.067738	32.998295	32.951096
72	33.943910	33.936800	34.068200	34.012510	33.951018
73	34.944317	34.935986	35.068660	35.026714	34.950963
74	35.944730	35.935184	36.069107	36.040924	35.950923
75	36.945137	36.934380	37.069550	37.055130	36.950894
76	37.945538	37.933575	38.070007	38.069344	37.950874
77	38.945953	38.932766	39.070446	39.083546	38.950859
78	39.946360	39.931957	40.070896	40.097750	39.950849
79	40.946774	40.931145	41.071377	41.111973	40.950841
80	41.947170	41.930340	42.071835	42.122370	41.950836

Difference method (FDM). After the comparison between the PINNs predicted European call option prices to the Online calculated option prices, we can observe that PINNs model gives accurate results, particularly after the comparison for In the Money (ITM) stocks. The PINNs predicted values are not very accurate compared to the At the Money (ATM) stocks. For example, the European call option price for At the Money (ATM) using PINNs is 2.2994223 at 15000 epochs, but the Option price value using the Online calculator is 3.738005, here we can observe the error is 1.43858. if we consider one In the Money (ITM) value, for example at 64 PINNs predicted option price is 25.943253 the Online calculator option price is 25.9535098 and the error is 0.010256.

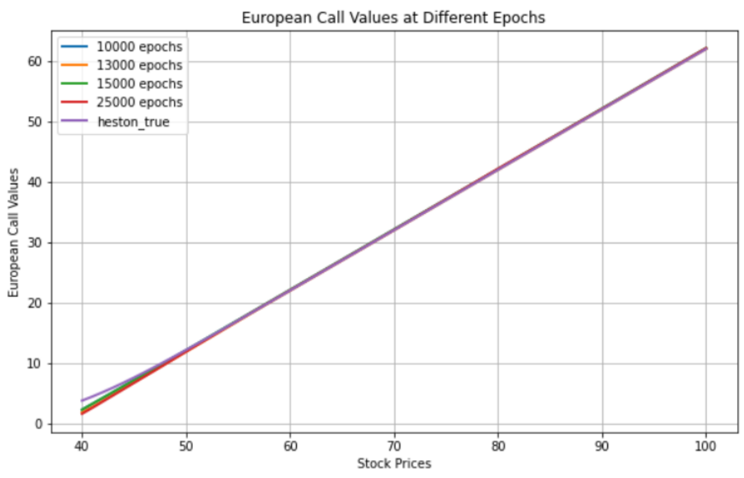


Figure 7.3: European Call Prices Across Various Epochs Vs Actual(True) Call Prices

7.3 Comparision of Heston model with Black Scholes model

As we know Black Scholes model is considered one of the best option models in financial mathematics. The Heston model is an extension of the Black Scholes. Now we are going to consider the parameters to calculate the European call option prices using the Black Scholes model. As we know already the Black-Scholes model follows a constant volatility, Whereas the Heston model follows a stochastic volatility process. In the last 2 sections, we predicted the Heston model option prices using PINNs and after comparing it with online calculators, now we will compare At the Money and In the Money stocks with the Black Scholes model.

In the table 7.8 Our PINNs model European call values compared with a Heston model online calculator and Black Scholes model. We considered the PINNs predicted European call option price after 15000 epochs, which provides the best results in our experiments. Here the PINNs model and Heston model values are calculated using the parameters at strike price (K) = 40, interest rate (r) = 0.05, annual volatility (σ) = 0.001, mean reversion speed (k_v)= 0.1, long term average (μ_v) = 0.01, correlation (ρ) = -0.167, initial volatility (V_0) = 0.03 and, at time to maturity(T) = 1. When it comes to the Black Scholes model, as we know Black Scholes model follows constant volatility, The Black Scholes model call values at the strike price (K) = 40, interest rate (r) = 0.05, annual volatility (σ) = 0.001, volatility (V) = 0.03 and time to maturity(T) = 1. We can see the difference between the PINNs model, the Heston model Online calculator calculated values using the Finite Difference method (FDM), and with Black Scholes model value volatility 0.03.

