Deep Learning Approaches to Partial Differential Equations with Applications in Finance: Pricing European Style Options

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Rujul Dwivedi & Pranavi Bannela
(2103319 & 2103306)

under the guidance of

Dr. Lok Pati Tripathi

to the

SCHOOL OF MATHEMATICS & COMPUTER SCIENCE
INDIAN INSTITUTE OF TECHNOLOGY GOA
PONDA - 403401, GOA

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Deep Learning Approaches to Partial Differential Equations with Applications in Finance: Pricing European Style Options**" is a bonafide work of **Rujul Dwivedi & Pranavi Bannela** (**Roll No. 2103319 & 2103306**), carried out in the **School of Mathematics & Computer Science (SMCS)**, **Indian Institute of Technology (IIT) Goa** under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Lok Pati Tripathi**

Assistant Professor,
School of Mathematics & Computer Science,
Indian Institute of Technology Goa.

November, 2024
Autumn Semester

Acknowledgements

First and foremost, we would like to thank our supervisor, **Dr. Lok Pati Tripathi**, for his invaluable guidance, constant encouragement, and insightful feedback throughout the course of this work. His expertise and mentorship were instrumental in shaping the direction of this research.

We extend our deepest appreciation to our **parents** and **family** for their unwavering support and patience. Their belief in our abilities gave us the strength and motivation to persevere through challenges.

Special thanks to our **friends** and **colleagues** for their collaborative spirit and constructive discussions. The countless brainstorming sessions and shared learning experiences made this journey both rewarding and enjoyable.

We would also like to acknowledge the educational resources provided by various online platforms and YouTube channels such as **3Blue1Brown**, **StatQuest** with **Josh Starmer**, **Steve Brunton** and **Veritasium**, which played a crucial role in enhancing our understanding of complex concepts in deep learning and financial mathematics.

Finally, we are grateful to the **faculty** and **staff** of the **Mathematics and Computer Science Department, IIT Goa**, for fostering an environment of academic excellence and providing the necessary infrastructure and resources for this research.

This project would not have been possible without the collective efforts and contributions of everyone mentioned above. Thank you all for your invaluable support and encouragement.

Abstract

This project explores the application of deep learning techniques to the pricing of European-style options [1], utilizing Physics-Informed Neural Networks (PINNs) [2] under the Black-Scholes [3] framework and its extensions. By embedding the governing principles of financial mathematics directly into the neural network training process, we aim to solve the partial differential equations (PDEs) [4] central to option pricing. The research adopts a progressive implementation strategy, beginning with basic neural networks to address standard PDEs, and then advancing to more complex, non-linear cases. The study culminates in the integration of Long Short-Term Memory (LSTM) [5] networks within the PINN framework, known as the Deep Galerkin method (DGM) [6], enabling the modelling of multi-asset scenarios and the capture of intricate temporal dependencies. Throughout this work, we demonstrate how deep learning can provide a robust and efficient alternative to traditional numerical methods, such as the Finite Difference Method (FDM) [7] and Monte Carlo [8] simulations, which can struggle with scalability and computational cost in high-dimensional settings. The results highlight the potential of neural networks to improve accuracy, speed, and flexibility in solving complex financial models. This research not only showcases the viability of deep learning for computational finance but also sets the stage for future developments in option pricing [9] and broader financial modelling applications.

Contents

Chapter 1

Introduction

This project investigates the application of deep learning techniques to solve partial differential equations (PDEs) [4], particularly in the context of financial mathematics. The main focus is on pricing European-style options, with the **Black-Scholes** model serving as the foundational equation for the analysis. The project leverages **Physics-Informed Neural Networks (PINNs)** [2], a novel deep learning approach that integrates the physical laws governing the problem into the training process. This enables efficient and accurate solutions to PDEs, overcoming traditional numerical methods' limitations.

1.1 Background and Motivation

In financial markets, accurately pricing derivative instruments like European options is crucial for both investors and financial institutions. The Black-Scholes model has long been a staple in this regard, providing a closed-form solution for pricing options. However, traditional numerical methods such as the **Finite Difference Method (FDM)** [7] and **Monte Carlo** [8] simulations often struggle with scalability, especially when dealing with high-dimensional or multi-asset scenarios. These methods become computationally expensive and less efficient in such cases. **Physics-Informed Neural Networks (PINNs)**

[2] offer a potential solution by embedding the differential equations directly into the neural network framework, allowing for more efficient computation and flexibility in handling complex scenarios, including multi-asset options.

1.2 Organization of the Report

This report is structured as follows:

- **Chapter 2** provides the theoretical background, discussing the algorithms to solve standard PDEs using the fundamentals of **Physics-Informed Neural Networks (PINNs)** [2].
- **Chapter 3** details the implementation of the proposed deep learning models, focusing on single-asset pricing using PINNs.
- **Chapter 4** presents an enhancement to the previous PINN model by using an **LSTM layer** instead of a feedforward layer to solve multi-asset pricing options, known as the **Deep Galerkin Method (DGM)** [6].
- **Chapter 5** concludes the report, summarizing the key findings, discussing the implications of the research, and proposing directions for future work in this field.

1.3 Objectives for Upcoming Chapters

The primary objective of this project is:

- To apply the PINN framework to solve the Black-Scholes PDE and extend the model to multi-asset options by integrating **Long Short-Term Memory (LSTM)** [5] networks within the **Deep Galerkin Method (DGM)** [6]. This extension addresses the temporal dependencies and inter-relationships between multiple assets.

Chapter 2

Basic Neural Network for Non-linear PDEs

2.1 Introduction

Neural networks have become a powerful tool for solving **partial differential equations (PDEs)** [4], providing an alternative to traditional methods like **Finite Difference Methods (FDM)** [7] or **Finite Element Methods (FEM)** [10]. This chapter introduces the basic concept of neural networks applied to standard **linear** and **non-linear PDEs**. The neural network is trained to approximate the solution to a PDE by minimizing a loss function that enforces both the equation itself and its boundary conditions.

We aim to demonstrate the basic neural network approach to solving standard PDEs and extend this method to handle non-linear PDEs, which arise in many complex physical, financial, and engineering systems.

2.2 Basic Neural Network for Solving PDEs

A basic neural network for solving PDEs typically uses a feed-forward architecture, where the input consists of spatial and temporal variables, and the output represents the solution

to the PDE. The neural network approximates the solution over the entire domain.

1. **Network Architecture:** - A **multi-layer feed-forward network** is used to represent the solution $u(x,t)$, where x and t are the spatial and temporal variables, respectively. - The neural network architecture consists of an input layer, several hidden layers with activation functions, and an output layer providing the predicted solution.

2. **Loss Function:** The neural network is trained to minimize the loss function, which incorporates both the PDE and the boundary conditions:

$$\mathcal{L} = \sum_i [\text{PDE Residual}]^2 + \sum_j [\text{Boundary Condition Error}]^2$$

The PDE residual is calculated by substituting the neural network's output into the PDE and computing the difference between the left-hand and right-hand sides of the equation.

