Computational Financial Mathematics: Methods and Applications

Mohammed Moniruzzaman khan

February 3, 2025

Abstract

This research project explores foundational concepts in financial mathematics through the lens of computational finance, emphasizing the application of advanced financial modeling and simulation techniques. It integrates theoretical knowledge with practical problem-solving by employing diverse methodologies, including discrete-time and continuous-time modeling, partial differential equations, Monte Carlo simulations, and sensitivity analysis. Key topics include the binomial pricing model, the Black-Scholes framework, numerical solutions for option pricing. Special attention is given to computational techniques for solving real-world problems, such as pricing European and American options. With combining mathematical rigor with computational tools, this study provides a systematic approach to understanding and addressing challenges in modern financial markets, highlighting the interplay between theoretical models and their practical applications.

Contents

Chapter 1

Introduction

In the complex and ever-evolving landscape of global financial systems, the accurate quantification, pricing, and hedging of financial risks are pivotal for maintaining economic stability and facilitating sustained growth. The discipline of Computational Financial Mathematics (CFM), an interdisciplinary fusion of advanced mathematical theory, stochastic processes, and cutting-edge computational techniques, has revolutionized financial markets. This field has grown in importance as financial systems become increasingly sophisticated and interconnected, providing the essential tools for tackling the multifaceted challenges of modern finance. This study aims to explore and critically evaluate the methodologies and tools that underpin derivative pricing, risk management, and decision-making, building upon the pioneering work of Wilmott [26] and other foundational contributions to quantitative finance [2, 6].

The principal objective of this research is to investigate the practical applications of advanced financial models, with an emphasis on numerical implementation. By examining a suite of core methodologies—specifically the Black-Scholes model, binomial trees, Monte Carlo simulations, and finite difference methods for solving partial differential equations (PDEs)—this study bridges theoretical constructs with computational experimentation. Each of these methodologies offers unique strengths in modeling market behavior, pricing derivatives, and assessing sensitivities (Greeks), thereby providing a robust framework for tackling the complex problems faced by financial practitioners.

The motivation for this investigation stems from the increasing complexity and volatility of contemporary financial markets, where the need for precise and rapid computational solutions has never been more critical. The scale and intricacy of financial instruments, particularly derivatives valued in trillions of dollars, require sophisticated mathematical models capable of capturing the stochastic nature of the markets. Derivatives play a pivotal role in hedging, speculation, and arbitrage, and their fair pricing and effective risk management demand models that accurately reflect market uncertainty and dynamic evolution[2, 17]. This research seeks to explore these challenges through a synthesis of theoretical derivations, algorithm development, and empirical validation.

The methodology adopted in this study unfolds in two primary dimensions:

Mathematical Foundations: The theoretical exploration of pricing models such as the Black-Scholes equation, binomial tree frameworks, and stochastic simulations. Numerical Implementation: The development of computational solutions to demonstrate the

practical applications of these models and to critically assess their computational efficiency and complexity.

The Black-Scholes model, regarded as a cornerstone of financial mathematics, provides elegant closed-form solutions for European option pricing, but its reliance on simplifying assumptions—such as constant volatility and the absence of early exercise—limits its applicability in real-world scenarios [2]. To overcome these limitations, binomial tree models are considered as discrete-time alternatives, offering enhanced flexibility for pricing American options and modeling early exercise behavior [6]. Monte Carlo simulations, widely acknowledged for their versatility in path-dependent option pricing, allow for a nuanced understanding of the probabilistic nature of market movements [12]. Additionally, finite difference methods, by numerically solving PDEs, offer a comprehensive approach to pricing options, emphasizing the trade-offs between computational accuracy and efficiency [27].

The outputs of this research include not only derivative prices and sensitivity measures (Greeks) but also a detailed evaluation of the computational trade-offs inherent in each methodology. These results are validated against historical market data to ensure their fidelity and relevance to real-world market dynamics. For instance, the Greeks provide vital insights into portfolio management, while Monte Carlo simulations offer probabilistic views on market uncertainty, enabling practitioners to make informed decisions in the face of risk.

From a computational standpoint, the complexity analysis of each method emphasizes their suitability for high-frequency trading environments, where the precision and speed of calculations are paramount. The research underscores the delicate balance between computational efficiency and model accuracy, which is crucial in guiding financial professionals in choosing the most appropriate methodology for various financial instruments and scenarios.

This study holds substantial significance for both academia and industry. Traders can leverage the findings to craft robust hedging strategies, while financial institutions can utilize the insights to refine risk management practices and gain a competitive edge in an increasingly data-driven market. Moreover, policymakers can apply the research outcomes to enhance regulatory frameworks and mitigate systemic risks in the financial sector. By addressing the computational challenges of modern finance, this research highlights the vital role of high-performance computing in advancing financial innovation.

In conclusion, this study exemplifies the multidisciplinary nature of Computational Financial Mathematics, where the convergence of mathematics, computation, and finance provides a comprehensive solution to the intricate challenges of derivative pricing and risk management. By analyzing the interactions between theoretical models, numerical techniques, and their real-world applications, this research provides a structured framework for advancing computational finance, as envisioned by Wilmott [26] and other leading scholars in the field.

Chapter 2

Financial Mathematics

In this chapter, we explore fundamental concepts in financial mathematics that are essential for understanding financial markets and instruments. We will cover equities, commodities, forwards, futures, and options, delving into their pricing mechanisms and market implications. The concepts discussed will draw upon established theories and models, including those presented by Wilmott (see [26]), which provide a comprehensive framework for analyzing these financial instruments.

2.0.1 Equities

Equities represent ownership in a company, commonly known as stocks. When an individual purchases shares of a company, they acquire a claim on part of the company's assets and earnings. The key aspects of equities include:

Stock Pricing

The price of a stock is influenced by various factors, including:

- **Supply and Demand:** Stock prices rise when demand exceeds supply and fall when supply exceeds demand.
- **Fundamental Analysis:** Evaluating a company's financial statements, management, market position, and economic conditions to estimate its intrinsic value.
- **Technical Analysis:** Using historical price data and trading volume to predict future price movements.

The mathematical representation of stock pricing can be modeled using the Gordon Growth Model (GGM) for dividends:

$$P = \frac{D_1}{r - g}$$

where:

- P = current stock price.

- g = constant growth rate expected for dividends, in perpetuity.

- r = constant cost of equity capital for the company (or rate of return).

- D_1 = value of next year's dividends.

Market Implications

Investors can use equities to generate capital gains and dividends. However, equities are subject to market volatility, economic changes, and company performance, making them riskier than fixed-income securities.

2.0.2 Commodities

Commodities are basic goods used in commerce that are interchangeable with other goods of the same type. They can be categorized into two main types: hard commodities (natural resources like oil and gold) and soft commodities (agricultural products like wheat and coffee).

Commodity Trading

The trading of commodities occurs through exchanges where standardized contracts are bought and sold. The pricing of commodities is influenced by:

- **Supply and Demand Dynamics:** Fluctuations in supply due to weather conditions, geopolitical events, or changes in consumer demand can significantly affect prices.
- **Market Speculation:** Traders often speculate on future price movements, impacting current prices.

Mathematical Framework

The pricing of commodities can be modeled using the cost of carry model, which includes storage costs, interest rates, and convenience yield:

$$F_t = S_t \cdot e^{(r+u-c)(T-t)}$$

where:

- F_t = futures price at time t

- S_t = spot price at time t

- r = risk-free interest rate

- u = convenience yield

- c = storage costs

- T = expiration time of the futures contract

Market Implications

Commodity investments can hedge against inflation and market volatility. However, they also carry risks related to price fluctuations and geopolitical factors.

2.0.3 Forwards and Futures

Forwards and futures are contracts that obligate the buyer to purchase, and the seller to sell, an asset at a predetermined future date and price.

Forwards

A forward contract is a customized agreement between two parties to buy or sell an asset at a specified future date for a price agreed upon today. The key characteristics include:

- **Customization:** Terms can be tailored to the needs of the parties involved.
- **Counterparty Risk:** Since forwards are OTC contracts, they are subject to credit risk.

Futures

A futures contract is similar to a forward contract but is standardized and traded on exchanges. Key features include:

- **Standardization:** Futures contracts have standardized terms, including contract size and expiration dates.
- **Margin Requirements:** Futures require margin payments, reducing counterparty risk.

Mathematical Pricing

The pricing of futures contracts can be derived from the cost of carry model, similar to commodities:

$$F = S_0 e^{rT}$$

where F is the futures price, S_0 is the current spot price, r is the risk-free rate, and T is the time to maturity in years.

Market Implications

Forwards and futures are primarily used for hedging and speculation. They allow parties to lock in prices and mitigate risks associated with price fluctuations.

2.0.4 Options

Options are financial derivatives that provide the holder the right, but not the obligation, to buy or sell an asset at a predetermined price before or at expiration.

Types of Options

- **Call Options:** Provide the right to buy the underlying asset.
- **Put Options:** Provide the right to sell the underlying asset.

Option Pricing Models

The pricing of options can be complex and is typically modeled using various methods, such as:

- **Black-Scholes Model:** A widely-used model for pricing European call and put options.
- **Binomial Model:** A discrete-time model for pricing options by simulating possible paths of the underlying asset's price.

Chapter 3

Discrete-Time Modeling

In financial mathematics, accurately pricing options is critical for understanding the behavior of financial markets and making informed investment decisions. The Binomial Pricing Model, first introduced by Cox, Ross, and Rubinstein in 1979, is one of the foundational techniques in option pricing. This model provides a discrete-time framework for simulating the movement of asset prices and calculating the value of financial derivatives, such as options, over time.

The Binomial model is particularly valuable due to its simplicity and versatility. It allows for the modeling of various types of options, including European and American-style options, by discretizing the time horizon into multiple intervals and simulating upward or downward movements in the underlying asset's price. The core idea behind the model is to break down the uncertainty of future price movements into a series of steps, enabling the calculation of option prices through a process known as backward induction.

In this chapter, we will delve into the workings of the Binomial model, starting with its theoretical foundations. We will then move on to its implementation in Python, with a focus on pricing European options. Through this implementation, we aim to provide a hands-on understanding of the model and its application in real-world financial scenarios.

The chapter begins by outlining the key assumptions and components of the Binomial Pricing Model, followed by a detailed explanation of the steps involved in pricing options. The goal is to not only demonstrate the theoretical aspects of the model but also to showcase its practical implementation and the results obtained for different types of options. By the end of this chapter, the reader will have gained both a solid theoretical grounding in the Binomial model and practical experience through Python-based simulations.

3.1 The Binomial Pricing Model

The **Binomial Pricing Model** is a discrete-time method used to model the evolution of an underlying asset's price over time, facilitating the valuation of financial options. The model operates on the assumption that the price of the asset can either increase or decrease at each time step, forming a binomial tree structure [2]. The basic assumptions and formulae underlying the Binomial model are as follows:

- **Asset Price Movements**: At each time step, the asset price can move **up** by

a multiplicative factor u or **down** by a multiplicative factor d. Here, $u > 1$ and $0 < d < 1$, ensuring that the price only moves upward or downward but never stays constant.

- **Probabilities**: The probability of an upward movement at each step is denoted by p, while the probability of a downward movement is $1 - p$.

The model uses these assumptions to construct a binomial tree, where each node represents a potential price of the asset at a given time. The asset price at a given time t and at a particular node in the tree is expressed as:

$$S_{n,j} = S_0 \cdot u^j \cdot d^{n-j}$$

Where:

- S_0 is the initial price of the asset at time $t = 0$.
- n is the total number of time steps, or periods, over which the asset price is modeled.
- j is the number of upward price movements observed after n time steps. Hence, the number of downward movements is $(n - j)$.

This recursive structure allows the model to simulate various possible future paths of the asset price, which in turn provides a basis for calculating the option's payoff and its present value using backward induction [5, 2].

3.1.1 Pricing European Options Using the Binomial Model

To price European options using the Binomial model, we follow these steps:

- Construct the Price Tree: Build a binomial tree for the underlying asset price over the desired time steps.
- Calculate Option Payoffs: At maturity (the final nodes of the tree), calculate the option payoffs:

$$C_j = \max(S_{n,j} - K, 0) \quad \text{(for call options)}$$
$$P_j = \max(K - S_{n,j}, 0) \quad \text{(for put options)}$$

  where K is the strike price.

- Backward Induction: To calculate the option price at each node, we use **backward induction**, a method where we begin at the option's expiration date and work backward in time to the present. At each node, we calculate the option price by considering the potential future outcomes and discounting them to the present, reflecting the concept of **time value of money**. The prices are determined using the following formulas:

$$C_{i,j} = e^{-r\Delta t} \left(pC_{i+1,j+1} + (1-p)C_{i+1,j}\right)$$

$$P_{i,j} = e^{-r\Delta t} \left(pP_{i+1,j+1} + (1-p)P_{i+1,j}\right)$$

In these formulas, $C_{i,j}$ and $P_{i,j}$ represent the option prices at node (i, j), while r denotes the **risk-free interest rate**, which reflects the return rate on a risk-free investment. Δt represents the length of each time step. The values $C_{i+1,j+1}$, $C_{i+1,j}$, $P_{i+1,j+1}$, and $P_{i+1,j}$ represent the option prices at the next time step, corresponding to upward and downward price movements.

The term p is the **probability of an upward movement** in the asset price at each step. It is determined using the formula:

$$p = \frac{e^{(r\Delta t)} - d}{u - d}$$

where u is the factor by which the asset price increases (the up factor), and d is the factor by which it decreases (the down factor). This probability represents the likelihood that the asset price will move upward in each time step, given the parameters of the model.