Table 7.8: PINNs call values comparison with Heston model FDM, and Black scholes

Stock Price	PINNs	Heston Online	Black Scholes
60	21.946484	21.960273	21.950823
61	22.945671	22.957752	22.950823
62	23.944864	23.955889	23.950823
63	24.944054	24.954517	24.950823
64	25.943253	25.953510	25.950823
65	26.942436	26.952772	26.950823
66	27.941631	27.952234	27.950823
67	28.940826	28.951843	28.950823
68	29.940020	29.951558	29.950823
69	30.939215	30.951352	30.950823
70	31.938408	31.951203	31.950823
71	32.937610	32.951096	32.950823
72	33.936800	33.951018	33.950823
73	34.935986	34.950963	34.950823
74	35.935184	35.950923	35.950823
75	36.934380	36.950894	36.950823
76	37.933575	37.950874	37.950823
77	38.932766	38.950859	38.950823
78	39.931957	39.950849	39.950823
79	40.931145	40.950841	40.950823
80	41.930340	41.950836	41.950823

In the figure 7.4 we can observe the graph between Stock prices to European call values for different models. Our PINNs model compared with the Heston model values calculated by FDM and Black Scholes model. Here we can observe at At the money (ATM) value which means at 40, we are getting a small deviation from the European call calculated by using FDM.

From the above numerical results, we can observe that the error between the Heston model and true values (Online calculator) are almost the same. When it comes to the In the Money (ITM) values European call option prices using PINNs are a little less relative to the Black-Scholes model and Online Heston option prices. The difference between the PINNs predicted values to the other 2 models is very small. At a stock value of 64, the European call option price predicted by PINNs is

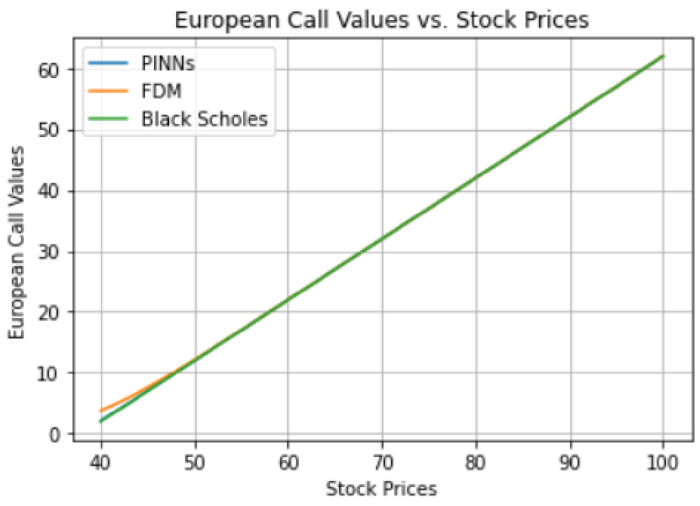


Figure 7.4: PINNs vs Heston vs Black Scholes

25.943253, at the same stock the Online calculated option price is 25.953510, and the Black Scholes option price is 25.950823. The difference between the PINNs to the Online option price is 0.010. we can consider this difference as very low. The difference between the PINNs option price and to Black Scholes option price is 0.00757. from these relative differences, we can note that PINNs predictions are very accurate for the Heston model.

7.4 Results for the Extended Heston model European Call option price using PINNs

Till now we predicted the Heston model PDE using PINNs, Now we are going to implement the PINNs on our derived extended Heston model PDE. In the previous chapter, we implemented the stochastic correlation process to the Heston model and derived the extended Heston model. We are implementing the PINNs method to calculate the extended Heston model European call option prices.

Considering the below parameters and solving the above pricing equation using the PINNs method. We are considering a strike price (K) is 40, interest rate (r) is 0.05, annual volatility (σ) is 0.25, mean reversion speed for volatility(k_v) is 2, mean reversion speed for correlation(k_ρ) is 2, long term average of volatility(μ_v) is 0.01, long term average of correlation(μ_ρ) is 0.01, volatility of volatility (σ_v) is 0.25, correlation (ρ_t) is 0.3, and correlation (ρ_{cs}) is 0.3.