3. **Training Process:** - The neural network is trained using **backpropagation** and **gradient descent** to minimize the loss function. - The optimization process adjusts the network parameters to satisfy both the PDE and the boundary conditions.

2.3 Problem Statement and Mathematical Formulation

The PDE we aim to solve is of the following form:

$$U_t - a(x,t)U_{xx} + b(x,t)U_x + c(x,t)U = f(x,t) \tag{2.1}$$

where $U(x,t)$ represents the unknown function, U_t is the time derivative of U, U_{xx} is the second spatial derivative of U, and U_x is the first spatial derivative of U.
The functions $a(x,t)$, $b(x,t)$, and $c(x,t)$ are given, and $f(x,t)$ represents the source term.

The boundary conditions are as follows:

$$U(x_L,t) = U_B(x_L,t), \quad U(x_R,t) = U_B(x_R,t), \quad U(x,0) = U_0(x) \tag{2.2}$$

where $(x,t) \in [x_L, x_R] \times [0,T]$. The parameters for the given example are:

$$x_L = 0, \quad x_R = 0.5, \quad T = 0.5 \tag{2.3}$$

With the following specific functions:

$$a(x,t) = 1, \quad b(x,t) = 0, \quad c(x,t) = 0 \tag{2.4}$$

The boundary conditions are given by:

$$U_B(0,t) = 0, \quad U_B(0.5,t) = e^{-t}, \quad U_0(x) = \sin(\pi x) \tag{2.5}$$

The exact solution is given as:

$$U(x,t) = \sin(\pi x) \cdot e^{-t} \tag{2.6}$$

To compute the source term $f(x,t)$, we use the following relations:

1. The time derivative $U_t(x,t)$ is:

$$U_t(x,t) = -\sin(\pi x) \cdot e^{-t} \tag{2.7}$$

2. The second spatial derivative $U_{xx}(x,t)$ is:

$$U_{xx}(x,t) = -\pi^2 \cdot \sin(\pi x) \cdot e^{-t} \tag{2.8}$$

Thus, the source term $f(x,t)$ is determined as:

$$f(x,t) = U_t - U_{xx} = -\sin(\pi x) \cdot e^{-t} + \pi^2 \cdot \sin(\pi x) \cdot e^{-t} = (\pi^2 - 1)\sin(\pi x) \cdot e^{-t} \tag{2.9}$$

This PDE setup, with its boundary conditions and the exact solution, serves as the basis

for applying the **Physics-Informed Neural Network (PINN)** [2] approach to solve it numerically.

2.4 Implementation Details

The implementation of the basic neural network for solving PDEs is shown in the following code notebooks: **Code.ipynb**, **Code2.ipynb** and **Code3.ipynb**.

In both cases, the neural network is trained using **backpropagation** and the **Adam optimizer** to minimize the loss function.

2.5 Results and Comparison

Below are the graphical results obtained from solving the standard PDE using the PINN approach. The first figure illustrates the comparison between the exact solution and the numerical solution obtained from the PINN framework:

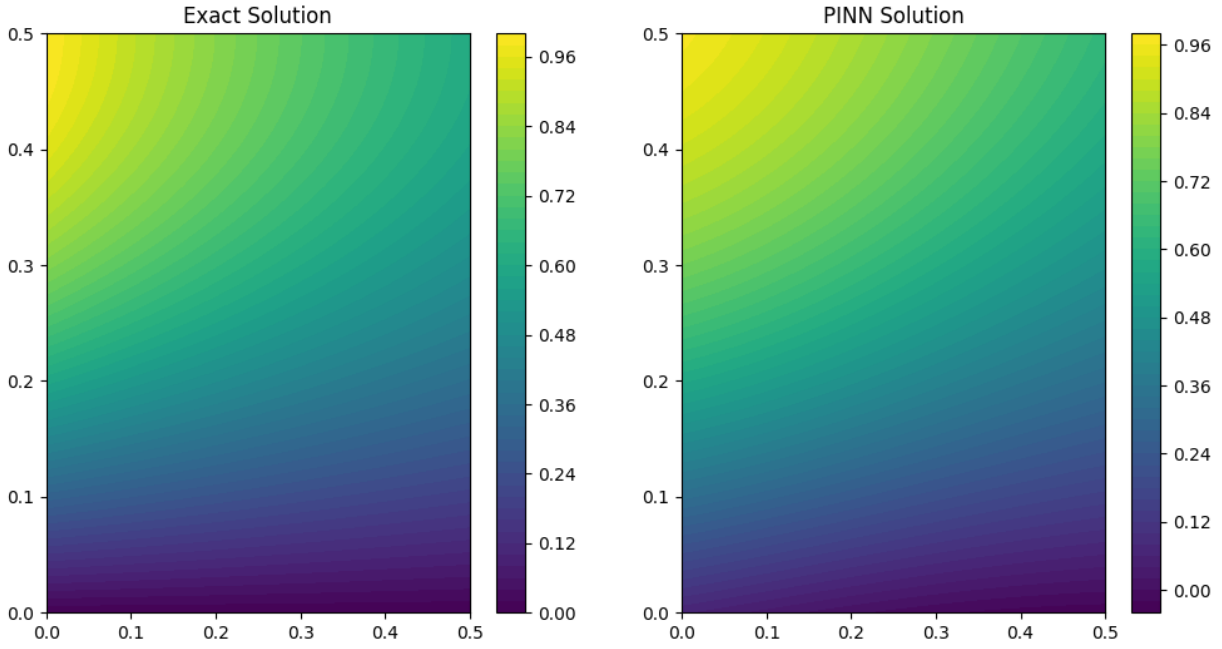

Fig. 2.1 Comparison between exact and PINN solutions for the standard PDE.

The second figure displays the results for the standard PDE solution:

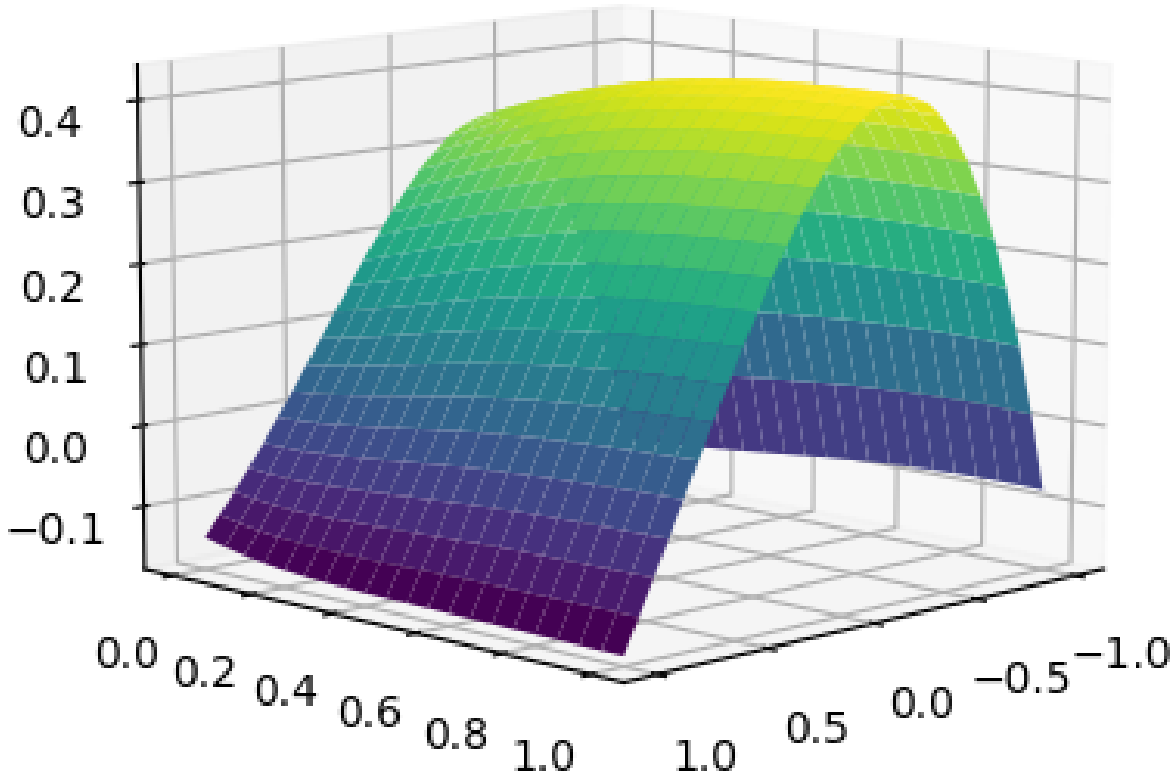

Fig. 2.2 Graph for the Standard PDE solution obtained through the PINN approach.

These figures visually highlight the effectiveness of the PINN model in solving the PDE and its ability to closely approximate the exact solution.

2.6 Conclusion

In this chapter, we introduced the basic neural network approach for solving linear and non-linear PDEs. The neural network provides a flexible framework for solving complex differential equations without relying on traditional numerical methods.

Chapter 3

PINN for Black-Scholes PDE

3.1 Introduction

Physics-Informed Neural Networks (PINNs) [2] are a class of machine learning models designed to solve **partial differential equations (PDEs)** [4] by incorporating the underlying physical laws directly into the learning process. Unlike traditional methods, which solve PDEs using **discretization techniques** (e.g., Finite Difference Methods (FDMs) [7], PINNs leverage neural networks to learn both the solution and the underlying physics. This chapter demonstrates how the PINN framework can be applied to solve the Black-Scholes PDE for European-style option pricing.

3.2 Physics-Informed Neural Networks (PINNs) [2]

A **PINN** is a neural network trained to minimize both the **traditional loss function** (e.g., mean squared error) and a **physics-based loss function**, which encodes the governing equations of the physical system. In the case of PDEs, the network is trained to satisfy both the boundary and initial conditions of the PDE, along with the equation itself, thereby learning the solution in a way that is consistent with the physics.

The main idea behind PINNs is to represent the solution to a PDE as the output of a

neural network, where the network parameters (weights and biases) are optimized through training to satisfy the equation, boundary conditions, and initial conditions.

3.3 Black-Scholes PDE

The Black-Scholes [3] equation is a fundamental model in financial mathematics used to price European-style options. It is a **parabolic partial differential equation** given by:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{3.1}$$

where:

- $C(S,t)$ is the option price as a function of the **stock price** S and **time** t,
- σ is the **volatility** of the underlying asset [11],
- r is the **risk-free interest rate** [12],
- t is the **time to expiration** [13], and
- S is the **price of the underlying asset** [14].

Boundary and initial conditions for European call options are:

- $C(S,T) = \max(S-K, 0)$, where T is the **time to maturity** [15] and K is the **strike price** [16],
- $C(0,t) = 0$, as the option price is zero when the stock price is zero,
- $C(\infty,t) = S - Ke^{-r(T-t)}$, as the option behaves like a forward contract at very high stock prices.

3.4 PINN Formulation for the Black-Scholes Equation

We begin by representing the **option price** $C(S,t)$ as the output of a neural network. The neural network takes the **stock price** S and **time** t as inputs and outputs the predicted **option price**.

The PINN framework for solving the Black-Scholes PDE is formulated as follows:

1. **Network Architecture:** - A **feed-forward neural network** with L layers and N neurons per layer. - The input layer has **two nodes** corresponding to S and t. - The output layer produces the **option price** $C(S,t)$.

2. **Loss Function:** The total loss function consists of two parts:

- *Physics Loss*: This enforces the Black-Scholes PDE:

$$\mathcal{L}_{\text{PDE}} = \left\| \frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC \right\|^2$$

- *Boundary and Initial Condition Loss*: This ensures that the neural network satisfies the boundary and initial conditions:

$$\mathcal{L}_{\text{BC/IC}} = \sum_i \left[C(S_i, 0) - \max(S_i - K, 0)\right]^2 + \sum_i \left[C(0, t_i) - 0\right]^2$$

3. **Training Process:** - The neural network is trained by minimizing the total loss function, which combines both the physics-based and data-based losses. - During training, the network learns the weights that satisfy the PDE as well as the boundary and initial conditions.

The optimization is performed using gradient descent methods, such as **Adam**, with **backpropagation** used to compute the gradients of the loss function with respect to the network parameters.

3.5 Implementation Details

The implementation of the PINN [17] framework for solving the Black-Scholes PDE is detailed in the following code notebooks: **Code4.ipynb** and **Code5.ipynb**.

In the code, the inputs to the network are the stock price S and time t, and the output is the option price $C(S, t)$. The neural network is trained to minimize the loss functions, ultimately learning an approximation to the solution of the Black-Scholes PDE.

3.6 Results and Validation

We tested the PINN model on several standard option pricing scenarios, including European call and put options [1]. The results were compared with the exact solution of the Black-Scholes equation and traditional numerical methods.