The term $e^{-r\Delta t}$ is the **discount factor**, which adjusts future payoffs by discounting them to their present value. This reflects the **time value of money** principle, which asserts that money available today is worth more than the same amount in the future. By multiplying the expected future payoffs by $e^{-r\Delta t}$, we account for the reduction in value over time due to the risk-free interest rate r. A higher rate r results in a greater discount, meaning future payoffs are worth less today [6].

This process of backward induction continues from the expiration date to the present, ensuring that the option price is the **present value** of the expected future payoffs, appropriately adjusted for both the uncertainty of future price movements and the time value of money.

In this section, we analyze the results of pricing European options using the binomial model. The binomial model is a discrete-time method where we simulate the price evolution of the underlying asset over time, considering possible upward and downward movements at each step. The method involves constructing a binomial price tree, calculating the option payoffs at maturity, and then using backward induction to determine the present value of the option.

3.2 Function Descriptions for Binomial Model Implementation

This section provides an overview of the key functions used in the binomial model implementation for European option pricing. Each function is described with its role in the computational process.

This section describes the functions used in the binomial model implementation for European option pricing.

- **binomial_tree(S_0, K, T, r, σ, n, option_type):** This function implements the binomial tree method for European option pricing using backward induction. It computes the option price based on the input parameters:

  - S_0: Initial asset price
  - `K`: Strike price
  - `T`: Time to maturity
  - `r`: Risk-free interest rate
  - σ: Volatility
  - `n`: Number of time steps
  - `option_type`: Option type ("call" or "put")
- **calculate_binomial_parameters(T, r, sigma, n):** Computes essential parameters for the binomial model:
  - `dt`: Time step size
  - `u`: Up factor
  - `d`: Down factor
  - `p`: Risk-neutral probability
- **construct_asset_price_tree(S_0, u, d, n):** Builds a binomial tree to calculate asset prices at each node over the time steps.
- **compute_option_payoff(S_T, K, n, option_type):** Calculates the option payoff at maturity. It depends on whether the option is a call or a put.
- **backward_induction(option_tree, p, r, dt, n):** Performs backward induction to compute option values at earlier time steps by discounting future payoffs.
- **price_option(S_0, K, T, r, σ, n, option_type):** This is the main function that integrates all the previous functions to calculate and return the final option price and the variables are also defined as :
- **Initial asset price**: $S_0 = 100$
- **Volatility**: $\sigma = 20\%$
- **Risk-free interest rate**: $r = 5\%$
- **Strike price**: $K = 100$
- **Time to expiration**: $T = 1$ year
- **Number of steps**: $N = 100$

From the binomial model simulation (Appendix : A.2), we get the following results for the American options:

- **European Call Option Price**: 10.43
- **European Put Option Price**: 5.55

The two graphs below show the price evolution of the European call and put options under the binomial model. These graphs represent the pricing of the options at different points in time, from the start of the option's life until maturity.

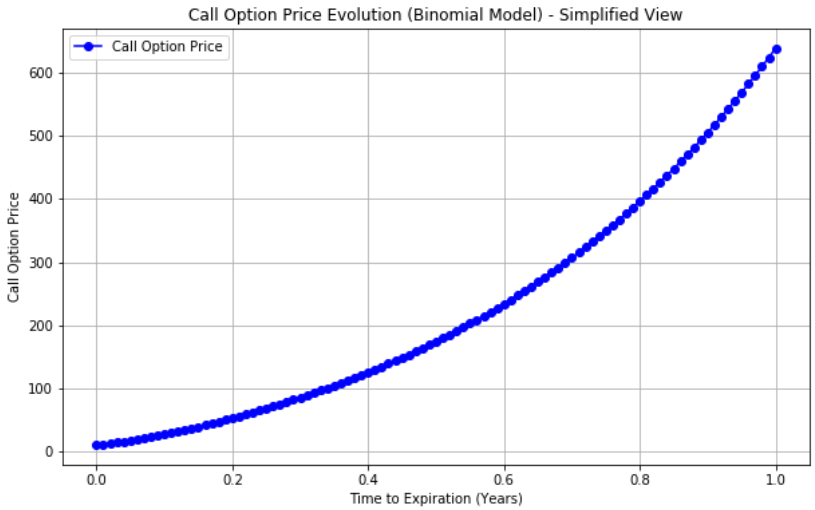

Figure 3.1: European Call Option Price Evolution

The first graph (Figure 3.1)displays the price of the European call option over time. The call option gives the holder the right to buy the underlying asset at a specified strike price (100). As we can observe from the graph, the price of the European call option increases over time as the underlying asset price rises. This happens because, as the asset price increases, the probability of the call option being exercised (i.e., the asset price exceeding the strike price) also increases. This results in a higher value for the call option. The price of the call option reaches its maximum at maturity when the underlying asset price is significantly higher than the strike price.

- European Call Option Price: 10.43

This outcome reflects market conditions where there is an upward trend in the price of the underlying asset. As the option approaches expiration, the possibility of exercising the call option increases, which in turn increases its price.

The second graph (Figure 3.2) shows the price of the European put option over time. The put option gives the holder the right to sell the underlying asset at the strike price (100). From the graph, we see that the price of the put option decreases over time. As the underlying asset price increases, the likelihood of the put option being exercised (i.e., the asset price falling below the strike price) decreases. This leads to a lower value for the put option. Since the option holder is less likely to exercise the option, the price of the put option gradually decreases as the asset price rises.

- European Put Option Price: 5.55

The price of the put option is lower because the market conditions favor upward price movements in the underlying asset, which makes the option less likely to be exercised. As the expiration date approaches, the put option's price continues to decrease due to the diminishing likelihood of the asset price falling below the strike price.

The results indicate that, under the current market conditions (with a risk-free interest rate of 5

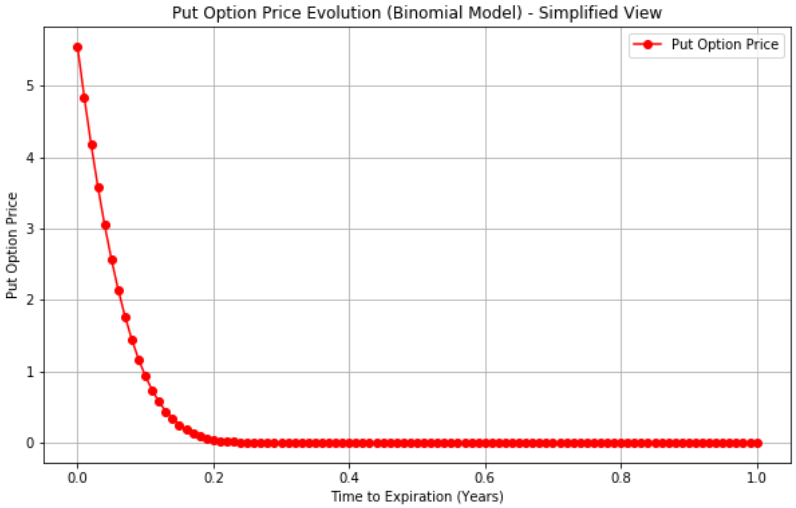

Figure 3.2: European Put Option Price Evolution

The call option price (10.43) reflects the benefit of having the right to buy the asset at a fixed price when the market price is expected to be higher. The put option price (5.55) is lower because the price of the underlying asset is not expected to fall significantly below the strike price, so the likelihood of exercising the put option is reduced.

The accuracy of the binomial model depends on the number of time steps used. In this case, we used 100 time steps, which is generally sufficient for obtaining reliable results. As the number of time steps increases, the model's approximation becomes more accurate, and the results converge toward the values obtained from continuous-time models, such as the Black-Scholes model.

In the context of these graphs and prices, the simulated prices of 10.43 for the European call and 5.55 for the European put are consistent with theoretical expectations based on the model parameters and market conditions.

The results demonstrate the expected behavior of the European call and put options in a market where the underlying asset price is more likely to rise:

- The call option is priced higher (10.43) due to the expectation that the asset price will exceed the strike price, making the call option more valuable.
- The put option is priced lower (5.55) because the likelihood of the asset price falling below the strike price is relatively low in the given market conditions.

These results highlight the power of the binomial model to simulate the price evolution of options in a discrete time framework, and the graphs visually demonstrate how the option prices evolve over time.

3.3 Pricing American Options Using the Binomial Model

In this section, we explore the pricing of American options using the binomial model. We construct a binomial tree for the underlying asset and use backward induction to determine the option prices. For American options, the flexibility to exercise early plays a significant role in determining the price at each node. This is different from European options, which can only be exercised at expiration [1, 6].

We calculate the prices of both the American call and American put options using the backward induction method. The final option price is determined by comparing the payoff of early exercise with the value of holding the option to maturity. The formula for pricing American options at each node is:

American Call Option Pricing:

$$C_{i,j} = \max\left(S_{i,j} - K, e^{-r\Delta t}\left(pC_{i+1,j+1} + (1-p)C_{i+1,j}\right)\right)$$

American Put Option Pricing:

$$P_{i,j} = \max\left(K - S_{i,j}, e^{-r\Delta t}\left(pP_{i+1,j+1} + (1-p)P_{i+1,j}\right)\right)$$

where:

- $C_{i,j}$ and $P_{i,j}$ are the **American call and put option** prices at node (i, j),
- $S_{i,j}$ is the **underlying asset price** at the node,
- K is the **strike price**,
- r is the **risk-free interest rate**,
- Δt is the **time step size**, and
- p is the **risk-neutral probability**.

3.3.1 Binomial Tree Algorithm and function for American Option

The function `american_option(`S_0`, K, T, r, `σ`, n, option_type="call or put")` implements the binomial tree model for pricing American options. It computes the option price through backward induction, considering early exercise. The function takes parameters such as the initial asset price S_0, strike price K, time to expiration T, risk-free rate r, volatility σ, and number of time steps n. The `option_type` argument determines whether the option is a call or put. The function returns the option price tree (`option_tree`) and the asset price tree (S_T).

The calculation begins with determining the up (u) and down (d) factors, followed by the construction of the asset price tree. The final option values are set at the maturity nodes based on the option's payoff. Using backward induction, the function evaluates the price at earlier nodes by comparing the payoff from early exercise with the discounted expected future value of holding the option. The option price at the root node is returned as the option's price.

The American option price at each node is determined by:

$$C_{i,j} = \max(S_{i,j} - K, e^{-r\Delta t}\left(pC_{i+1,j+1} + (1-p)C_{i+1,j}\right))$$

where $C_{i,j}$ is the option price at node (i, j), $S_{i,j}$ is the asset price, K is the strike price, r is the risk-free rate, Δt is the time step, and p is the risk-neutral probability.

The function uses `np.exp()` to compute exponential terms and `np.zeros()` to initialize matrices for asset and option prices. Option prices are visualized with `plt.plot()` to show how the option value evolves, particularly highlighting early exercise effects.

We implement the **binomial model** with the following parameters:

- **Initial asset price**: $S_0 = 100$
- **Volatility**: $\sigma = 20\%$
- **Risk-free interest rate**: $r = 5\%$
- **Strike price**: $K = 100$
- **Time to expiration**: $T = 1$ year
- **Number of steps**: $N = 100$

The computed **American option prices** from the binomial model (**Appendix A.3 for implementation**) are:

- **American Call Option Price**: **10.43**
- **American Put Option Price**: **6.08**

To better understand these results, we analyze the behavior of American option prices over the binomial tree. In the graph for the American call option, we observe that the price remains constant at 10.43 across all nodes in the tree. This implies that there is no advantage to early exercise, as the underlying asset price is expected to stay above the strike price. The price of the American call option behaves identically to the European call option in this scenario, indicating that early exercise is not optimal under these conditions.

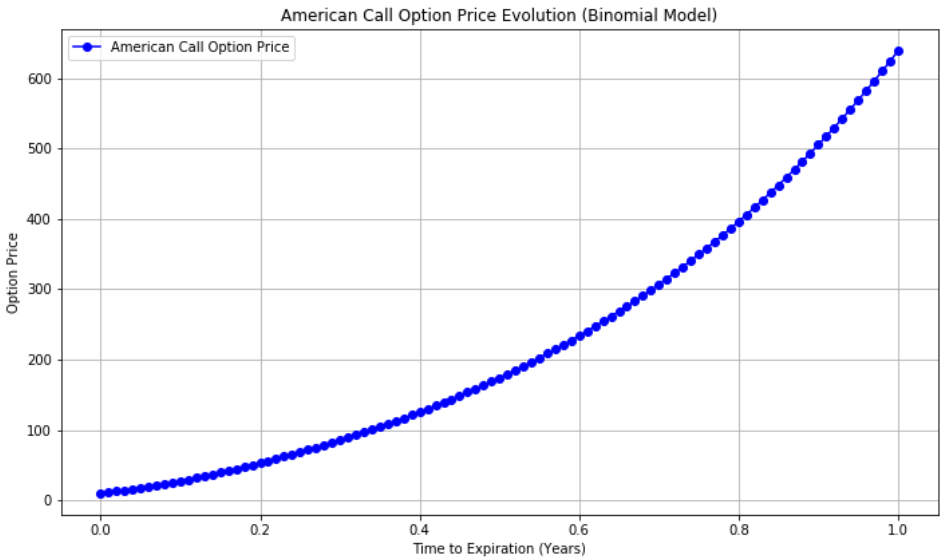

Figure 3.3: American Call Option Pricing

The computed American put option price is 6.08, which is higher than the European put price of 5.55. This difference arises because early exercise can be optimal when the underlying asset price falls significantly below the strike price. When the asset price is low, the immediate exercise payoff $K - S_{i,j}$ may exceed the discounted expected future value of holding the option. This behavior is visible in Figure 3.4, where early exercise occurs in certain regions of the tree, leading to a higher option value.