Here are the European option call price values of the Heston model using PINNs. We implemented the PINNs model on the extended Heston model PDE with 4 values as inputs (Stocks, time, volatility, and correlation), 5 hidden layers and each layer has 128 neurons, and one output as a Call value. Below PINNs predicted results are obtained after 15000 epochs.

Table 7.9: Extended Heston Model option price call values at correlation ρ= 0.3, Time to Maturity t = 0.25 with an initial variance $V_0 = 0.03$

Stock Price	Extended Heston model Call price
60	20.707626
62	22.722560
64	24.737535
66	26.744610
68	28.749668
70	30.754711

Table 7.10: Extended Heston Model option price call values at correlation $\rho = -0.3$, Time to Maturity t = 0.50 with an initial variance $V_0 = 0.03$

Stock Price	Extended Heston Model Call price
60	21.096830
62	23.101880
64	25.106922
66	27.111969
68	29.117023
70	31.122066

From the four tables (7.9, 7.10, 7.11, 7.12) we can see the European call values of the extended Heston model calculated by using PINNs model at 15000 epochs. The stock values are considered in between 60 to 80. here we considered the In the Money (ITM) values. As the stock values are increasing for In the Money values, the European call values are also gradually increasing. The four tables exhibit the European call option prices of the extended Heston model at different time maturities, different correlations, and different conditions.

Table 7.11: Extended Heston Model option price call values at correlation $\rho = 0.4$, Time to Maturity t = 0.75 with an initial variance $V_0 = 0.03$

Stock Price	Extended Heston Model Call price
60	21.457830
62	23.462881
64	25.467928
66	27.472982
68	29.478025
70	31.483074

Table 7.12: Extended Heston Model option price call values at correlation $\rho = -0.4$, Time to Maturity t = 1 with an initial variance $V_0 = 0.03$

Stock Price	Extended Heston Model Call price
60	21.826176
62	23.831223
64	25.836270
66	27.841328
68	29.846367
70	31.851416

7.5 Comparision of Extended Heston model with Heston model

In the previous section, we observed the PINNs predicted European call option prices of the extended Heston model for different stocks. Now we will compare the the Heston model and extended Heston model European call option values which are predicted by using the PINNs, and at the same time we will compare these 2 PINNs predicted values with an Online calculator.

The Heston model values predicted at a strike price $(K) = 40$, interest rate $(r) =$ 0.05, annual volatility $(\sigma) = 0.001$, mean reversion speed (k_v)= 0.1, long term average $(\mu\_v) = 0.01$, correlation (ρ) = -0.167, initial volatility $(V_0) = 0.03$

The extended Heston model values predicted at a strike price $(K) = 40$, interest

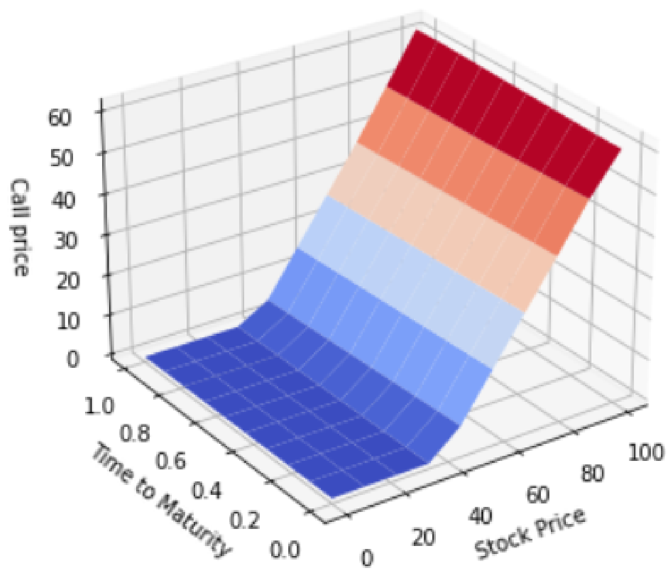


Figure 7.5: Call Prices in Extended Heston Model using PINNs

rate (r) = 0.05, annual volatility (σ) = 0.001, mean reversion speed of volatility (k_v)= 0.1, mean reversion speed of correlation (k_ρ)= 0.1, long term average of volatility (μ_v) = 0.01, long term average of volatility of correlation (μ_ρ) = 0.01, correlation ρ= 0.3, time to maturity t = 1 with an initial variance V_0 = 0.03.