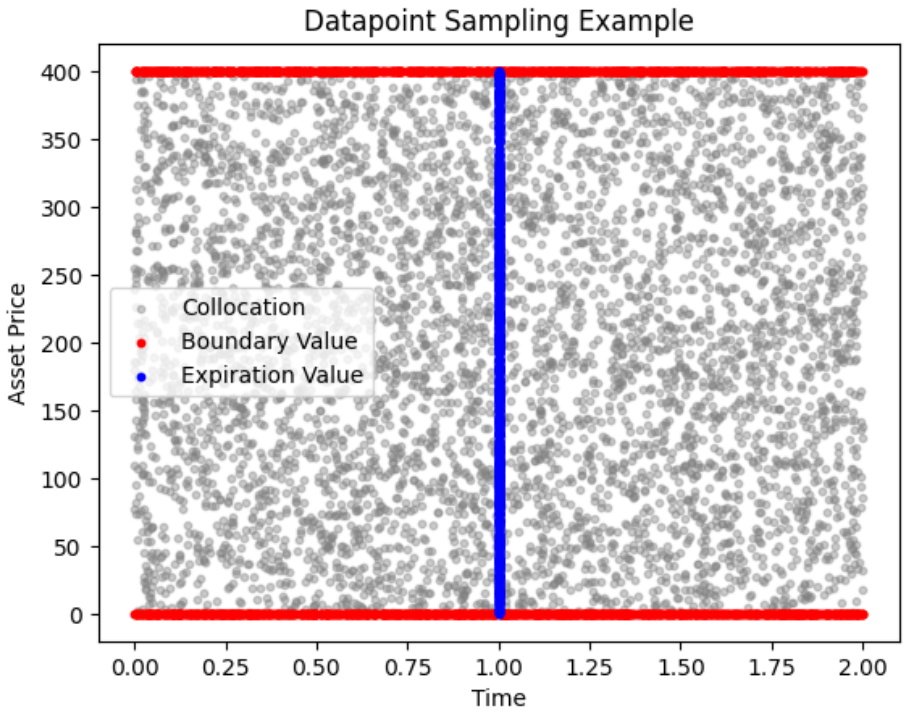

Fig. 3.1 Sampling Data-Points

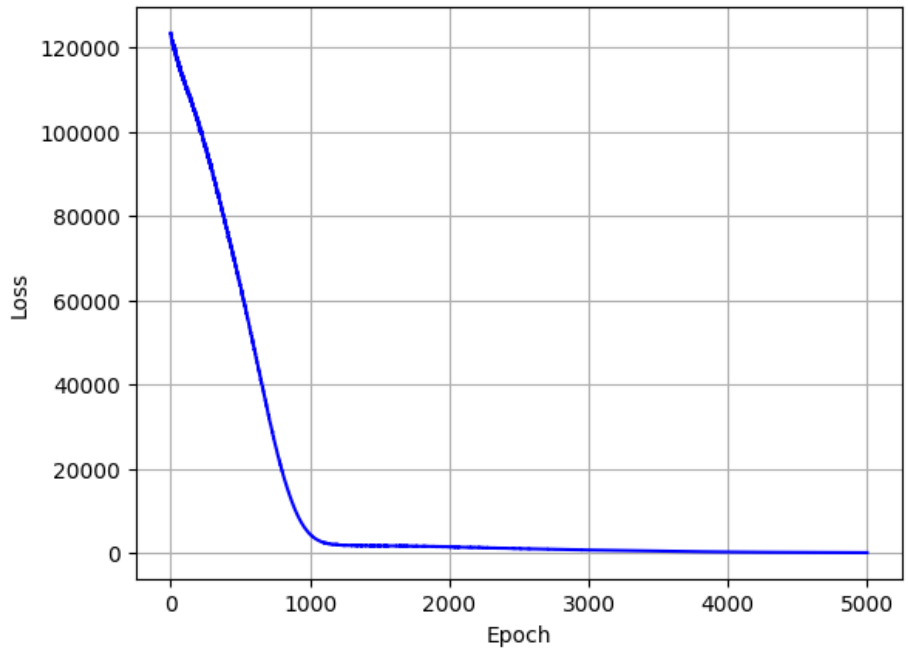

Fig. 3.2 Loss Function

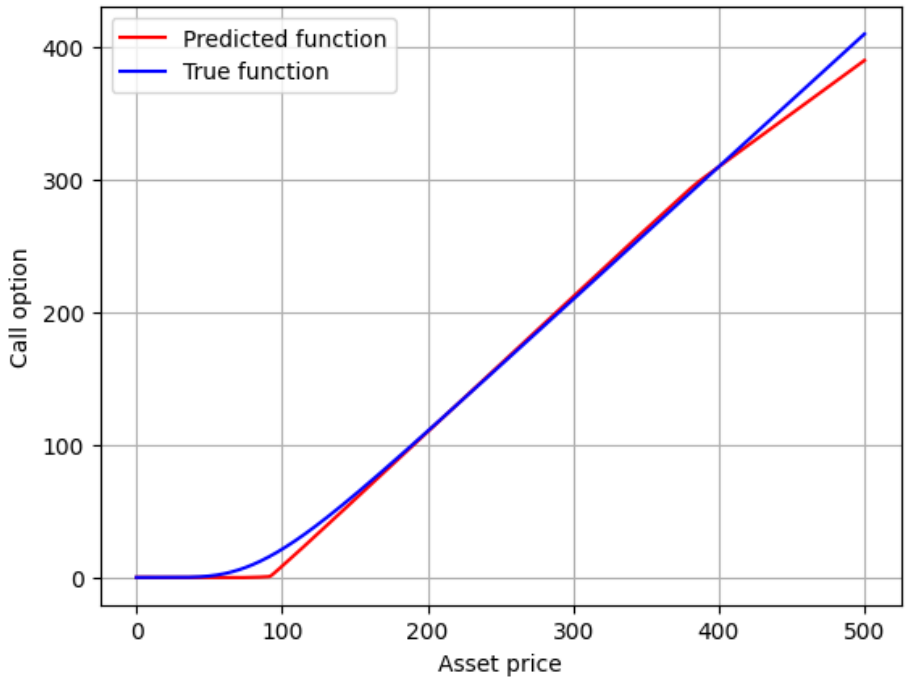

Fig. 3.3 Comparison between the PINN solution and the exact Black-Scholes solution for a European call option