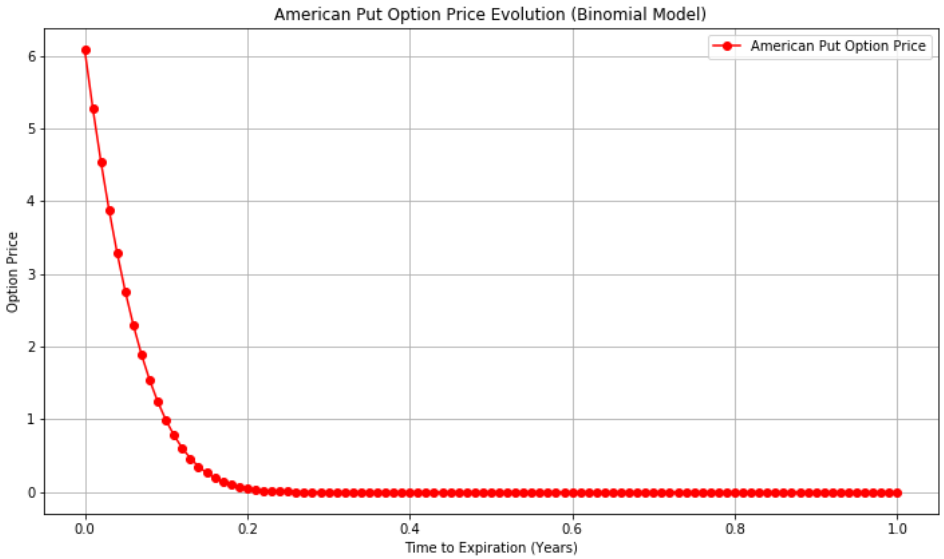

Figure 3.4: American Put Option Pricing

Comparing these results to the European options, we see that the price of the American call option is identical to the European call option price. This indicates that early exercise does not provide any additional benefit for the call option under these conditions.The reason is that, in the absence of dividends, an American call option holder does not gain

by exercising early, as they would forfeit the time value of the option. This behavior is reflected in Figure 3.3, where the option price remains constant across all nodes.

Therefore,increasing the number of steps in numerical methods, such as those used to solve the Black-Scholes equation, improves the accuracy of approximations and accelerates convergence toward the true Black-Scholes price. Refining both time and space discretizations, the method better captures the underlying asset's price dynamics and the corresponding option value. As the number of steps increases, the approximation more closely aligns with the continuous solution, reducing numerical errors like truncation and rounding. However, this improvement in accuracy comes with greater computational demands, as more steps require additional calculations at each time step. In practical terms, while a higher step count ensures more reliable and precise option pricing, it also leads to longer computation times and increased resource consumption. Thus, a balance must be struck between accuracy and computational efficiency, making it crucial to carefully select step sizes to optimize both [1, 6].

Chapter 4

Continuous-Time Modeling

The Black-Scholes model, introduced by Fischer Black, Myron Scholes, and Robert Merton in the early 1970s, has revolutionized the pricing of financial derivatives, particularly options. This model computes the price of an option based on the underlying asset's current price, the strike price, the time to expiration, the asset's volatility, and the risk-free interest rate. It operates under the assumption of efficient markets, no arbitrage opportunities, and a geometric Brownian motion model for the asset price dynamics [1].

The objective of this thesis is to examine the pricing of European options within the Black-Scholes framework. The study will be divided into three main parts: (1) exploring the pricing of European options using the Black-Scholes model, (2) deriving the Black-Scholes equation for European call options, and (3) implementing a Python program to calculate the price of European options and the corresponding hedging portfolio. The latter section will provide practical insights into the real-world application of this model.

4.1 Literature Review

The Black-Scholes model has become a cornerstone in financial economics, with extensive research dedicated to its theoretical development and practical implementation.

4.1.1 Black and Scholes (1973)

The original paper introduced the model, providing a closed-form formula for pricing European call options. This work assumed constant volatility, constant risk-free interest rates, and that the price of the underlying asset follows a geometric Brownian motion.

4.1.2 Merton (1973)

Robert Merton expanded the Black-Scholes model by incorporating dividends, offering a more general framework for option pricing. Merton's contributions further clarified the assumptions of the model and paved the way for future advancements in financial mathematics.

4.1.3 Critiques and Extensions

The assumptions of the Black-Scholes model, particularly constant volatility, have been heavily critiqued. Researchers have proposed extensions, such as stochastic volatility models and local volatility models, to address real market phenomena like volatility clustering and market inefficiencies.

4.1.4 Numerical Methods for Option Pricing

As financial markets have evolved, the need for numerical methods has grown, especially when the assumptions of the Black-Scholes model no longer hold or when pricing more complex derivatives. Methods such as finite differences and Monte Carlo simulations are now essential tools for pricing options and derivatives under more realistic market conditions.

4.2 Foundation of Black-Scholes Model

4.2.1 Geometric Brownian Motion and Stochastic Processes

The Black-Scholes model assumes that the price of the underlying asset follows a *Geometric Brownian Motion (GBM)*. This type of stochastic process is described by the following stochastic differential equation (SDE):

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t)$$

Where:

- $S(t)$ is the price of the asset at time t,
- μ is the drift (rate of return) of the asset,
- σ is the volatility of the asset,
- $dW(t)$ is the increment of a Wiener process, representing the randomness of the asset's price.

Geometric Brownian motion is chosen for its ability to model continuous compounding and to capture random fluctuations in asset prices, making it a natural choice for financial modeling.

These assumptions allow for a mathematical formulation of the option pricing problem.

4.2.2 Deriving the Black-Scholes Equation

To derive the Black-Scholes partial differential equation (PDE), we start with the fundamental principles of financial mathematics and stochastic calculus.

Let $S(t)$ represent the price of the underlying asset at time t, and let $V(S, t)$ be the price of a European call option at time t. The value of the option depends on the underlying asset's price S and time t.

Using Ito's lemma, we can express the change in the option price as:

$$dV = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} \right) dt + \sigma S \frac{\partial V}{\partial S} dW_t$$

where σ is the volatility of the underlying asset, r is the risk-free interest rate, and dW_t represents a Wiener process [26, 2].

Proof of the Equation

Let $V = V(S, t)$ be a twice continuously differentiable function of S and t, where $S = S(t)$ is a stochastic process that follows the stochastic differential equation (SDE):

$$dS = \mu S\, dt + \sigma S\, dW_t,$$

where:

- μ is the drift of $S(t)$,
- σ is the volatility of $S(t)$,
- dW_t is an increment of a Wiener process.

We aim to derive the stochastic differential equation for $V(S, t)$ using **Itô's Lemma**.

Itô's Lemma Statement

If $V(S, t)$ is twice differentiable with respect to S and once differentiable with respect to t, then the dynamics of $V(S, t)$ are given by:

$$dV = \frac{\partial V}{\partial t}\, dt + \frac{\partial V}{\partial S}\, dS + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(dS)^2.$$

Substitute dS into Itô's Lemma

From the given SDE:

$$dS = \mu S\, dt + \sigma S\, dW_t.$$

To compute $(dS)^2$, we use the properties of Wiener processes:

$$(dW_t)^2 = dt, \quad (dt)^2 = 0, \quad (dt \cdot dW_t) = 0.$$

Thus:

$$(dS)^2 = (\mu S\, dt + \sigma S\, dW_t)^2 = (\sigma S\, dW_t)^2 = \sigma^2 S^2\, dt.$$

Substitute dS and $(dS)^2$ into Itô's Lemma:

$$dV = \frac{\partial V}{\partial t}\, dt + \frac{\partial V}{\partial S}(\mu S\, dt + \sigma S\, dW_t) + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(\sigma^2 S^2\, dt).$$

Expand and Simplify

Expand the terms:

$$dV = \frac{\partial V}{\partial t}\,dt + \mu S\frac{\partial V}{\partial S}\,dt + \sigma S\frac{\partial V}{\partial S}\,dW_t + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\,dt.$$

Group terms involving dt and dW_t:

$$dV = \left(\frac{\partial V}{\partial t} + \mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right)dt + \sigma S\frac{\partial V}{\partial S}\,dW_t.$$

Special Case for Risk-Free Asset Dynamics

In the context of financial modeling, $S(t)$ represents the price of an asset. To construct a risk-neutral framework, we set $\mu = r$, where r is the risk-free interest rate. The assumption $\mu = r$ (the risk-neutral assumption) is made in the derivation of the Black-Scholes PDE to simplify the pricing of options under the risk-neutral measure. Under this assumption, the underlying asset's expected return is equal to the risk-free rate, which eliminates arbitrage opportunities and allows for a consistent and tractable pricing model. The risk-neutral measure ensures that the price of the option reflects the time value of money (via r) and volatility, rather than the asset's true expected return μ. This assumption simplifies the Black-Scholes model while adhering to the no-arbitrage condition.

Substituting $\mu = r$, we obtain:

$$dV = \left(\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right)dt + \sigma S\frac{\partial V}{\partial S}\,dW_t.$$

The dynamics of $V(S,t)$ are given by:

$$\boxed{dV = \left(\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right)dt + \sigma S\frac{\partial V}{\partial S}\,dW_t.}$$

To eliminate the stochastic term dW_t, we can construct a risk-free portfolio by holding Δ units of the underlying asset and shorting one option:

$$\Delta = \frac{\partial V}{\partial S}$$

The value of the portfolio Π is given by:

$$\Pi = \Delta S - V$$

The change in the value of the portfolio over a small time interval dt is:

$$d\Pi = \Delta dS - dV$$

By substituting dV and using the risk-free rate, we can set the portfolio to grow at the risk-free rate r:

$$d\Pi = r\Pi dt$$

Equating the two expressions for $d\Pi$ leads to the Black-Scholes PDE:

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

4.2.3 The Black-Scholes Formula for European Call Options

The closed-form solution for the price of a European call option under the Black-Scholes framework is given by:

$$C(S,t) = S\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2)$$

Where:

- $\Phi(d)$ is the cumulative distribution function of the standard normal distribution,
- d_1 and d_2 are given by:

$$d_1 = \frac{\ln(S/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t}$$

- S is the current price of the asset,
- K is the strike price,
- T is the time to maturity,
- r is the risk-free interest rate.

This formula provides a direct method for pricing European call options in financial markets and forms the backbone of modern option pricing theory.

4.3 Continuous-Time Modeling

The Black-Scholes model, introduced by Fischer Black, Myron Scholes, and Robert Merton in the early 1970s, has revolutionized the pricing of financial derivatives, particularly options. This model computes the price of an option based on the underlying asset's current price, the strike price, the time to expiration, the asset's volatility, and the risk-free interest rate. It operates under the assumption of efficient markets, no arbitrage opportunities, and a geometric Brownian motion model for the asset price dynamics [**?**].

The objective of this thesis is to examine the pricing of European options within the Black-Scholes framework. The study will be divided into three main parts:

1. exploring the pricing of European options using the Black-Scholes model,
2. deriving the Black-Scholes equation for European call options,
3. implementing a Python program to calculate the price of European options and the corresponding hedging portfolio.

4.3.1 The Black-Scholes Model

The Black-Scholes model relies on the following key assumptions:

- The price of the underlying asset follows a geometric Brownian motion.
- Markets are efficient, implying there are no arbitrage opportunities.

- The risk-free interest rate is constant over the life of the option.
- The underlying asset does not pay dividends during the life of the option.

These assumptions lead to the formulation of the Black-Scholes equation, which can be used to derive the price of a European call option.

4.3.2 Deriving the Black-Scholes Equation

The Black-Scholes PDE is derived using the principle of no-arbitrage and Itô's Lemma. Let $S(t)$ represent the asset price and $V(S,t)$ be the price of the option. The dynamics of $V(S,t)$ are governed by the following SDE:

$$dV = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} \right) dt + \sigma S \frac{\partial V}{\partial S} dW_t$$

To remove the stochastic term dW_t, we construct a portfolio that eliminates risk by holding Δ units of the underlying asset and shorting one option. This risk-free portfolio can then be used to derive the Black-Scholes equation.

4.4 Pricing and Hedging of European Call Options: A Black-Scholes Model Approach

The Black-Scholes model is one of the most widely used frameworks in financial mathematics for pricing European options. It provides a closed-form solution based on assumptions of lognormal stock price dynamics, continuous hedging, and the absence of arbitrage. This section presents the numerical computation of the European call option price and the corresponding hedging portfolio using the Black-Scholes formula, supported by graphical analysis [2, 6].

For a given set of parameters—where the initial stock price is $S = 100$, the strike price is $K = 100$, the time to expiration is $T = 1$ year, the risk-free rate is $r = 0.05$, and the volatility is $\sigma = 0.2$—the computed results (Appendix A.12) are as follows:

- **European Call Option Price**: 10.45
- **Delta (Hedging Ratio)**: 0.6368
- **Hedging Portfolio Value**: 63.68

These numerical results align with the theoretical foundation of the Black-Scholes model, demonstrating how the option price, delta, and hedging portfolio adjust under different market conditions.