In the above table 7.13 we compared different In the Money (ITM) stocks European call option prices for the extended Heston model and Heston model. The first 2 columns in the above table are predicted by using the PINNs model for the extended Heston model and Heston model respectively. finally, we compared these two PINNs predicted option prices with the Heston model online calculator values, which are calculated by using the numerical method Finite Difference method (FDM).

Table 7.13: PINNs predicted Extended Heston Model Vs Heston model

Stock Price	Extended Heston model	Heston model	Online Calculator
61	22.716866	22.945671	22.957752
62	23.719040	23.944864	23.955889
63	24.721224	24.944054	24.954517
64	25.723398	25.943253	25.953510
65	26.725570	26.942436	26.952772
66	27.727753	27.941631	27.952234
67	28.729916	28.940826	28.951843
68	29.732107	29.940020	29.951558
69	30.734280	30.939215	30.951352
70	31.736452	31.938408	31.951203
71	32.738613	32.937610	32.951096
72	33.740810	33.936800	33.951018
73	34.742985	34.935986	34.950963
74	35.745163	35.935184	35.950923
75	36.747334	36.934380	36.950894
76	37.749516	37.933575	37.950874
77	38.751143	38.932766	38.950859
78	39.753864	39.931957	39.950849
79	40.756030	40.931145	40.950841
80	41.758230	41.930340	41.950836

Chapter 8

Discussions

In this chapter, we are going to discuss the numerical results that we produced in the previous chapter. We implemented the PINNs deep learning method on the Heston model PDE and extended model PDE. We will discuss how the PINNs model behaved for At the Money (ATM) stocks, In the Money (ITM) stocks, and Out of the Money (OTM) stocks. Is the PINNs method suitable to solve the Heston and extended Heston model PDEs or not?

In the Money (ITM) stocks, which means if the stock prices are more than the Strike price then the particular stock prices are called In the Money (ITM) stocks. In Numerical results, we already observed that the PINNs predictions for In the Money (ITM) values are very accurate for different epochs. Even the error margin is very low when we are comparing our PINNs predicted values with the Finite Difference Method (FDM) online calculator for Heston model European call option price values. For In the Money (ITM) values PINNs are the best method to solve the Heston model.

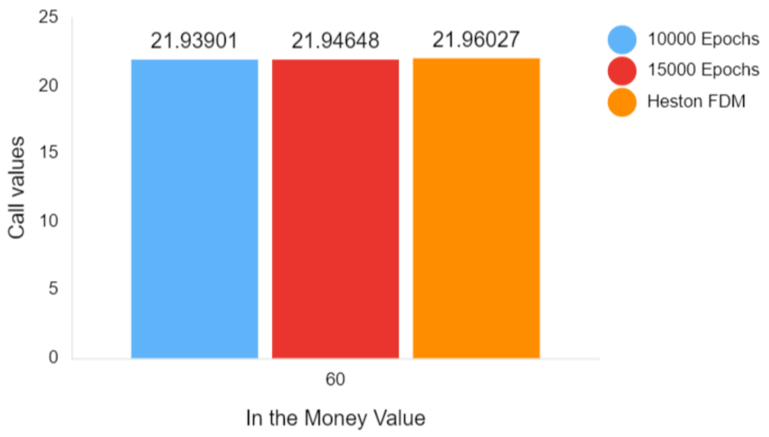


Figure 8.1: European Call Option Price for In the Money Value stock with price 60