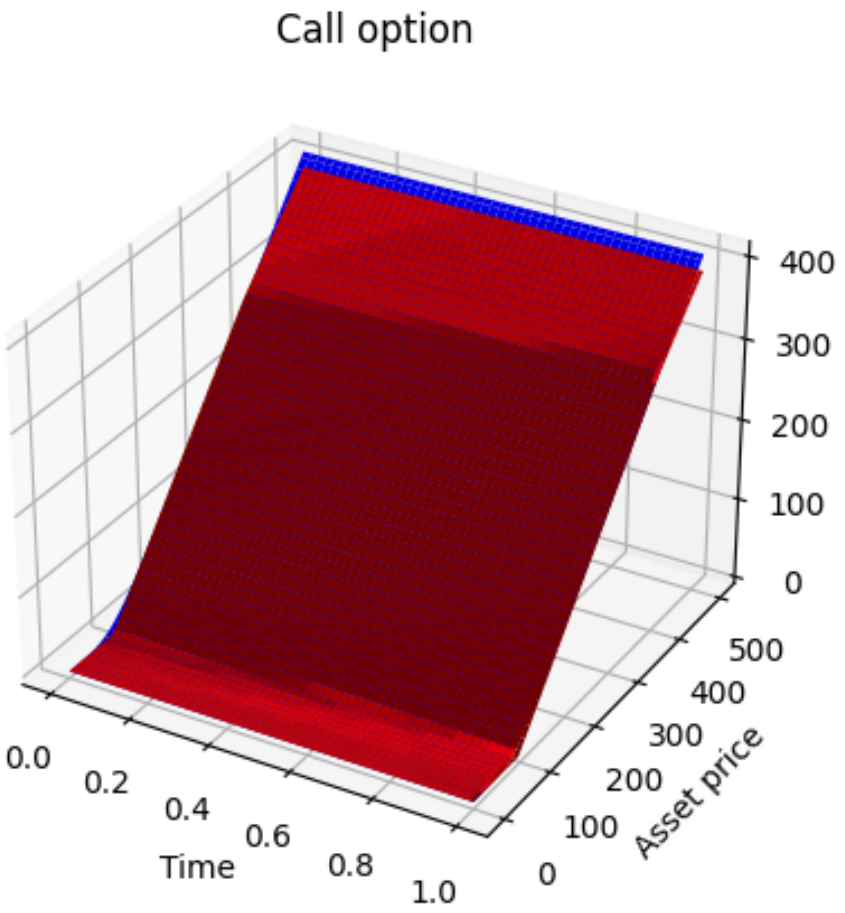

Fig. 3.4 Surface Graph

The PINN model was able to approximate the option prices with high accuracy, and the error in the predictions decreased as the network was trained for a longer duration. The results demonstrate that the PINN approach can effectively solve the Black-Scholes PDE and provide accurate pricing for European-style options.

3.7 Conclusion

In this chapter, we introduced the PINN framework to solve the Black-Scholes PDE leveraging the power of neural networks to learn the solution while incorporating the physics of the problem directly into the training process. The results show that PINNs can provide accurate solutions offering a promising alternative to traditional numerical methods.

Chapter 4

LSTM-enhanced PINN for Multi-asset PDEs

4.1 Introduction

In this chapter, we extend the **Physics-Informed Neural Network (PINN)** [2] framework to handle the complexities of **multi-asset option pricing**. While the original PINN formulation was applied to single-asset problems, financial markets often involve multiple assets whose prices are **interdependent**. To address this, we integrate **Long Short-Term Memory (LSTM)** [5] layers into the PINN framework, enabling the model to account for the **temporal dependencies** between multiple assets in the option pricing model.

This chapter demonstrates how the LSTM-enhanced PINN framework can be applied to multi-asset PDEs, specifically for option pricing involving multiple assets.

4.2 LSTM-enhanced PINN Framework

The key idea behind integrating LSTM layers into PINNs is to capture the **temporal dependencies** between the assets' prices. The LSTM network is well-suited for **time-series data**, making it ideal for modeling the relationship between multiple assets over

time.

4.2.1 Network Architecture

The architecture of the LSTM-enhanced PINN consists of the following:

- The input layer receives both the stock prices of multiple assets $S_1, S_2, \ldots, S_n$ and the time variable t.
- The LSTM layers process the temporal dependencies between the asset prices over time, capturing the dynamics of multiple assets in the market.
- The output layer predicts the option price based on the current asset prices and time.

4.2.2 Loss Function

The loss function consists of several components, which help enforce the desired properties in the network. Specifically, the total loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{Boundary}} + \mathcal{L}_{\text{Initial}} + \sum_{i=1}^{n} \sigma(S_i - S)$$

where:

- $\mathcal{L}_{\text{PDE}}$: This enforces the multi-asset version of the PDE, similar to the Black-Scholes equation but extended to handle multiple assets.
- $\mathcal{L}_{\text{Boundary}}$: This ensures that the neural network satisfies the boundary conditions for the multi-asset option pricing problem.
- $\mathcal{L}_{\text{Initial}}$: This enforces the initial conditions for the problem, often based on the payoff of the option at time $t = 0$.
- The term $\sum_{i=1}^{n} \sigma(S_i - S)$: This additional term ensures that the network minimizes the difference between the **predicted values of all assets** S_i and the **actual values** S over all layers **continuously**. This helps in adjusting the weights such that

the model better correlates the **interdependencies** between the asset prices, thus enhancing the prediction accuracy.

4.2.3 Training Process

The network is trained to minimize the total loss function using **backpropagation** and **gradient descent**. The LSTM layers allow the network to account for the temporal relationships between the assets, ensuring that the predictions reflect both the **instantaneous** and **dynamic changes** in asset prices over time. The loss function ensures the model accounts for both spatial (asset price) and temporal dependencies, allowing for improved predictions of multi-asset option prices.

4.3 Multi-asset Option Pricing

Multi-asset option pricing is a complex problem that involves modeling the correlation between the prices of multiple underlying assets. The LSTM-enhanced PINN framework can handle this complexity by processing the asset prices over time and learning the interdependencies between them.

4.3.1 Multi-asset Black-Scholes Equation

The multi-asset version of the Black-Scholes equation is given by:

$$\frac{\partial C}{\partial t} + \sum_{i=1}^{n} \frac{1}{2}\sigma_i^2 S_i^2 \frac{\partial^2 C}{\partial S_i^2} + \sum_{i=1}^{n} r S_i \frac{\partial C}{\partial S_i} - rC = 0$$

where C is the option price, S_i is the price of asset i, and σ_i and r are the volatility and risk-free rate for asset i, respectively.

4.3.2 Boundary and Initial Conditions

The boundary conditions for the multi-asset case are similar to those in the single-asset case, but extended to handle multiple assets. The initial condition is based on the payoff of the option at time $t = 0$, typically $\max(S_1 - K, 0)$ for a call option.

4.4 Implementation Details

The implementation of the DGM [18] framework for multi-asset PDEs is demonstrated in the following code notebooks: **Code6.ipynb**, **Code7.ipynb**, and **Code8.ipynb**.

The code shows how the LSTM-enhanced PINN can be trained for multi-asset option pricing. Visualizations comparing the results are included in the following figures.

4.5 Results

By integrating an LSTM layer, we were able to capture the temporal dependencies of the assets, helping to model their interdependencies. This was not possible with a feedforward neural network alone. Below are the results obtained using the LSTM-enhanced PINN for multi-asset option pricing.

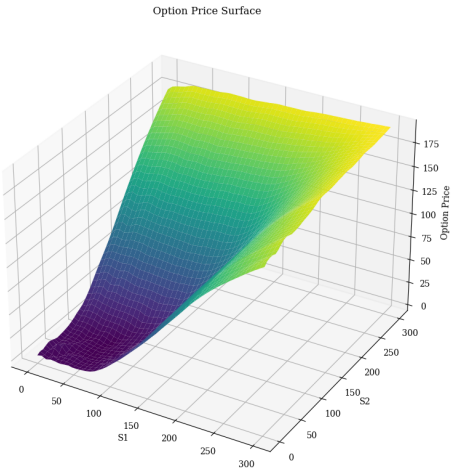

Fig. 4.1 Asset 1 and 2 vs Option Price (Fixed Time)