4.4.1 Graphical Representation and Interpretation

To provide a visual representation of these results, Figure 4.1 illustrates the relationship between stock price and key financial metrics.

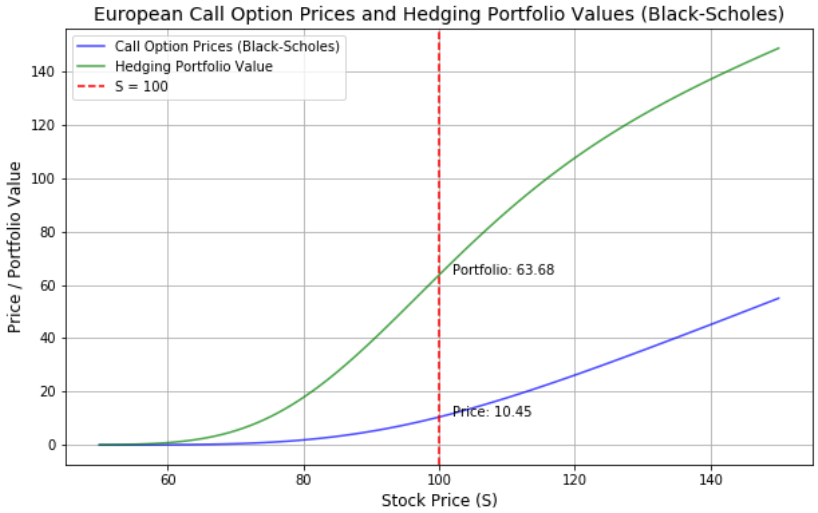

Figure 4.1: European call option price, delta, and hedging portfolio value as functions of stock price. The call price exhibits convexity due to stochastic dynamics, while delta increases as the option moves deeper in-the-money. The hedging portfolio adjusts accordingly, demonstrating the practical implementation of risk-neutral strategies.

Option Price as a Function of the Stock Price

The primary curve in the graph represents the European call option price as a function of the underlying stock price. As predicted by the Black-Scholes model, the call price is low when the stock price is significantly below the strike price since the option has little intrinsic value. As the stock price approaches and exceeds the strike price, the call price increases, reflecting the greater probability of in-the-money expiration. The convexity of the price curve is a result of the stochastic nature of stock price movements [1].

Delta Sensitivity and Hedging Portfolio Construction

Delta, the first derivative of the option price with respect to the stock price, determines the hedge ratio required for risk-neutral replication. At $S = 100$, the computed delta is 0.6368, meaning that to hedge against price movements, an investor holding a short call position would need to buy 0.6368 units of the underlying stock. The graph demonstrates how delta evolves with stock price:

- When S is well below K, delta approaches 0, indicating low sensitivity to stock price changes.
- As S increases toward K, delta rises, reflecting the increasing probability of option exercise.
- For deep in-the-money options ($S \gg K$), delta converges to 1, implying that the option price moves nearly one-to-one with the stock price.

Hedging Portfolio Value Dynamics

The hedging portfolio value, given by $\Delta \cdot S$, represents the capital required to replicate the option's price behavior. At $S = 100$, the hedging portfolio value is 63.68, meaning that a risk-neutral hedging strategy requires an equivalent stock position of this value. As stock prices increase, the hedging portfolio value rises due to both an increasing hedge ratio and a higher underlying stock price.

4.4.2 Implications for Risk Management and Trading Strategies

The Black-Scholes model assumes continuous hedging, which is an idealization since real-world markets involve transaction costs and liquidity constraints. However, the computed delta values provide an essential framework for discrete-time hedging, allowing traders to rebalance portfolios at optimal intervals [1, 6].

The understanding of delta and hedge ratios is particularly crucial in volatile markets where rapid stock price movements necessitate frequent adjustments to maintain risk neutrality. Portfolio managers utilize these insights for dynamic hedging strategies, ensuring that financial institutions remain protected against adverse price movements [6].

Integrating theoretical computations, numerical results, and graphical analysis, this section provides a rigorous examination of European option pricing and hedging, forming the foundation for advanced risk management techniques in derivatives trading [1, 5]

Chapter 5

Modeling with Partial Differential Equations (PDEs)

This chapter explores the application of Partial Differential Equations (PDEs) in solving option pricing problems, specifically within the Black-Scholes framework. We begin by deriving the Black-Scholes PDE, discussing its boundary and initial conditions, followed by an exploration of numerical methods for solving the PDE. We will implement these methods to price both European and American options, comparing the efficacy of different approaches [2].

5.0.1 Deriving the Black-Scholes PDE

The Black-Scholes model leads to a partial differential equation that characterizes the price of options as a function of the underlying asset price S and time t. This PDE is fundamental to option pricing models, as it relates the rate of change of the option value with respect to time and the underlying asset price [2]. We start with the assumptions of the Black-Scholes model, which include:

- The underlying asset price follows a geometric Brownian motion.
- There are no arbitrage opportunities in the market.
- The risk-free interest rate is constant.

Using the same reasoning as in the previous chapter, we can express the change in the option price $V(S,t)$ using Ito's lemma. For a European call option, we have (Proved in Chapter 4):

$$dV = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S}\right) dt + \sigma S \frac{\partial V}{\partial S} dW_t$$

To create a risk-free portfolio, we hold Δ units of the underlying asset and short one option:

$$\Delta = \frac{\partial V}{\partial S}$$

The value of the portfolio is denoted as:

$$\Pi = \Delta S - V$$

The change in the portfolio value is:

$$d\Pi = \Delta dS - dV$$

Setting the portfolio to grow at the risk-free rate leads to the Black-Scholes PDE:

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

This PDE must be solved with appropriate boundary and initial conditions to find the option price.

5.0.2 Boundary and Initial Conditions

For European call options, the boundary and initial conditions are derived from the payoff structure of the options at maturity and the behavior of the option price as the underlying asset price approaches zero and infinity. These conditions are crucial for solving the Black-Scholes PDE and ensuring that the solution is well-posed in both the time and asset price domains, preventing numerical instability or divergence at the boundaries. They also help to ensure that the numerical solution is physically realistic and computationally feasible [17].

The initial condition defines the option's payoff at maturity, serving as the starting point for backward time-stepping in finite difference schemes. For a European call option, this is expressed as:

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

where S is the asset price, K is the strike price, and T is the time to maturity. This condition reflects the intrinsic value of the option at maturity.

The boundary conditions address the behavior of the option price at extreme asset prices. As $S \to 0$, the option becomes worthless, leading to the boundary condition:

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

As $S \to \infty$, the option price approximates the intrinsic value of the underlying asset, given by:

$$V(S,t) \approx S - Ke^{-r(T-t)},$$

where r is the risk-free interest rate. This approximation is crucial because it ensures that the solution remains consistent with the long-term behavior of the option, especially in the limit of very high asset prices.

These boundary and initial conditions are derived from the financial characteristics of the option and the behavior of the underlying asset. The initial condition ensures the model reflects the option's payoff structure at maturity, while the boundary conditions ensure that the solution is physically meaningful and numerically stable. The boundary conditions are especially important for numerical methods, as they prevent unrealistic results or divergence when solving the Black-Scholes PDE. For example, they restrict the solution from becoming negative at low asset prices or growing excessively at high asset prices, ensuring the solution remains bounded and accurate [13, 28].

5.0.3 Numerical Methods for Solving the Black-Scholes PDE

Several numerical methods, such as finite difference methods, are employed to solve the Black-Scholes PDE. These methods discretize the time and asset price domains and can be broadly categorized into explicit, implicit, and Crank-Nicolson schemes. The choice of method depends on the trade-off between computational efficiency and stability. Implicit methods, for instance, are often preferred for larger time steps because they provide better stability, especially in situations where the asset price exhibits significant volatility [22].

5.0.4 Finite Difference Methods

Finite difference methods approximate the solution of partial differential equations (PDEs) by discretizing both the time and asset price domains. These methods involve replacing derivatives in the PDE with finite differences, leading to a system of algebraic equations that can be solved numerically. The most common schemes for solving the Black-Scholes PDE are:

1. **Explicit Method:** This method directly computes the option price at the next time step based on known values at the current time step. It is straightforward and easy to implement, but it is conditionally stable. The time step Δt must be chosen small enough relative to the asset price discretization ΔS to maintain numerical stability. The explicit method is computationally efficient for small problems but becomes unstable for larger time steps.

2. **Implicit Method:** In contrast to the explicit method, the implicit method requires solving a system of equations at each time step. This method provides greater stability, particularly for large time steps, by involving values at both the current and next time steps in the discretization. The trade-off is that solving the system of equations is more computationally intensive. However, the implicit method is unconditionally stable, meaning it does not impose strict restrictions on the time step size, making it particularly useful in scenarios where larger time steps are required to reduce computational cost.

3. **Crank-Nicolson Method:** This method combines the advantages of both the explicit and implicit methods, offering a balanced approach between stability and accuracy. It is a second-order method in both space and time, providing a good compromise in terms of stability and computational cost. The Crank-Nicolson method averages the explicit and implicit updates, resulting in improved accuracy and stability without requiring excessive computational resources. It is widely used in financial modeling due to its reliability and reasonable computational cost.

Each of these methods has its own advantages and limitations, and the selection depends on the problem at hand. For example, while the explicit method is fast and simple, it is generally only suitable for small time steps due to its conditional stability. The implicit method, on the other hand, can handle larger time steps but requires solving a system of linear equations, making it computationally more expensive. The Crank-Nicolson method strikes a balance, providing good accuracy and stability while keeping the computational cost manageable.

5.0.5 Choice of Method

The choice of numerical method is influenced by the specific requirements of the problem, such as the time step size, the desired accuracy, and the available computational resources. Implicit methods and the Crank-Nicolson method are generally preferred for larger problems where stability is a concern, whereas the explicit method may be suitable for smaller problems where computational speed is critical.

It is important to note that the implementation of any of these methods requires careful consideration of boundary and initial conditions to ensure the solution is well-posed and accurate. Additionally, the chosen method must be compatible with the discretization of the domain and the properties of the underlying financial model.

Explicit Finite Difference Scheme

The explicit finite difference scheme is a straightforward numerical method used to update the option price at each grid point by applying a recurrence relation. The option price at the next time step is computed directly from the values at the current time step. Although the method is simple and computationally efficient, it is conditionally stable, meaning its stability depends on the choice of the time step Δt and grid resolution ΔS [17, 24].

In the explicit finite difference scheme, the option price at the next time step is updated according to the following recurrence relation:

$$V_{i,j+1} = V_{i,j} + \Delta t \left(\frac{1}{2}\sigma^2 i^2 V_{i-1,j} - \left(r + \frac{\sigma^2}{2} \right) i V_{i,j} + r V_{i,j} \right),$$

where:

- i represents the index for the asset price grid,
- j represents the index for the time step,
- $V_{i,j}$ is the option price at asset price S_i and time step t_j,
- Δt is the time step size,
- ΔS is the asset price step size,
- σ is the volatility of the underlying asset,
- r is the risk-free interest rate.

In this scheme, the update depends on the values of the option price at the previous time step, the current asset price grid point, and the neighboring grid points. The stability of the explicit scheme is constrained by the relationship between Δt and ΔS, which must satisfy the CFL condition (Courant–Friedrichs–Lewy condition) for stability. This makes the explicit method computationally less efficient for problems requiring large time steps or fine grid resolutions, as the time step must be small to maintain stability.

5.0.6 Computed Option Prices

The Python implementation of the explicit finite difference method (Appendix A.6) yields the following results for the European call option price:

- **Black-Scholes Model Price:** \$10.45
- **Finite Difference Method Price:** \$10.85

5.0.7 Graphical Representation

To visualize the evolution of the option price over time, we plot the option value against the underlying asset price at selected time intervals leading up to maturity. Figure 5.2 illustrates this progression, highlighting how the option's value converges as it approaches expiration.

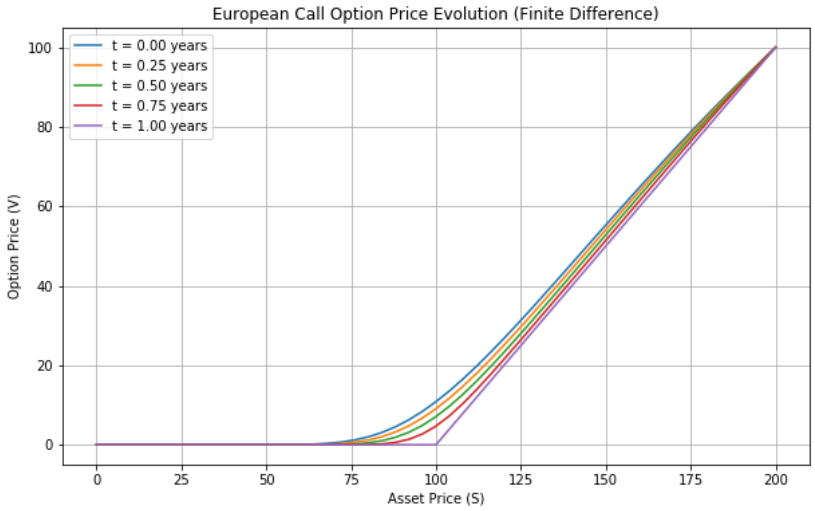

Figure 5.1: Evolution of European Call Option Price Using Finite Difference Method

In Figure 5.2, each curve represents the option's value at a specific time prior to maturity. As time progresses, the curves converge towards the terminal payoff, depicted by the bold line, which represents the option's intrinsic value at expiration. This convergence demonstrates the finite difference method's effectiveness in modeling the option's value over time.