We considered 2 In the Money (ITM) values to show the difference between the

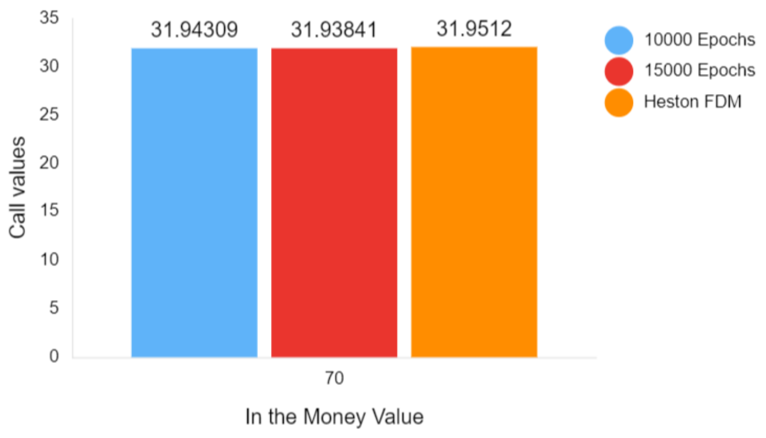


Figure 8.2: European Call Option Price for In the Money Value stock with price 70

PINNs prediction at different epochs to the Finite Difference Method (FDM). At the stock price of 60, PINNs predicted values at 10000 and 15000 epochs. at 10000 epochs the European option call value is 21.93901, at 15000 epochs PINNs predicted value is 21.94648. Now we considered the European call value from the Finite Difference method (FDM) as 21.96027. Here we can observe that PINNs predicted value is very close to the FDM values calculated using an online calculator. In the same way, it's producing values for another stock price of 70 also. From the figures 8.1 8.2 we can say that PINNs predict the more accurate results at In the Money (ITM) values.

Out of the Money (OTM) stocks, which means if the Stock price is the less than to the Strike price, then the stock price is called the Out of the Money (OTM) value. As we have seen in the numerical results section Out of the Money stocks, the PINNs prediction is not accurate.

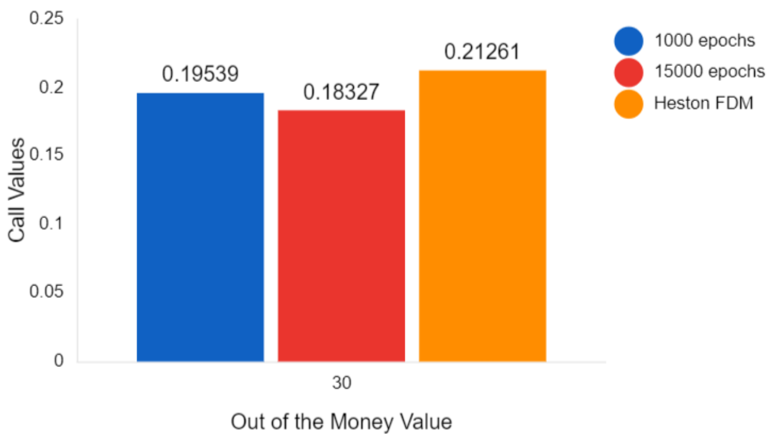


Figure 8.3: European Call Option Price for Out of the Money Stock Value

In the figure 8.3 we can see the error between the PINNs predicted values at different epochs to the True values. From the numerical results, it is validated that PINNs can provide promising results for Out of the Money (OTM) values.

At the Money (ATM) stocks, which means if the Stock price is the same as the Strike price, then the stock price is called the At the Money (ATM) value. As we have seen in the numerical results section particularly at At the Money (ATM) stocks, the PINNs prediction is not being accurate at any epochs. The PINNs predicting the European Call option prices at ATM stocks are between 1.5 to 2.5. But the European call option price using the Finite Differences Method (FDM) value is 3.7385004.