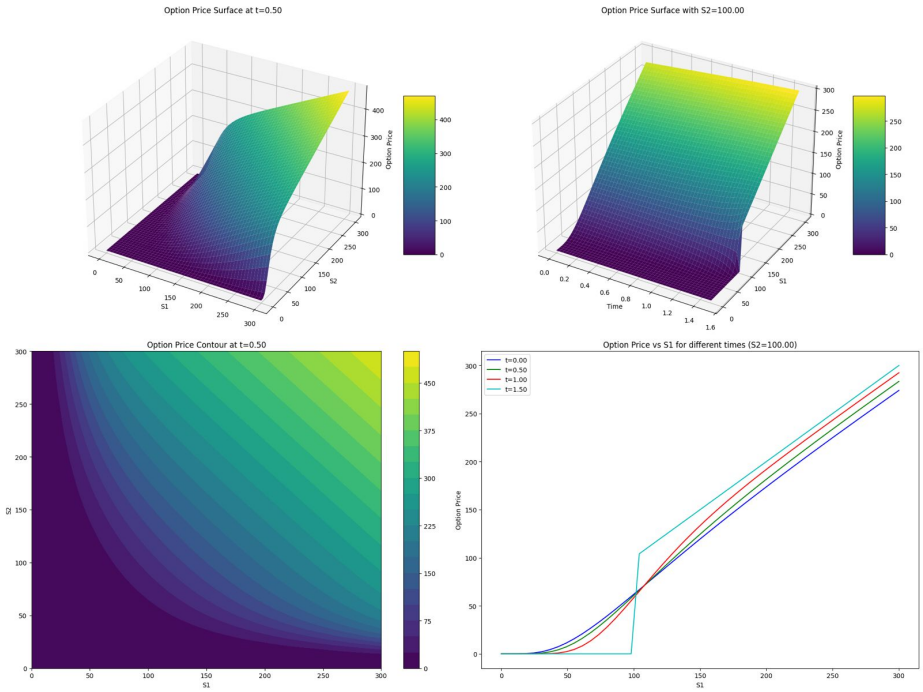

Fig. 4.2 Graphs with Boundary Conditions

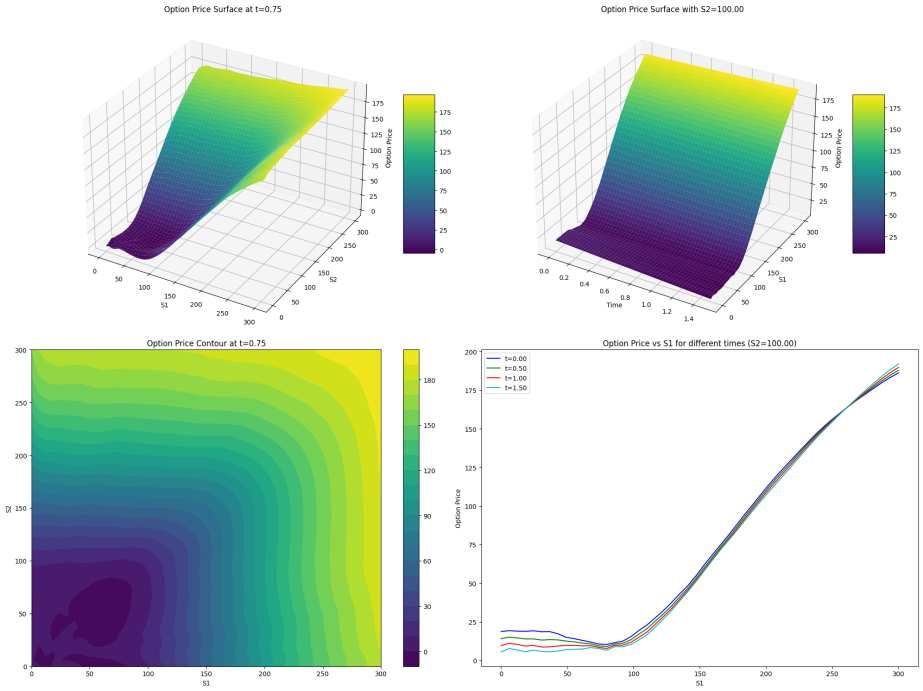

Fig. 4.3 Graphs without Boundary Conditions

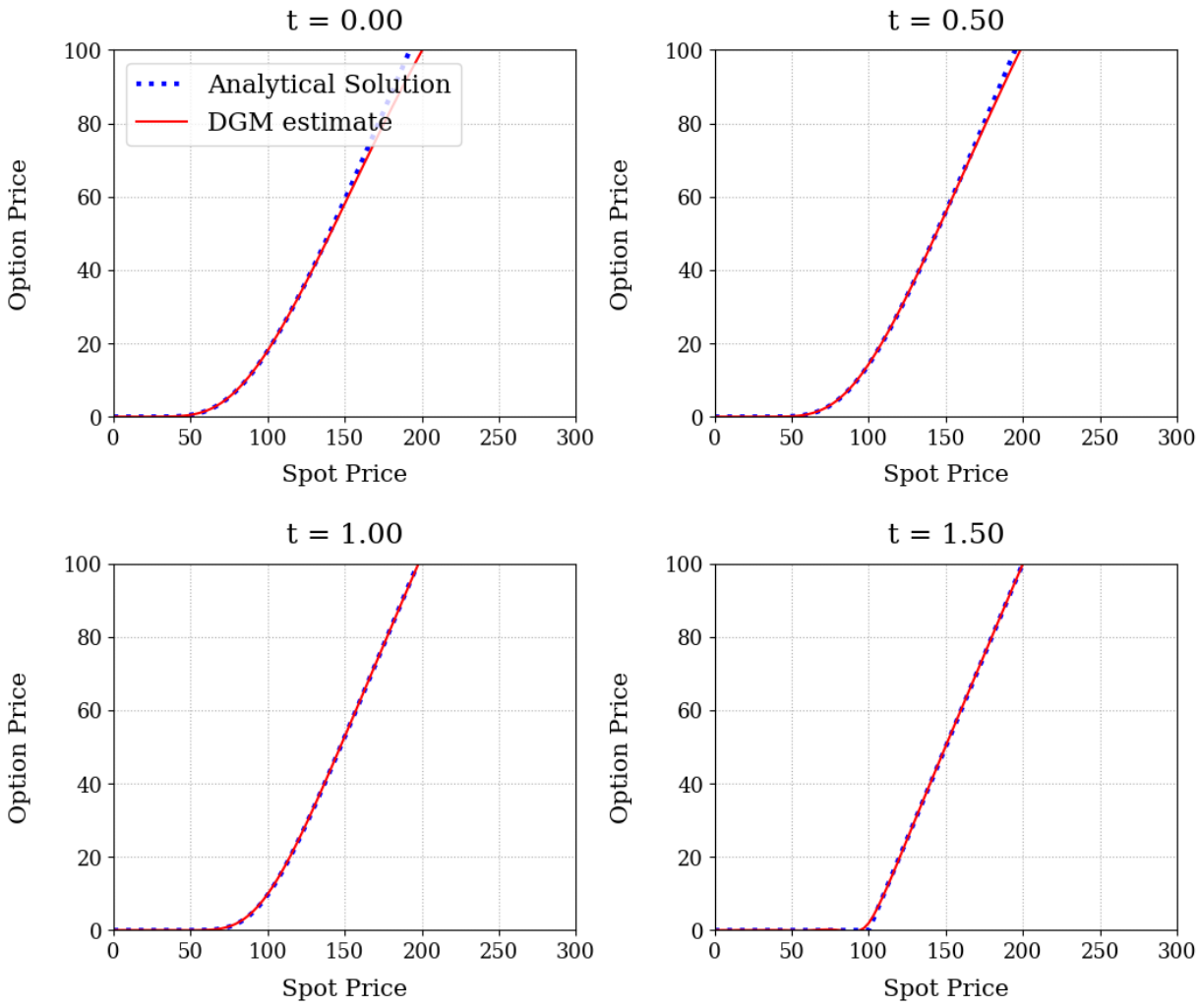

Fig. 4.4 Comparison of the Solutions (any asset)