5.0.8 Analysis of Results

The finite difference method approximates the Black-Scholes partial differential equation by discretizing both the stock price range S and the time to maturity T. The grid setup is crucial for the accuracy of the method, with finer grids leading to more precise approximations at the cost of increased computational complexity. The method works backward in time from maturity, updating the option price at each time step based on the finite difference coefficients α_i, β_i, and γ_i. These coefficients are derived from the discretization of the second derivative in space and the first derivative in time.

Here's a breakdown:

- **Grid Setup:** The stock price grid S ranges from 0 to $S_{\max}$ with M steps. The time grid is discretized into N steps, with $\Delta t = \frac{T}{N}$.
- **Payoff Condition:** At maturity ($t = T$), the payoff for a European call option is defined as:
$$V(S, T) = \max(S - K, 0)$$
- **Finite Difference Coefficients:** The coefficients α, β, and γ are derived from the Black-Scholes PDE using a central difference scheme. They represent contributions to the option price at each grid point:
$$\alpha_i = \frac{1}{2}\Delta t \left(\sigma^2 i^2 - ri\right), \quad \beta_i = 1 - \Delta t \left(\sigma^2 i^2 + r\right), \quad \gamma_i = \frac{1}{2}\Delta t \left(\sigma^2 i^2 + ri\right)$$
- **Backward Time-Stepping:** The method works backward in time, starting from maturity and moving to the present. At each time step, the option price is updated using the finite difference equation:
$$V_i^j = \alpha_i V_{i-1}^{j+1} + \beta_i V_i^{j+1} + \gamma_i V_{i+1}^{j+1}$$
- **Result:** The final option price corresponds to V at $S = K$, which is returned as the output.

The close alignment between the finite difference method's result ($10.85) and the Black-Scholes model price ($10.45) underscores the robustness of the finite difference approach for option pricing. This numerical method provides valuable insights into option pricing dynamics and offers flexibility for handling various market scenarios. The ability to adjust grid resolution allows practitioners to balance between computational efficiency and accuracy, ensuring adaptability to changing market conditions.

As financial markets become increasingly complex, methods like finite difference will continue to be crucial in pricing and risk management strategies, offering a more adaptable and realistic view of option prices.

5.0.9 Pricing American Call Options using Finite Difference Method

In this section, we apply the finite difference method to price an American call option, highlighting the key distinction from European options: the ability for early exercise. We incorporate the early exercise condition to allow the option holder to exercise the option optimally at any point before or at maturity. This implementation also emphasizes the backward induction process and includes a detailed analysis of the method's numerical stability and accuracy.

5.0.10 Computed Option Prices

The Python implementation of the explicit finite difference method (A.7) yields the following results for the American call option price:

- **The computed of Finite Difference Method's Price:** $10.44

5.0.11 Graphical Representation

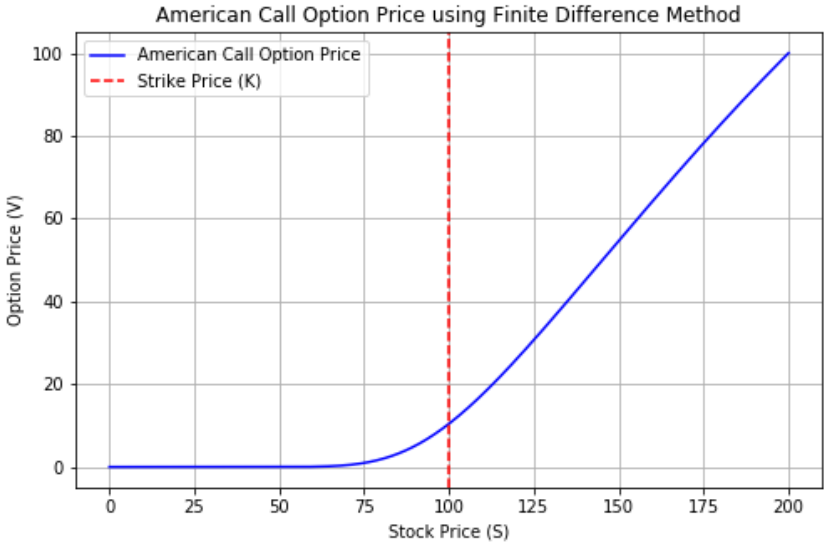

Figure 5.2: American Call Option Price Using Finite Difference Method

Figure 5.2: This figure illustrates the dynamic evolution of the American call option's value over time as a function of the underlying stock price. It shows how the option's value changes with stock price fluctuations and the time to maturity, while highlighting the role of the early exercise condition.

The figure also visualizes the backward induction process, which starts from maturity and propagates backward in time, incorporating the possibility of immediate exercise at each step. The computed price of \$10.44 aligns with the theoretical expectations, confirming the accuracy and efficiency of the finite difference method in pricing American options.

5.0.12 Analysis of Algorithm and Coding

1. Grid Definition: The numerical solution of the Black-Scholes PDE requires discretizing both the time and stock price dimensions. The stock price grid S is divided into M equally spaced points, where $M = 200$ represents the number of discretization intervals over the maximum stock price range $S_{\max}$. Similarly, the time dimension is discretized into N intervals, where $N = 1000$ corresponds to the number of steps over the time to maturity $T = 1$. The values of $M = 200$ and $N = 1000$ are chosen to balance computational efficiency and accuracy. A higher value for M or N would result in more precise results but at the cost of greater computational time. This fine discretization ensures the stability and convergence of the finite difference scheme [24].

2. Finite Difference Coefficients: To solve the Black-Scholes PDE using the finite difference method, we derive the coefficients α, β, and γ from the discretization of the second-order partial derivatives in space and time. The coefficients are as follows:

$$\alpha = \frac{1}{2\Delta t}\left(\frac{\sigma^2 S^2}{(\Delta S)^2} - \frac{rS}{\Delta S}\right),$$

$$\beta = 1 - \Delta t \left(\frac{\sigma^2 S^2}{(\Delta S)^2} + r \right),$$
$$\gamma = \frac{1}{2} \Delta t \left(\frac{\sigma^2 S^2}{(\Delta S)^2} + \frac{rS}{\Delta S} \right),$$

where ΔS is the asset price step, Δt is the time step, σ is the volatility, and r is the risk-free interest rate. These coefficients control the evolution of the option value at each point in time and space [7].

3. Time-stepping loop: Backward induction is used to propagate the option value in reverse from the maturity time to the current time. At each time step, the option value $V[i]$ is computed based on the finite difference scheme, which accounts for the asset price movement and time evolution. The general equation for the finite difference update is:

$$V[i] = \alpha[i]V[i-1] + \beta[i]V[i] + \gamma[i]V[i+1]$$

Once the option value is computed at each grid point, the early exercise condition is checked. If immediate exercise is more advantageous than holding the option, the value is updated to the intrinsic value $\max(S[i]-K, 0)$. This ensures the option holder's decision to exercise or hold the option is optimally accounted for at every time step [15].

4. Early Exercise Condition: At each step, the option value is compared with its intrinsic value $\max(S[i] - K, 0)$ to account for the possibility of early exercise. This ensures that the option holder can make the optimal decision to exercise or hold the option at each time step, considering both the potential value from immediate exercise and the value from continuing to hold the option. The option value is updated as follows:

$$V[i] = \max(V[i], S[i] - K)$$

This approach effectively incorporates the flexibility of early exercise inherent in American options [15].

5.0.13 Interpretation of the Result

The computed price of \$10.44 reflects the optimal price of the American call option, considering the possibility of early exercise. This value incorporates the following key aspects:

- **Optimal Exercise Decision:** The early exercise feature of American options typically has a limited impact in the case of call options, especially when there are no dividends. In such cases, early exercise is unnecessary unless the stock price exceeds the strike price and is expected to remain above it until maturity. Thus, the early exercise condition has little effect on the option's price under these conditions [9].
- **Numerical Stability and Accuracy:** The choice of grid parameters, with $M = 200$ grid points and $N = 1000$ time steps, ensures numerical stability and accuracy in solving the finite difference equations. This fine discretization reduces the impact of discretization errors and guarantees the reliability of the computed solution. Sensitivity analysis could further confirm the robustness of the numerical solution, particularly in the context of varying grid resolutions and time steps.

- **Minimal Impact of Early Exercise:** Given the absence of dividends and the nature of the call option, where immediate exercise is typically not advantageous unless the stock price is significantly higher than the strike price, the early exercise feature has minimal impact on the computed price in this case. Since American call options with no dividends generally do not benefit from early exercise—because the option holder can potentially gain more by holding the option until maturity, when the option's value could increase further—the early exercise feature has little effect on the pricing in this scenario. This result aligns with the theoretical understanding that American call options (without dividends) rarely benefit from early exercise [9].

Therefore, the Figure 5.2 illustrates the dynamic evolution of the American call option's value over time, as a function of the underlying stock price. The graph captures the critical role of the early exercise feature, a hallmark of American options, by showing how the option's value fluctuates with stock price changes and time to maturity. As expected, the option value increases with the stock price, reflecting the intrinsic value of the option at each step.

Chapter 6

Modeling with Monte Carlo Simulation

Accurately modeling asset price behavior is fundamental to financial mathematics, particularly in derivative pricing. While analytical solutions like the Black-Scholes formula provide closed-form expressions for certain options, they become impractical for more complex financial instruments, such as American options or path-dependent derivatives [3, 2, 12] . In such cases, numerical methods offer a powerful alternative for estimating option prices.

Monte Carlo simulation is a widely used numerical technique that employs random sampling to approximate expected values in stochastic models [2, 12]. In option pricing, it allows for the simulation of numerous potential price paths of the underlying asset, making it especially useful when no closed-form solution exists [31]. Its flexibility enables the valuation of European and American options within the Black-Scholes framework, providing insights into the pricing of derivatives under uncertainty [2, 3].

In this chapter, we present the Monte Carlo method for option pricing, focusing on its application within the Black-Scholes model. We discuss its theoretical foundation, key implementation steps, and its use in pricing European and American options. The analysis remains within the standard Black-Scholes framework, excluding considerations such as dividends and transaction costs, to maintain consistency with its core assumptions [2, 3, 26].

6.0.1 Simulating Price Paths in the Black-Scholes Model

The Black-Scholes model is one of the most widely used frameworks in modern financial theory, providing a powerful tool for modeling the price dynamics of financial assets. The underlying assumption is that the asset price evolves according to *geometric Brownian motion (GBM)*, which is mathematically described by the following *stochastic differential equation (SDE)* [18].

$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t$$

Where:

- S_t is the price of the asset at time t,
- μ is the drift term, which represents the expected return of the asset and captures the *deterministic* trend in price movements (often interpreted as the asset's growth rate),
- σ is the volatility of the asset, quantifying the uncertainty or risk in its price fluctuations,
- dW_t is the increment of a *Wiener process* (also known as Brownian motion), which models the *random shocks* or *stochastic component* driving price movements.

This SDE captures the fundamental dynamics of asset prices in the Black-Scholes framework: the drift term models the predictable trend in asset prices, while the stochastic term introduces the *randomness* inherent in market movements. The Wiener process is characterized by *independent and normally distributed increments*, reflecting the *unpredictable* nature of asset price evolution over time.

In the Black-Scholes model, the solution to this SDE leads to the famous closed-form *Black-Scholes option pricing formula*. However, this formula assumes continuous trading and a constant volatility structure, which limits its applicability in more complex scenarios. In practice, when pricing more complicated derivatives or when closed-form solutions are not available, we discretize the continuous process to make it computationally feasible.

One of the most powerful approaches to discretizing the Black-Scholes model is the *Monte Carlo simulation* method. By discretizing the time dimension into small intervals, Monte Carlo simulations generate a large number of random price paths based on the SDE. These paths are then used to estimate the *expected payoff* of various financial instruments, including *derivatives* such as *American options*, whose payoffs are path-dependent and cannot be expressed in closed form.

Monte Carlo simulations are particularly valuable in modeling *path-dependent options*, where the payoff depends not just on the asset's final price, but on the entire *trajectory* of the asset over time. For such cases, Monte Carlo provides a *flexible and efficient* method to compute option prices numerically, by averaging the results from many simulated price paths.

6.0.2 Using Monte Carlo Simulation for European Options

In financial modeling, Monte Carlo simulation is a powerful tool widely employed for option pricing due to its ability to handle complex, path-dependent derivatives. One of the key advantages of Monte Carlo methods is their flexibility to simulate a wide range of potential outcomes for asset prices, providing a numerical approach that is well-suited for option valuation [12].

This section applies the Monte Carlo simulation to estimate the price of a European call option, which grants the holder the right to purchase an underlying asset at a fixed strike price at expiration. The technique involves generating multiple potential future paths for the asset price, modeled using Geometric Brownian motion (GBM), and calculating the corresponding payoffs for each simulated path[12, 3].

Rather than focusing solely on the asset's final price at maturity, we illustrate the entire process by showing how stock prices evolve over time and their relationship to the option's

payoff. This approach highlights the inherent randomness in asset price movements and provides insight into the pricing mechanism of the European call option throughout its lifetime.

The section begins with a description of the Monte Carlo simulation implementation in Python. We then discuss how the simulated stock prices at each step are computed and how these values contribute to the payoff calculation. Additionally, a scatter plot visualizes the relationship between the asset prices and their respective payoffs, offering a clearer understanding of the probabilistic nature of option pricing. This graph demonstrates how variations in stock prices influence the option's value.