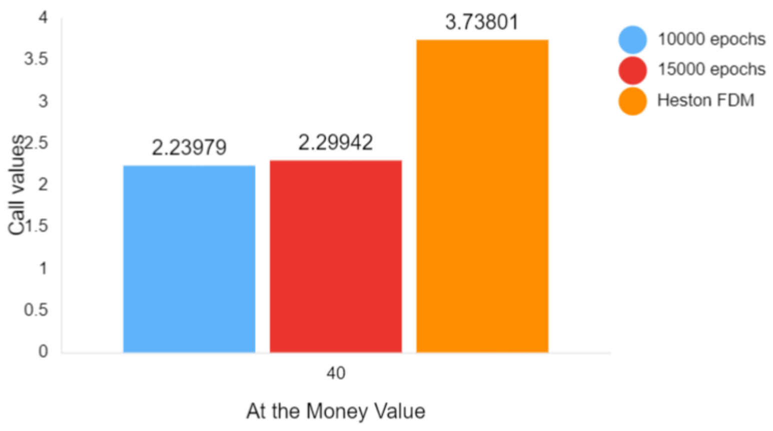


Figure 8.4: European Call Option Price for At the Money Stock Value

As we already mentioned we can see the difference for European option call price values in the figure 8.4 of At the Money value. By using PINNs we predicted the above results for 10000 and 15000 epochs. On the other side, we calculated the Heston model European option call price value using the Finite Difference method. The figure shows that PINNs didn't provide accurate results at At the Money (ATM) value.

Chapter 9

Conclusion and Outlook

We have implemented Physics Informed Neural Networks (PINN) to solve the Heston and extended Heston model PDEs for European call option prices. In this work, we derived the PDEs of the Heston model using the Feynman-Kac theorem. Subsequently, we extended the model by incorporating a stochastic correlation between the underlying asset and stochastic volatility. The PDE for the extended Heston model was also derived using the Feynman-Kac theorem. We implemented the Deep learning technique called PINNs for solving the PDEs and nonlinear PDEs. After a series of experiments using PINNs on the Heston model, the resulting methods showcased promising results for both the Heston and extended Heston models, particularly at In the Money (ITM) stocks.

The PINNs can provide promising results for different problems in financial mathematics. Machine learning and deep learning are continuing to grow rapidly both in terms of methodological and algorithmic developments, we believe that the PINNs technique is the best alternative method to solve PDEs. We should observe that the proposed PINNs method is just an alternative method, and should not be viewed as a replacement of classical numerical methods for solving PDEs like the Finite element method, and spectral method.

The PINNs predicted Heston model European call option prices are compared with an online calculator, which derives the European call prices using Finite Difference Methods (FDM). We compared our Heston model predicted results with an online calculator in three sections, Out of the Money (OTM), At the Money (ATM) stocks, and In the Money (ITM) stocks. Our numerical results demonstrated that PINNs provide the results with great accuracy for In the Money (ITM) stocks. The PINNs predictions for In the Money stocks align well with the online calculator values. for Out of the Money (OTM) stocks, PINNs are providing promising results. However, while examining the At the Money (ATM) stocks, we observed that PINNs predicted values exhibit some deviation when compared to the values obtained from the online calculator using FDM. The experiments on the Heston model to calculate the European call option price using PINNs show that PINNs can be used to solve the Heston model for option pricing. Particularly for In the Money (ITM) stocks, it will give very accurate results than At the Money (ATM) stocks.

In this work after implementing the PINNs method on the Heston model, we have investigated an effective implementation of PINNs on the extended Heston model by incorporating a stochastic correlation between the underlying asset and stochastic volatility. The derived PDE was solved using the PINNs setup, and PINNs predicted the promising European call option prices for the extended Heston model at In the Money (ITM) values. The presented numerical results show that significant accurate European call option prices for In the Money (ITM) stocks can be achieved by solving the Heston model and extended Heston model PDEs using PINNs. When it comes to the At the Money (ATM) and Out of the Money (OTM) values, PINNs can achieve good results but there is an error when compared with benchmark values.

After implementing the PINNs method on the Heston model and the extended Heston model we are getting accurate results for In the Money values, when it comes to At the Money values, we are not getting accurate results. The reasons for the inaccurate results are as follows. Firstly, there could be a coding problem while training the model, which leads to inaccurate results. Secondly, we weren't able to find the best tuning hyperparameters to train the model perfectly. These reasons affected the model and led to inaccurate results for At the Money values.

Bibliography

[1] M. N. Alonso and D. Maxwell, "Physics-informed neural networks (pinns) in finance," *SSRN*, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4598180.