4.6 Conclusion

In this chapter, we extended the Physics-Informed Neural Network (PINN) [2] framework by incorporating Long Short-Term Memory (LSTM) [5] layers to model the temporal dependencies in multi-asset option pricing. The LSTM-enhanced PINN framework enables the model to account for the interactions between multiple assets and predict option prices more accurately. The **additional loss function** term, $\sum_{i=1}^{n} \sigma(S_i - S)$, helps in minimizing the discrepancies between the predicted asset prices, thereby improving the model's predictive capability.

Chapter 5

Conclusion and Future Work

5.1 Summary of Results

This report presented a novel approach for solving the **Black-Scholes partial differential equation (PDE)** using deep learning techniques, particularly **Physics-Informed Neural Networks (PINNs)** [2]. The study began with applying PINNs to price **European-style options**, followed by extending the method to handle multi-asset options by incorporating **Long Short-Term Memory (LSTM)** [5] networks. The results demonstrate the potential of PINNs in providing accurate and efficient solutions compared to traditional numerical methods such as the **Finite Difference Method (FDM)** [7] and **Monte Carlo** [8] simulations.

The accuracy of the PINN-based models was validated through several experiments and benchmarks. In particular, the use of LSTM layers to address temporal dependencies in multi-asset pricing demonstrated improved performance over simpler approaches. The proposed PINN-DGM model provided solutions that were both computationally efficient and highly accurate, even in multi-dimensional settings, which are typically challenging for conventional numerical methods.

5.2 Comparative Analysis

A comparative study is presented between **closed-form geometric solutions** and solutions provided by the PINN-DGM model for different prices of the two assets S_1 and S_2. Results indicate that the PINN-DGM model closely matches the analytical solutions for various sampled points, showcasing its effectiveness in approximating the closed-form solution. Below is a table that compares the results for different asset prices, highlighting the performance of the PINN-DGM model versus the closed-form geometric solution.

Table 5.1 Comparison of Neural Network vs. Geometrical Solutions

S1	S2	Model	Analytical
135.53	102.11	37.29	37.25
162.23	18.92	61.36	59.49
175.85	283.37	185.96	170.15
3.40	55.55	0.43	0.51
137.63	233.53	139.82	124.05

5.3 Experimental Results

The experimental results are based on the implementation of the PINN-DGM model, where the neural network was trained to solve the Black-Scholes PDE for European options. The performance of the model was evaluated using a range of test cases, and the results were compared to traditional numerical methods such as the **Finite Difference Method (FDM)** [7] and **Monte Carlo** simulations. The comparison showed that the PINN-based approach significantly reduced computation time while maintaining high accuracy.

Several images and plots were generated to visualize the results. The following figures illustrate the error distribution of the model and other images/results generated by it.