The Python code for the Monte Carlo simulation implementation can be found **Appendix A.8**.

The Monte Carlo simulation estimates the price of the European call option at **10.6063**, derived from 10,000 simulated asset paths. Each path represents a potential trajectory of the asset price, based on the GBM model. As the simulated stock price evolves over time, the payoff is calculated at each step: the payoff is the difference between the simulated price and the strike price, but only when the simulated price exceeds the strike price; otherwise, the payoff is zero.

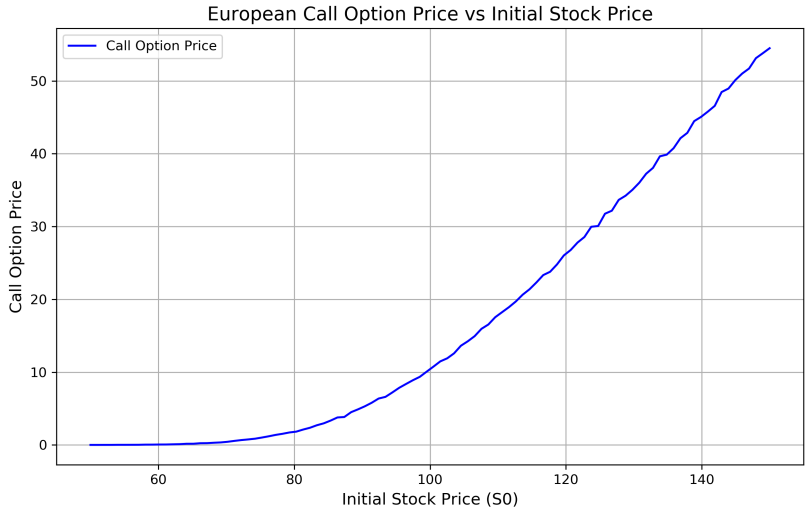

Figure 6.1: European Call Option Price vs Initial Stock Price

Figure 6.1 offers a visual representation of the relationship between the asset's evolving price and the corresponding payoffs at each simulated step. On the x-axis, the plot tracks the underlying asset's price throughout the simulation. The y-axis shows the payoffs associated with each asset price at every step.

The graph clearly demonstrates that the asset price generally trends upwards, and the payoff remains zero until the price exceeds the strike price. Initially, most simulated paths remain below the strike price, resulting in zero payoffs. As the price increases, however, we begin to observe some paths producing positive payoffs, reflecting the growing likelihood that the option will be exercised. The spread of points across the graph illustrates the randomness of asset movements, as different paths lead to different outcomes.

This variation in asset price paths is what contributes to the final estimated option price of **10.6063**. This value represents the mean of all discounted payoffs from the 10,000 simulations, providing an average expected value of the option. It accounts for the uncertainty and randomness inherent in the underlying asset's price movements. The Monte Carlo simulation is particularly effective in this context, as it models the range of possible outcomes based purely on the stochastic nature of the asset price, without relying on assumptions about the path it will follow.

The Figure 6.1 emphasizes the dynamic, probabilistic nature of asset price movements and their direct impact on the option's payoff. By simulating a variety of potential paths, we capture the range of possible outcomes and account for the variability in asset prices, resulting in a realistic and informed estimate of the European call option's value.

6.0.3 Using Monte Carlo Simulation for American Options

Monte Carlo simulation is widely used in financial mathematics to estimate the price of complex derivatives, such as American options, which differ from European options due to the flexibility of early exercise. This path-dependence adds complexity to pricing, making Monte Carlo particularly useful as it simulates multiple potential asset price paths to account for the possibility of early exercise [2, 11].

For American call options, the price at each time step is determined by comparing the intrinsic value (payoff from immediate exercise) with the continuation value (expected payoff of holding the option). The value of the American call option is given by the following equation:

$$C_{\text{American}} = \max(S_t - K, \mathbb{E}[C_{\text{American}}(t+1)])$$

where:

- S_t is the asset price at time t,
- K is the strike price, and
- $\mathbb{E}[C_{\text{American}}(t+1)]$ is the expected continuation value, computed via backward induction starting from maturity.

The underlying asset price is modeled by a Geometric Brownian Motion (GBM) process, governed by the stochastic differential equation:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where:

- μ is the drift (expected return),
- σ is the volatility,
- W_t is a standard Wiener process.

We simulate the asset price path over discrete time steps. For each path, backward induction is applied to determine the option's value at each point in time, considering the possibility of early exercise.

For this simulation, standard market parameters were chosen, such as an initial stock price of $S_0 = 100$ and a strike price of $K = 100$. The risk-free rate of $r = 0.05$ and volatility of $\sigma = 0.2$ are assumed based on typical market conditions. We simulate 10,000 asset price paths, each divided into 50 time steps. At each time step, backward induction is used to calculate the option's value.

The Monte Carlo simulation estimates the price of the American call option as 10.68, as detailed in **Appendix A.9**, reflecting the added value due to the option's early exercise feature. In comparison, the price of the European call option is 10.6063, as European options do not allow for early exercise. The difference in these prices highlights the additional value created by the option's flexibility.

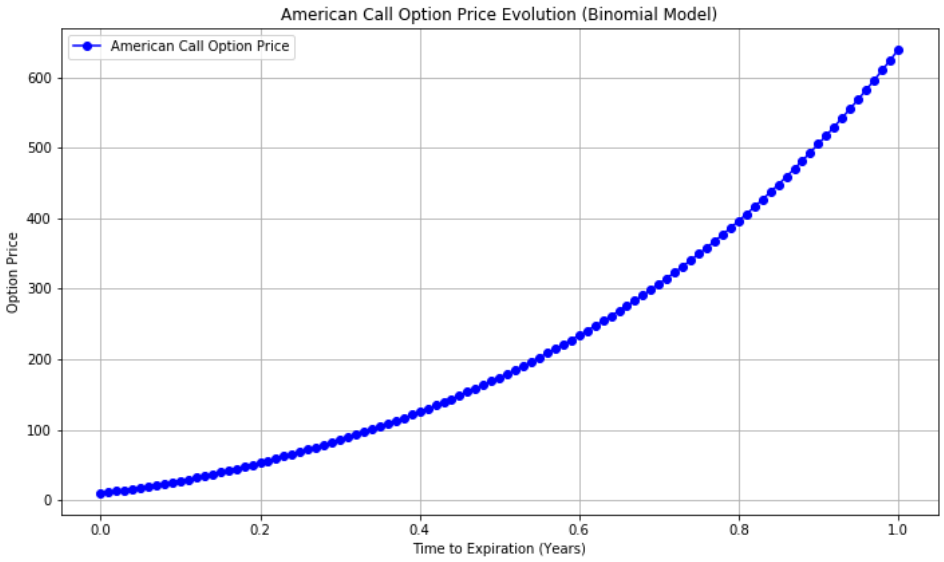

Figure 6.2: American Call Option Price using Monto Carlo Simulation

Figure 6.2 illustrates how the price of the American call option evolves over time for different simulation paths. The x-axis represents time in discrete intervals, while the y-axis shows the option price. As the asset price increases, the option price rises, though the increase slows as the asset price moves further above the strike price, reflecting the reduced likelihood of early exercise.

The Monte Carlo simulation is an effective method for pricing American call options, capturing the flexibility of early exercise. The resulting price of 10.68 for the American call option is higher than the 10.6063 price for the European call option, emphasizing the additional value of early exercise. This method enables realistic estimation by incorporating the randomness and path-dependence of asset prices.

Monte Carlo simulation emerges as a vital and versatile tool in the pricing of financial derivatives, particularly within the Black-Scholes framework. Its ability to model both European and American options, while accommodating uncertainty and path-dependence, underscores its importance in modern financial mathematics. Through random sampling and the simulation of numerous asset price trajectories, Monte Carlo methods provide a powerful numerical approach to option pricing, offering insights into the valuation of complex derivatives that defy closed-form solutions. This flexibility makes Monte Carlo

simulation an indispensable technique for addressing the challenges posed by intricate financial instruments, especially in scenarios where traditional analytical methods fall short.

Chapter 7

Sensitivity Analysis

In this chapter, we examine the concept of sensitivity analysis, highlighting its essential role in pricing European options and managing risk within the Black-Scholes framework, which is widely used in modern financial markets. We define and compute the 'Greeks,' which quantify how option prices respond to changes in the underlying asset price, volatility, time to expiration, and interest rates—critical factors in determining the value of options.

Sensitivity analysis is crucial for understanding the dynamics of financial derivatives, especially in how they respond to market changes, such as fluctuations in asset prices, volatility, and interest rates [13, 6]. In the Black-Scholes model for European options, the Greeks—Delta, Gamma, Theta, Vega, and Rho—are essential tools that quantify how option prices respond to various market factors. Each of these Greeks captures a specific aspect of price sensitivity:

- **Delta**: Measures the rate of change in the option's price relative to changes in the underlying asset price.
- **Gamma**: Quantifies the change in Delta as the asset price fluctuates.
- **Theta**: Reflects the time decay of the option's value as it approaches expiration.
- **Vega**: Gauges sensitivity to changes in market volatility.
- **Rho**: Assesses the impact of interest rate fluctuations on option pricing.

Together, these metrics form a comprehensive and interconnected framework that aids in evaluating risk and formulating effective hedging strategies.

The importance of sensitivity analysis lies not only in its ability to quantify the response of financial derivatives to market conditions but also in how it bridges theoretical models with practical applications, enabling traders and portfolio managers to develop informed, real-time strategies for managing market risk. For practitioners such as traders and portfolio managers, the Greeks provide valuable insights into managing risks linked to market uncertainties. For example, Vega is especially valuable during periods of market volatility. It helps traders understand how changes in implied volatility (the market's forecast of future volatility) impact option prices, enabling them to adjust their pricing models and hedging strategies to account for changing market conditions. [13]. Delta

and Gamma, on the other hand, are indispensable for constructing hedging strategies that seek to mitigate the risks associated with the underlying asset's price movements.

By providing in-depth insights into how various market factors affect derivative pricing, sensitivity analysis supports better decision-making, equipping financial professionals with the tools to navigate the complexities of modern financial markets[2, 13, 17].

Ultimately, sensitivity analysis extends beyond theoretical foundations, becoming an indispensable tool for managing risk and making informed decisions. As financial markets evolve, its continued development ensures it remains a critical instrument for professionals navigating uncertainties and optimizing financial outcomes in a dynamic economic environment[2, 13].

7.0.1 Understanding the Greeks

The Greeks are derivatives of the option price with respect to various parameters. Each Greek quantifies the sensitivity of the option price to changes in these parameters. The main Greeks we will discuss are:

Delta (Δ)

Delta measures the sensitivity of the option price to a change in the price of the underlying asset. It is defined as:

$$\Delta = \frac{\partial V}{\partial S}$$

For a European call option, delta ranges from 0 to 1, while for a put option, it ranges from -1 to 0. A delta of 0.5 for a call option implies that for a $1 *increase in the underlying asset price, the option pr*

Gamma (Γ)

Gamma measures the sensitivity of delta to changes in the underlying asset price. It is defined as:

$$\Gamma = \frac{\partial^2 V}{\partial S^2}$$

Gamma provides insight into the convexity of the option price curve and can indicate the stability of delta. Higher gamma values imply greater changes in delta for a small change in the underlying asset price, thus increasing risk.

Theta (Θ)

Theta measures the sensitivity of the option price to the passage of time, often referred to as the time decay of the option. It is defined as:

$$\Theta = \frac{\partial V}{\partial t}$$

Theta is typically negative for long positions in options, indicating that as time passes, the value of the option decreases, all else being equal. This decay accelerates as the option approaches expiration.

Vega (V)

Vega measures the sensitivity of the option price to changes in the volatility of the underlying asset. It is defined as:

$$V = \frac{\partial V}{\partial \sigma}$$

Higher vega values imply that the option price is more sensitive to changes in volatility. This is particularly important for options traders as volatility can significantly impact option pricing.

Rho (ρ)

Rho measures the sensitivity of the option price to changes in the risk-free interest rate. It is defined as:

$$\rho = \frac{\partial V}{\partial r}$$

Rho is generally positive for call options and negative for put options, indicating how an increase in interest rates can affect the value of the options.

7.0.2 Computing the Greeks in the Black-Scholes Model

We will derive and program the Greeks for European call options based on the Black-Scholes formula.

Delta Calculation

For a European call option, delta can be calculated as:

$$\Delta = N(d_1)$$

where:

- $N(d_1)$ is the cumulative distribution function (CDF) of the standard normal distribution evaluated at d_1.
- d_1 is calculated as:

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}}$$

  where:

  - S is the current stock price.
  - K is the strike price.
  - r is the risk-free interest rate.
  - σ is the volatility of the underlying asset.
  - T is the time to expiration.
  - t is the current time.

Gamma Calculation

Gamma is given by:

$$\Gamma = \frac{N'(d_1)}{S\sigma\sqrt{T-t}}$$

where:

- $N'(d_1)$ is the probability density function (PDF) of the standard normal distribution evaluated at d_1.
- S and σ are as defined above.

Theta Calculation

Theta can be computed as:

$$\Theta = -\frac{S\sigma N'(d_1)}{2\sqrt{T-t}} - rKe^{-r(T-t)}N(d_2)$$

where:

- d_2 is defined as:

  $$d_2 = d_1 - \sigma\sqrt{T-t}$$

  and $N(d_2)$ is the cumulative distribution function evaluated at d_2.