[2] J. Bai, H. Jeong, and S. Xiao, "An introduction to programming physics-informed neural network-based computational solid mechanics," *arxiv*, 2023. [Online]. Available: https://arxiv.org/abs/2210.09060.

[3] H. Baty and L. Baty, "Solving differential equations using physics informed deep learning: A hand-on tutorial with benchmark tests," *HAL*, 2023. [Online]. Available: https://arxiv.org/abs/2302.12260.

[4] A. G. Baydin, "Automatic differentiation in machine learning: A survey," *Arxiv*, 2018. [Online]. Available: https://arxiv.org/pdf/1502.05767.pdf.

[5] M. Bazmara, M. Mianrood, and M. Silani, "Application of physics-informed neural networks for nonlinear buckling analysis of beams," *Acta Mechanica Sinica*, 2022. [Online]. Available: https://doi.org/10.1007/s10409-023-22438-x.

[6] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials," *physics.comp-ph*, 2020. [Online]. Available: https://arxiv.org/abs/1912.01085.

[7] B. cheng and D.M.Titterington, "Neural networks: A review from a statistical perspective," *Statistical Science*, vol. 9, no. 1, 1994. [Online]. Available: https://www.jstor.org/stable/2246275.

[8] W. contributors. "Black-scholes-merton model." Last accessed: October 23, 2023. (2023), [Online]. Available: https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model#Black%E2%80%93Scholes_formula.

[9] J. C. Cox, J. E. Ingersoll.Jr, and S. A. Ross. "A theory of the term structure of interest rates." (1985).

[10] S. Cuomo, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics-informed neural networks: Where we are and what's next," *arxiv*, 2022. [Online]. Available: https://arxiv.org/abs/2201.05624.

[11] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neuro Computing, Elsevier*, 2022. [Online]. Available: https://arxiv.org/abs/2109.14545.

[12] R. Dunn, P. Hauser, T. Seibold, and H. Gong, "Estimating option prices with heston's stochastic volatility model," *Valparaiso University Department of Mathematics and Statistics*, [Online]. Available: https://www.valpo.edu/mathematics-statistics/files/2015/07/Estimating-Option-Prices-with-Heston%E2%80%99s-Stochastic-Volatility-Model.pdf.

[13] U. G. E and Ornstein, "On the theory of brownian motion," *Physical Review*, vol. 36, 1930.

[14] O. E.BARNDORFF-NIELSEN and A. BASSE-O'CONNOR, "Quasi ornstein–uhlenbeck processes," *Bernoulli*, vol. 17, no. 3, pp. 916–941, 2009. [Online]. Available: https://arxiv.org/abs/0912.3091.

[15] H. Fink, H. Port, and G. Schlüchtermann, "Free cir processes," *ARXIV*, 2019. [Online]. Available: https://arxiv.org/abs/1912.11714.

[16] B. Fischer and S. Myron, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973. DOI: https://www.journals.uchicago.edu/doi/10.1086/260062. [Online]. Available: https://www.jstor.org/stable/1831029.

[17] ulia García Cabello, "Mathematical neural networks," *Axioms*, 2022. [Online]. Available: https://www.mdpi.com/2075-1680/11/2/80.

[18] S. L. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *The Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, 1993.

[19] T. F. Heston, "Quantifying uncertainty: Potential medical applications of the heston model of financial stochastic volatilit," *SSRN*, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4650868.

[20] M. Higgins, "Stochastic spot/volatility correlation in stochastic volatility models and barrier option pricing," *Washington Square Technologies*, 2014. [Online]. Available: https://arxiv.org/abs/1404.4028.

[21] K. Itô. "On stochastic differential equations." (1951).

[22] M. Kac. "Enigmas of chance: An autobiography." (1987).

[23] A. Kovacs, L. Exl, and A. Kornell, "Physics-informed neural networks for power systems," *Communications in Nonlinear Science and Numerical Simulation*, vol. 104, pp. 104–106, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1007570421003531.