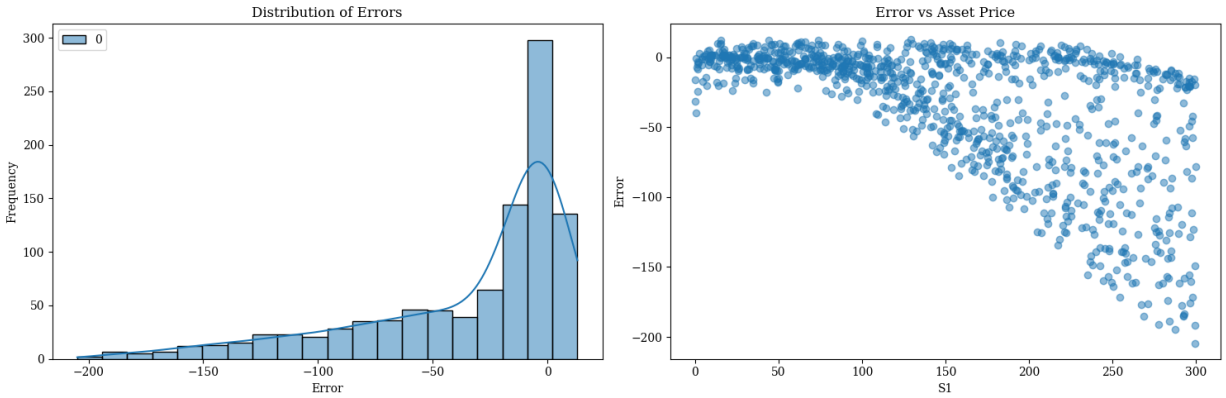

Fig. 5.1 Distribution and Comparison with Asset prices

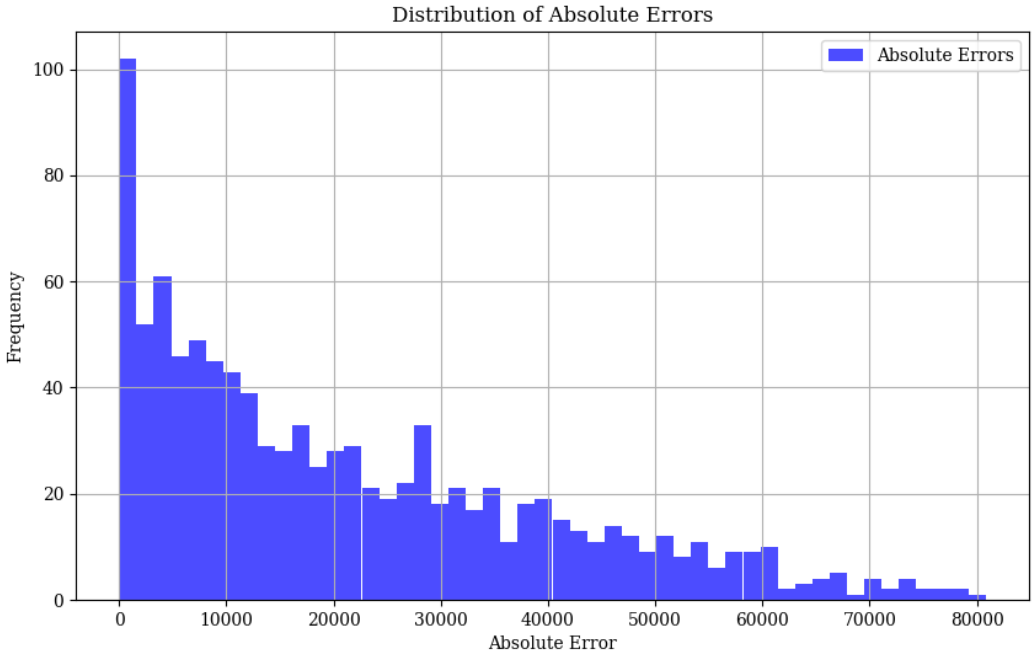

Fig. 5.2 Distribution of Absolute Errors

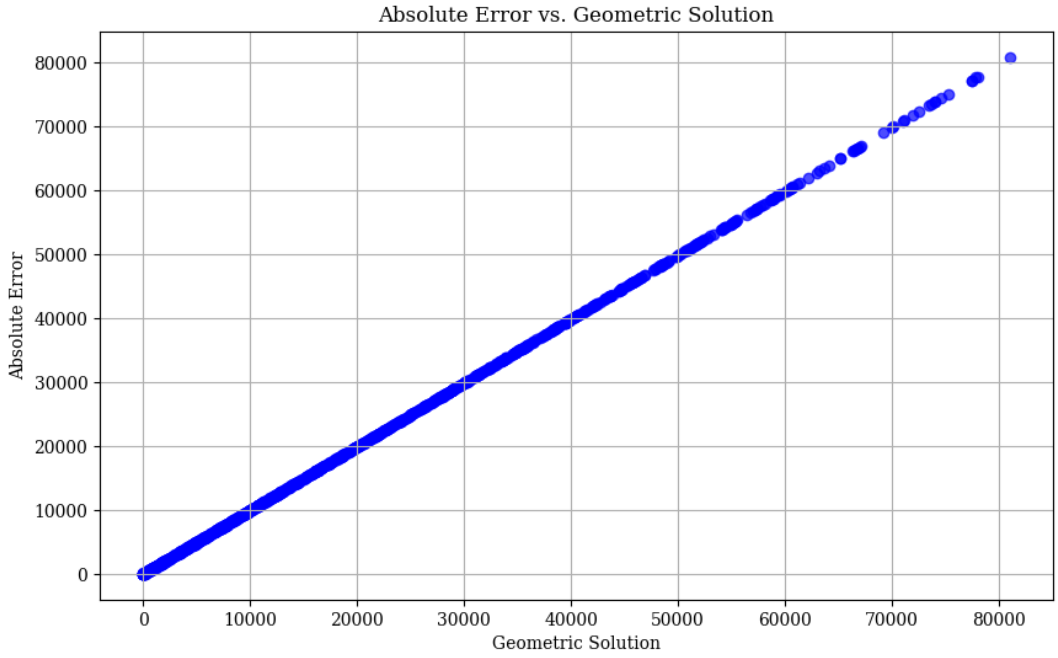

Fig. 5.3 Error vs Geometric Solution

5.4 Future Work

While the PINN-DGM model demonstrated promising results, there are several avenues for future research and improvement:

- **Improvement of Training Efficiency:** While the model performs well in terms of accuracy, the training time could be improved further, especially for larger datasets. Techniques such as **transfer learning** or more advanced optimization algorithms could be explored to speed up convergence.

- **Handling More Complex Models:** Future work can focus on expanding the current framework to handle more complex derivative instruments, such as **exotic options**, or **multi-factor models** in finance.

- **Real-time Pricing:** Incorporating real-time market data and implementing the

model in a production environment could be a valuable extension, allowing the model to perform dynamic pricing based on **live market conditions**.

- **Generalization to Other PDEs:** The principles of PINNs used for the Black-Scholes PDE can be extended to other important equations in financial mathematics, as well as in physics and engineering, where PDEs play a crucial role.

5.5 Conclusion

This study demonstrates the potential of using deep learning, particularly PINNs, for solving financial PDEs. The integration of LSTM networks with the **Deep Galerkin Method (DGM)** [6] offers a novel solution for pricing multi-asset options, addressing the **temporal dependencies** in the financial markets. Compared to traditional methods like FDM and **Monte Carlo** [8] simulations, the PINN-DGM model offers superior efficiency and accuracy, especially in high-dimensional settings. The comparative analysis shows that the model closely matches the closed-form geometric solutions, validating its effectiveness.

In conclusion, this approach paves the way for more scalable and efficient techniques in financial modeling, with potential applications across a wide range of domains requiring the solution of complex PDEs.

References

[1] Wikipedia, "Valuation of options," 2023. https://en.wikipedia.org/wiki/Valuation_of_options.

[2] G. E. K. M. Raissi, P. Perdikaris, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving partial differential equations," 2019. https://arxiv.org/abs/1711.10561.

[3] A. Hayes, "Black-scholes model: What it is, how it works, and options formula," 2024. https://www.investopedia.com/terms/b/blackscholes.asp.

[4] H. J. Ricardo, "Introduction to differential equations," *A Modern Introduction to Differential Equations (Third Edition)*, 2021. https://www.sciencedirect.com/topics/mathematics/partial-differential-equation.

[5] Colah, "Understanding lstm networks," 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[6] X. Ji, "Solving nonlinear partial differential equations via deep galerkin method," 2022. https://iopscience.iop.org/article/10.1088/1742-6596/2381/1/012047/pdf.

[7] T. W. Kerlin and B. R. Upadhyaya, "Finite difference method," *Dynamics and Control of Nuclear Reactors*, 2019. https://www.sciencedirect.com/topics/engineering/finite-difference-method.

[8] W. Kenton, "Monte carlo simulation: What it is, how it works, history, 4 key steps," 2024. https://www.investopedia.com/terms/m/montecarlosimulation.asp.

[9] A. Ganti, "Option pricing theory: Definition, history, models, and goals," 2024. https://www.investopedia.com/terms/o/optionpricingtheory.asp.

[10] N. Morita, "Finite element methods," *Finite Element Programming in Nonlinear Geomechanics and Transient Flow*, 2021. https://www.sciencedirect.com/topics/engineering/finite-element-method.

[11] A. Hayes, "Volatility: Meaning in finance and how it works with stocks," 2024. https://www.investopedia.com/terms/v/volatility.asp.

[12] A. Hayes, "What is the risk-free rate of return, and does it really exist?," 2024. https://www.investopedia.com/terms/r/risk-freerate.asp.

[13] A. Ganti, "Expiration time explained: What it is, how it works, example," 2022. https://www.investopedia.com/terms/e/expiration-time.asp.

[14] J. Chen, "Underlying asset (derivatives)—definition, how it works, examples," 2024. https://www.investopedia.com/terms/u/underlying-asset.asp.

[15] E. Blessing, "Maturity: Definition, how maturity dates are used, and examples," 2024. https://www.investopedia.com/terms/m/maturity.asp.

[16] J. Fernando, "Option strike prices: How it works, definition, and example," 2023. https://www.investopedia.com/terms/s/strikeprice.asp.

[17] K. Lee, "Solving black-scholes equation with pinns," 2024. https://github.com/keonly/black-scholes-pinn.

[18] A. Al-Aradi, "Implementation of the deep galerkin method," 2024. https://github.com/alialaradi/DeepGalerkinMethod.