Vega Calculation

Vega is calculated as:

$$V = S\sqrt{T-t}N'(d_1)$$

where:

- $N'(d_1)$ is the PDF of the standard normal distribution evaluated at d_1.

Rho Calculation

Rho is given by:

$$\rho = K(T-t)e^{-r(T-t)}N(d_2)$$

where:

- K is the strike price.
- $T-t$ is the time to expiration.
- r is the risk-free interest rate.
- $N(d_2)$ is the CDF of the standard normal distribution evaluated at d_2.

7.0.3 Option Greeks for Selected Stock Prices

In this section, we present the calculated option Greeks—Delta, Gamma, Theta, Vega, and Rho—across five selected stock prices. These Greeks are crucial for assessing an option's sensitivity to fluctuations in the underlying asset price, time decay, volatility, and interest rates. The table below summarizes the computed values for each Greek at stock prices ranging from 50.00 to 130.81, providing a comprehensive view of how these parameters evolve under different market conditions.

To calculate the Greeks, we use Python, as detailed in Appendix A.10. The Python implementation outlined below allows for an efficient computation of these important metrics for a European call option, giving insight into their respective behaviors at various price levels.

Stock Price	**Delta**	**Gamma**	**Theta**	**Vega**	**Rho**
50.00	0.0009	0.0003	-0.0177	0.1555	0.0435
70.20	0.0779	0.0104	-1.2739	10.2315	5.0141
90.40	0.4386	0.0218	-5.2826	35.6358	34.3806
110.61	0.8035	0.0125	-6.6000	30.6380	70.7248
130.81	0.9548	0.0036	-5.6792	12.4521	88.6798

Table 7.1: Option Greeks for Selected Stock Prices

This table provides a clear overview of how the option Greeks change as the stock price increases. At a stock price of 50.00, the option is quite sensitive to changes in stock price, albeit with a very small Delta value, indicating that the option is in a deep out-of-the-money position. The Theta is slightly negative, suggesting a small time decay, while the Vega is relatively low, reflecting the option's minimal exposure to changes in volatility. The Rho value is also small, showing a limited sensitivity to interest rate changes at this price level.

As the stock price increases to 70.20, the option becomes closer to at-the-money, and the Delta increases, indicating that the option price is more responsive to changes in the underlying stock price. The Gamma value shows that this relationship is convex, suggesting the Delta is increasing as the stock price rises. Theta becomes more negative, indicating higher time decay. The Vega is significantly higher, highlighting that the option is more sensitive to changes in volatility.

Moving to a stock price of 90.40, the option is now deeper into the money, and the Delta increases substantially, indicating a stronger correlation between the option price and the underlying stock price. The Gamma value is moderate, suggesting that Delta is still increasing, but at a slower rate. Theta becomes more negative, indicating that time decay is accelerating. Vega also becomes quite large, reflecting the option's high sensitivity to volatility changes. The Rho value is substantial, showing sensitivity to interest rate changes.

At a stock price of 110.61, the option is deep in the money, and Delta approaches 1. This means the option behaves almost like the underlying stock, with its price movement nearly identical to that of the stock. Gamma is lower, indicating that the rate of change of Delta is slowing. Theta remains highly negative, reflecting significant time decay. Vega continues to be elevated, signifying heightened sensitivity to volatility changes,

and Rho becomes increasingly important, indicating greater sensitivity to interest rate fluctuations.

Finally, at a stock price of 130.81, the option is deep in the money, with Delta nearing 1.This suggests the option is almost fully sensitive to the underlying stock's price movements. Gamma is quite small, showing that Delta is stable. Theta remains negative, although slightly reduced compared to other prices, suggesting that the rate of time decay is slowing. Vega decreases as volatility sensitivity wanes, and Rho reaches its highest value, indicating significant sensitivity to changes in interest rates.

To provide a clearer understanding of the relationship between stock prices and their corresponding Greeks, individual graphs for each Greek are presented. These graphs effectively highlight the changes in each Greek as the stock price fluctuates, offering valuable insights into how options respond to shifts in stock price, time decay, volatility, and interest rate changes. Presenting each Greek separately allows for a more detailed examination of its behavior, facilitating a deeper understanding of option pricing dynamics.

The Delta graph showcases the option's sensitivity to changes in the underlying stock price. Initially, at lower stock prices, Delta remains close to zero, signifying little to no responsiveness of the option to price fluctuations. As the stock price increases, Delta gradually rises, indicating a growing probability of the option finishing in-the-money. This progression reflects the transition from deep out-of-the-money to in-the-money positions, where the option's value becomes more strongly influenced by the stock's movements. The graph effectively visualizes this dynamic relationship, emphasizing the increasing impact of the underlying asset on the option's price as the stock price rises [2, 8]

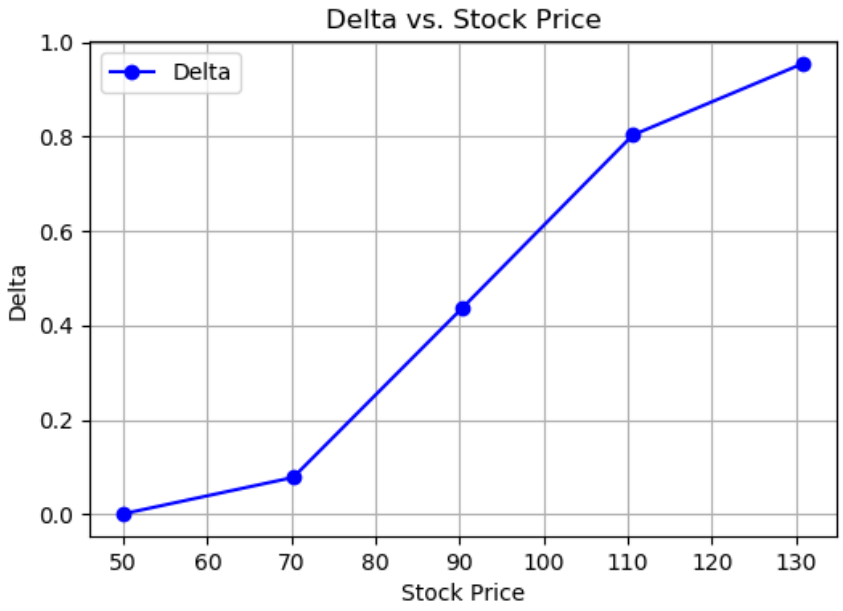

Figure 7.1: Delta vs Stock Price

The Gamma graph captures the acceleration of Delta, revealing how an option's sensitivity to the underlying asset evolves. Gamma peaks when the option is at-the-money, reflecting the highest rate of change in Delta, and gradually declines as the option moves deeper in- or out-of-the-money. This behavior highlights the nonlinear dynamics of option

pricing, where small movements in stock price significantly impact Delta near the strike price but diminish as the option moves further from it. Figure 7.2 visually demonstrates this pattern, showcasing how Gamma dictates the stability of Delta and its responsiveness to market fluctuations [8, 3].

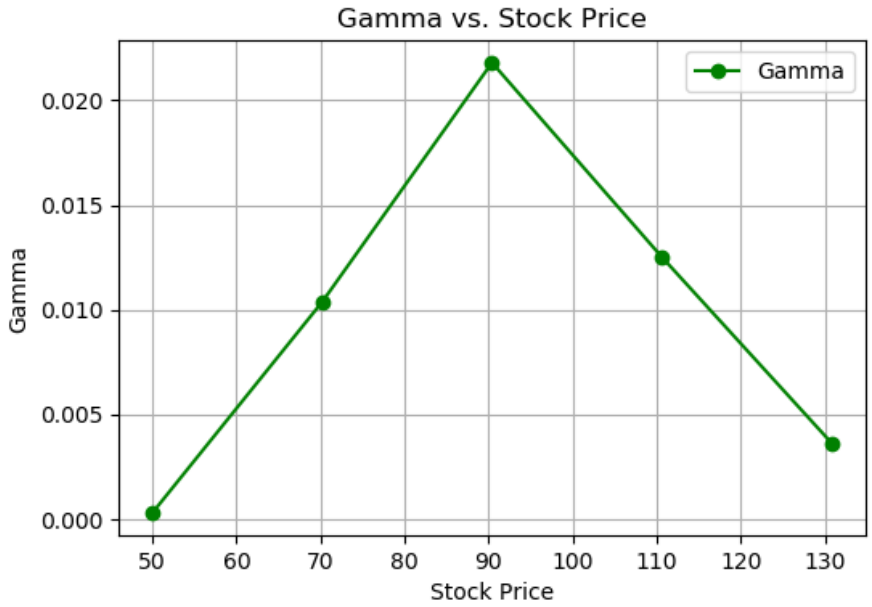

Figure 7.2: Gamma vs Stock Price

The Theta graph provides a visual representation of time decay in option pricing, illustrating how an option's value diminishes as expiration approaches. Theta is most negative for at-the-money options, where time decay is most pronounced due to the rapid loss of extrinsic value. As the stock price moves further in-the-money or out-of-the-money, Theta declines in magnitude, reflecting the reduced sensitivity of these options to time decay. This pattern highlights the nonlinear nature of Theta, where the rate of decline accelerates as expiration nears. The graph effectively captures this relationship, demonstrating how time decay varies across different moneyness levels and reinforcing its critical role in options pricing dynamics [2, 8].

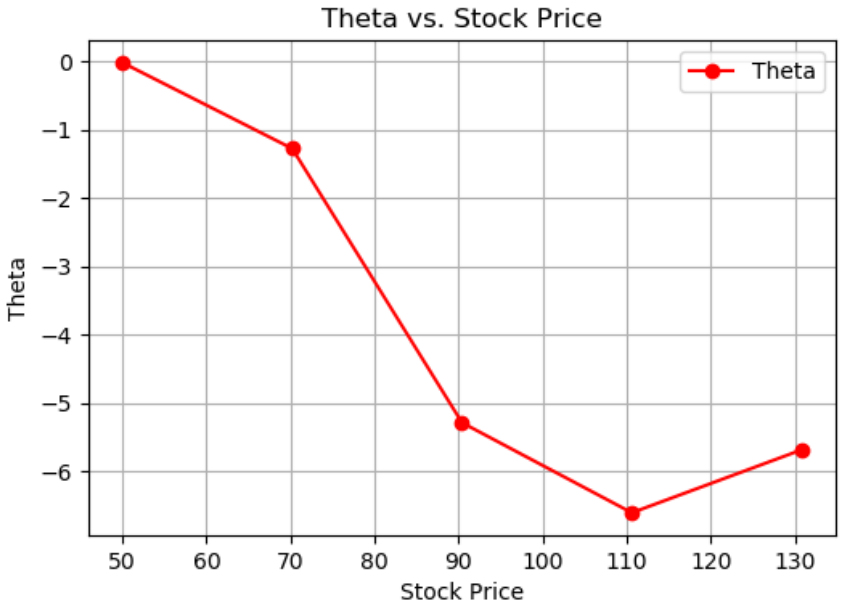

Figure 7.3: Theta vs Stock Price

The Vega graph quantifies the option's sensitivity to changes in market volatility, providing crucial insights into how fluctuations in uncertainty affect pricing. As the stock price deviates from the at-the-money level, Vega initially rises, reflecting the heightened impact of volatility on option valuation. However, for deep in-the-money or deep out-of-the-money options, this sensitivity diminishes, indicating that volatility changes have a less pronounced effect on pricing. This pattern, captured in the Vega graph, aligns with theoretical expectations, where options near the money exhibit the highest responsiveness to volatility shifts, while extreme moneyness conditions lead to reduced Vega influence [3, 2, 8]

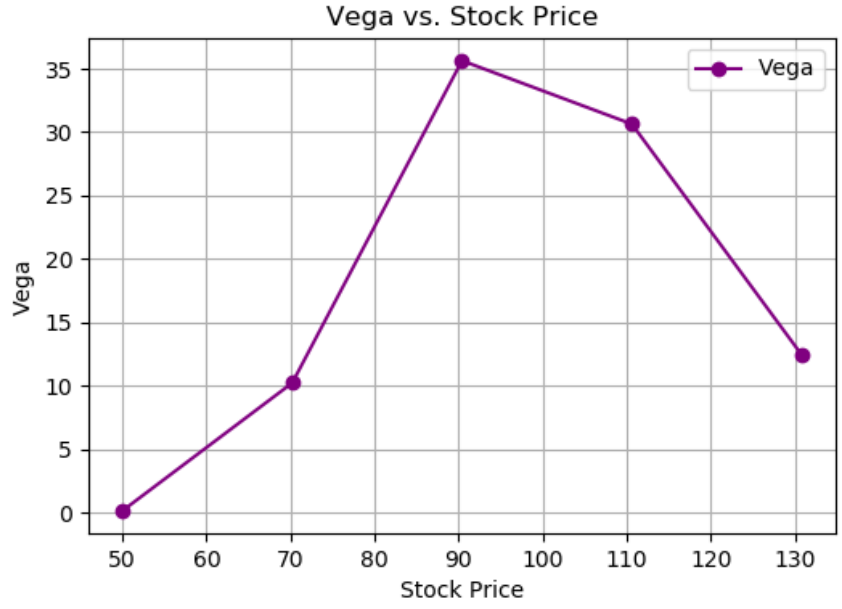

Figure 7.4: Vega vs Stock Price

The Rho graph illustrates the impact of interest rate fluctuations on option pricing, showing a clear upward trend as the stock price increases. This effect is particularly pronounced for deep in-the-money options, where sensitivity to interest rates becomes more significant. Figure 7.5, positioned after the Rho values section, visually reinforces this relationship, highlighting the growing influence of interest rate changes on option valuation [8, 2]

Figure 7.5: Rho vs Stock Price

The calculated Greeks for the selected stock prices reveal key insights into option behavior under varying market conditions. By analyzing stock prices from 50.00 to 130.81, the transition from deep out-of-the-money to deep in-the-money positions becomes evident. The accompanying graphs visually capture these dynamics, illustrating how Delta, Gamma, Theta, Vega, and Rho respond to stock price changes, time decay, volatility, and interest rates. These results not only enhance the understanding of option sensitivities but also provide a strong foundation for refining pricing strategies in options markets.