[24] L.TENG, M.EHRHARDT, and M.GUNTHER, "Numerical simulation of the heston model under stochastic correlation," *International Journal of Financial Studies*, vol. 6, no. 3, 2017. [Online]. Available: https://www.mdpi.com/2227-7072/6/1/3.

[25] L.TENG, M.EHRHARDT, and M.GUNTHER, "On the heston model with stochastic correlation," *International Journal of Theoretical and Applied Finance*, vol. 19, no. 6, pp. 94 845–94 861, 2016.

[26] I. E. Lagaris and A. C. Likas, "Neural-network methods for boundary valueproblems with irregular boundaries," *IEEE*, 2000. [Online]. Available: https://www.researchgate.net/publication/5602194_Neural-network_methods_for_boundary_value_problems_with_irregular_boundaries.

[27] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, pp. 503–528, 1989.

[28] R. Maller, G. Muller, and A. Szimayer, "Ornstein–uhlenbeck processes and extensions," *arxiv*, 2009. [Online]. Available: https://www.researchgate.net/publication/226891483_Ornstein-Uhlenbeck_Processes_and_Extensions.

[29] C. C. Margossian, "A review of automatic differentiation and its efficient implementation," *arxiv*, 2019. [Online]. Available: https://arxiv.org/abs/1811.05031.

[30] S. Natenberg, "Option volatility and pricing strategies : Advanced trading techniques for professionals," *McGraw-Hill*, 1994.

[31] R. D. Neidinger, "Introduction to automaticdifferentiation and matlab object-oriented programming," *Society for Industrial and Applied Mathematics*, vol. 52, no. 3, pp. 545–563, 2010.

[32] S. Rainer and Z. Jianwei, "Stochastic volatility with an ornstein-uhlenbeck process: An extension," *econstor*, 1998. [Online]. Available: https://www.econstor.eu/handle/10419/104833.

[33] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Numerical solution for high order differential equations using a hybrid neural network—optimization method," *Applied Mathematics and Computation*, vol. 1, no. 183, pp. 260–271, 2006. [Online]. Available: https://www.researchgate.net/publication/223495999_Numerical_solution_for_high_order_differential_equations_using_a_hybrid_neural_network-Optimization_method.

[34] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven discovery of nonlinear partial differential equations," *arxiv*, 2017. [Online]. Available: https://arxiv.org/abs/1711.10561.

[35] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations," *arxiv*, 2017. [Online]. Available: https://arxiv.org/abs/1711.10566.

[36] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *arxiv*, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0021999118307125.

[37] M. G. S, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," *IEEE Power and Energy Society General Meeting (PESGM)*, 2020. [Online]. Available: https://arxiv.org/abs/1911.03737.

[38] Z. Selk and H. Honnappa, "A feynman-kac type theorem for odes: Solutions of second order odes as modes of diffusions," *arxiv*, 2021. [Online]. Available: https://arxiv.org/abs/2106.08525.

[39] M. Suárez-Taboada, M. Witteveen, and Grzelak, "Uncertainty quantification and heston model," *Mathematics in industry*, vol. 8, no. 1, pp. 1–12, 2018. [Online]. Available: https://doi.org/10.1186/s13362-018-0047-2.

[40] L. Teng, M. Ehrhardt, and M. Gunther, "The dynamic correlation model and its application to the heston model," *BUW-IMACM*, 2016. DOI: 10.1007/978-3-319-33446-2. [Online]. Available: https://www.researchgate.net/publication/271516439_The_Dynamic_Correlation_Model_and_its_Application_to_the_Heston_Model.

[41] L. Teng, C. van Emmerich, M. Ehrhardt, and M. Gunther, "A versatile approach for stochastic correlation using hyperbolic functions," *International Journal of Computer Mathematics*, 2015. [Online]. Available: https://www.researchgate.net/publication/272013033_A_versatile_approach_for_stochastic_correlation_using_hyperbolic_functions.

[42] O. Vašíček, "An equilibrium characterization of the termstructure," *Journal of Financial Economics*, vol. 5, pp. 177–188, 1997.

[43] X. Wang, J. Li, and J. Li, "A deep learning based numerical pde method for option pricing," *Computational Economics*, 2022.