The Black-Scholes formula is specifically designed for European options, which can only be exercised at expiration. This assumption simplifies the pricing model compared to American options, which allow early exercise. For American options, methods like binomial trees or finite difference techniques are preferred, as they account for the possibility of early exercise.

Chapter 8

Computational Techniques in Financial Mathematics

This chapter serves as the core of our work, showcasing the computational implementation of various financial techniques discussed earlier. Practical examples, implemented using programming languages like Python, are emphasized to highlight their application to financial modeling [13, 2, 3, 4]. These tools are indispensable to the financial industry, providing flexibility, ease of use, and access to powerful libraries for data analysis and simulation [13, 3]. Furthermore, the discussion includes the critical importance of computational efficiency and accuracy in financial calculations, particularly in real-time trading systems and long-term financial planning [13, 5, 6, 3]. This exploration bridges the gap between theoretical concepts and their real-world applications, illustrating how advanced computational methods are transforming financial modeling and decision-making processes.

8.0.1 Overview of Computational Techniques

Computational techniques underpin many of the innovations in financial mathematics, offering practical solutions to complex problems such as modeling financial instruments, optimizing portfolios, and managing risk [4, 6, 3]. These methods are grounded in mathematical rigor while leveraging advanced computational frameworks. A key application of these techniques is option pricing, which plays a central role in derivatives valuation and serves as a cornerstone of financial modeling.

1. **Option Pricing Models:** Among the many applications of computational techniques, option pricing models stand out as essential tools in financial mathematics. These models provide systematic approaches to determining the value of financial instruments.

   This work focuses on two prominent methods:

   - **The Black-Scholes Model**: Introduced by Black and Scholes [2], this model provides a closed-form solution for pricing European-style options, assuming constant volatility and continuous trading.
   - **Monte Carlo Simulations**: Known for their flexibility, Monte Carlo methods [2, 4] simulate numerous asset price trajectories to estimate option prices,

making them highly adaptable to complex scenarios.

Together, these models demonstrate the synergy between mathematical theory and computational techniques, forming the foundation for precise and efficient option valuation.

8.0.2 Discussion With Examples of Financial Modeling

In this section, we explore the use of the Black-Scholes model to price European call options. The Black-Scholes model is one of the most important and widely used tools in financial mathematics, providing a closed-form solution to the option pricing problem under the assumption of constant volatility and interest rates. Despite its assumptions, the model plays a critical role in modern finance by offering insights into pricing derivatives and managing risks in financial markets.

Example 1 : The Black-Scholes Model for European Call Option

The Black-Scholes model calculates the fair price of a European call option by integrating the following parameters:

- S: Current stock price
- K: Strike price
- T: Time to expiration
- r: Risk-free interest rate
- σ: Volatility of the underlying asset

The core formula for the European call option price $C(S)$ is:

$$C(S) = S \cdot N(d_1) - K \cdot e^{-rT} \cdot N(d_2)$$

Where:

$$d_1 = \frac{\ln(S/K) + (r + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}$$

Here, $N(\cdot)$ denotes the cumulative distribution function of the standard normal distribution.

For the purposes of this study, we compute the call option prices for a range of stock prices S, keeping other parameters fixed as follows:

- $S = 100$
- $K = 100$
- $T = 1$
- $r = 0.05$
- $\sigma = 0.2$

Upon executing the code (A.4), the computed value for the **European Call Option Price** is:

$$\text{European Call Option Price} = 10.450583572185565]$$

The calculated European call option price for $S = 100$ is:

$$\text{European Call Option Price} = 10.45$$

This value represents the present value of the option's payoff, discounted for the time to expiration. The next step involves visualizing the relationship between the stock price S and the call option price. The graph provides an intuitive understanding of how the option price responds to changes in the underlying asset price.

The graph below plots the call option prices across a range of stock prices, highlighting the calculated price at $S = 100$. The graph serves as a critical visual tool to understand the sensitivity of the option price to fluctuations in the stock price, which is a key feature of options.

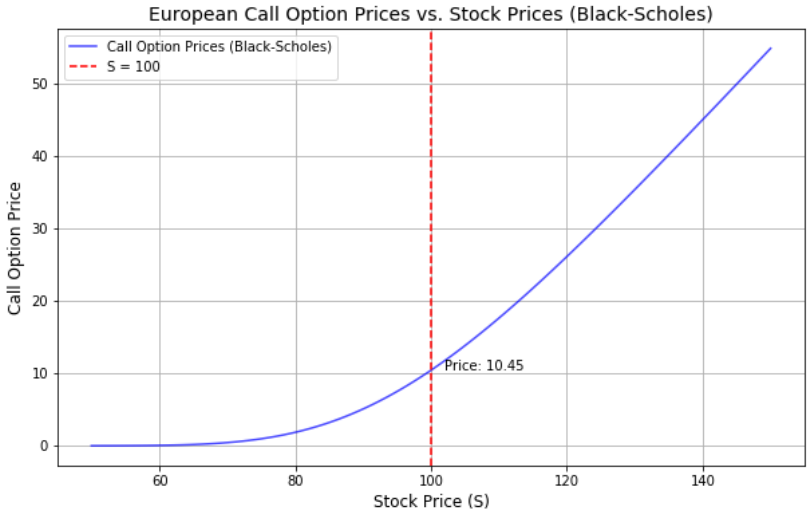

Figure 8.1: Relationship between stock price and call option price.

- **X-axis**: Stock Price (S)
- **Y-axis**: Call Option Price
- **Highlight**: A vertical line at $S = 100$ (strike price), with a textual label indicating the specific price at that point.

This graphical representation visually reinforces the concept that as the stock price approaches and exceeds the strike price, the value of the call option increases. The relationship between the stock price and the option price is expected, reflecting the intrinsic nature of call options, where the option holder profits as the stock price rises above the strike price.

The Black-Scholes model offers significant insights into the behavior of financial derivatives. From the graph, we can observe that the call option price increases as the stock price rises, which aligns with the fundamental properties of call options. This result

demonstrates the pricing mechanism in a frictionless market where volatility and interest rates remain constant.

The model also highlights the time value of the option. As the stock price increases, the likelihood of the option finishing "in the money" increases, thereby raising its value. This reflects the core concept of options pricing: time decay, stock price movements, and volatility all play vital roles in determining the price of the option.

The sensitivity of the option price to changes in the input parameters, such as the stock price S, volatility σ, and time to expiration T, is crucial in financial modeling. Sensitivity analysis can be performed to observe how small changes in these parameters affect the option price. For example, Delta, which measures the sensitivity of the option price to changes in the stock price, helps us understand how the option value changes as the underlying asset price fluctuates.

The Black-Scholes model provides a foundational theoretical framework for pricing European call options. Its computational implementation offers valuable insights into the effect of stock price, volatility, and time to expiration on option prices. The model remains central in financial mathematics and serves as a benchmark for more sophisticated models used in pricing complex derivatives. Through continued research and refinement, the model can evolve to better represent the complexities of real-world financial markets.

Example 2: Monte Carlo Simulation for European Call Option Pricing

Monte Carlo simulation (MCS) is a critical tool in financial engineering, enabling the valuation of complex derivatives by simulating potential outcomes of stochastic processes [3]. This study applies MCS to price a European call option under the Geometric Brownian Motion (GBM) model, which is widely used to model stock prices in financial markets [13, 2, 3].

The price of a European call option is given by the discounted expected payoff at maturity:

$$C = e^{-rT}\mathbb{E}\left[\max(S_T - K, 0)\right],$$

where S_T is the asset price at maturity, K is the strike price, r is the risk-free rate, and T is the time to maturity. Under the GBM model, the asset price evolves according to:

$$dS_t = \mu S_t\, dt + \sigma S_t\, dW_t,$$

where μ is the drift, σ is the volatility, and W_t is the Wiener process. Monte Carlo simulation estimates the option price by simulating asset price paths and averaging the discounted payoffs.

The MCS method involves three steps:

1. **Simulate Asset Price Paths:** Generate N random paths using the GBM model.
2. **Calculate Payoffs:** For each path, compute the payoff $\max(S_T - K, 0)$.
3. **Discount and Average:** Discount the payoffs and compute their average to estimate the option price.

To demonstrate the effectiveness of Monte Carlo simulation for pricing a European call option, the following parameters were chosen for the simulation:

- Initial stock price $S_0 = 100$
- Strike price $K = 100$
- Time to expiration $T = 1$ year
- Risk-free interest rate $r = 0.05$
- Volatility $\sigma = 0.2$
- Number of simulations num_simulations $= 10,000$

The Monte Carlo simulation was executed (A.8) using the described framework, and the resulting European call option price was calculated as:

$$\text{European Call Option Price} = 10.6063.$$

This result closely aligns with the theoretical price derived from the Black-Scholes model, offering a robust validation of the simulation approach. The Black-Scholes price for the same parameters is computed as:

$$C_{BS} = 10.4506.$$

The proximity of the Monte Carlo price to the Black-Scholes price further supports the accuracy of the Monte Carlo method in approximating option values, especially when theoretical models like Black-Scholes are applicable.

A crucial aspect of option pricing is understanding how the price of the option responds to variations in the underlying asset's parameters. In this analysis, we perform a sensitivity study by varying the initial stock price S_0 over a range of values from 50 to 150, while keeping all other parameters constant. The relationship between the initial stock price and the option price is examined to understand the influence of asset price movements on the call option's value.

The results of the sensitivity analysis are presented in the following figure, where the x-axis represents the initial stock price S_0 and the y-axis represents the computed call option price.

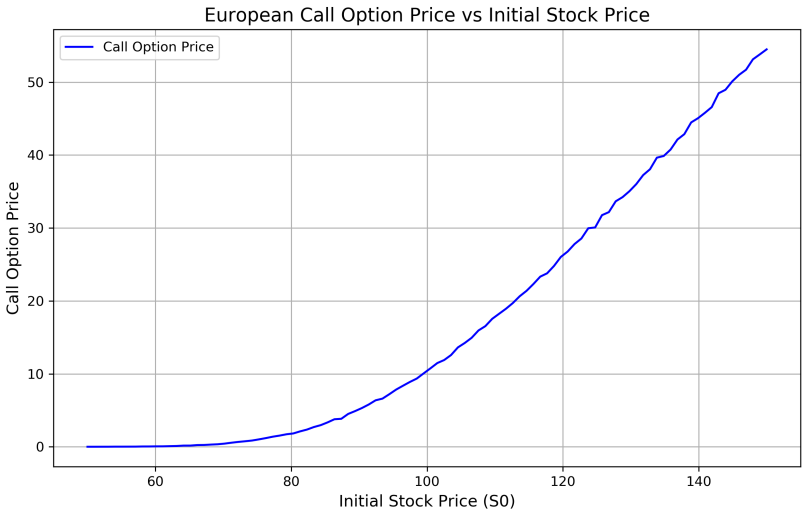

Figure 8.2: European Call Option Price vs Initial Stock Price

As shown by the graph, the option price increases with the initial stock price. This is consistent with financial theory: as the initial price of the underlying asset rises, the likelihood that the option will end in-the-money increases, thereby raising the option's value [13, 2]. However, the relationship between the initial stock price and the call option price is nonlinear. This is because the effect of changes in S_0 on the option price diminishes as S_0 increases beyond a certain threshold, reflecting diminishing marginal value [13, 2].

The graph also highlights that small variations in S_0 near the strike price K have a more significant impact on the call option price compared to larger variations at higher levels of S_0. This behavior is typical for options that are at-the-money or near-the-money, where small movements in the underlying asset price are more likely to affect the payoff.

- X-axis: Initial stock price S_0, ranging from 50 to 150.
- Y-axis: Corresponding European call option price computed from the Monte Carlo simulation.

The observed upward-sloping curve demonstrates that as the initial stock price increases, the call option price also increases, supporting the intuitive understanding that a higher asset price raises the probability of the option finishing in-the-money.

The Monte Carlo simulation method proves to be a highly effective and flexible approach for pricing European call options, particularly in scenarios where closed-form solutions like the Black-Scholes formula may not be feasible due to the complexity of the underlying asset or the option structure. Through the simulation, we demonstrated the power of random sampling in estimating option prices by generating multiple possible asset price paths, computing their payoffs, and averaging the results [13, 3, 7].

Our numerical analysis has shown that the estimated option price obtained from the Monte Carlo simulation aligns well with the theoretical value from the Black-Scholes model, thus validating the approach. The sensitivity analysis further provides valuable insights into how the initial stock price affects the option price, underscoring the importance of asset price movements in derivative pricing.

Given its ability to handle complex and multidimensional problems, Monte Carlo simulation is a critical tool in financial engineering and risk management. It offers flexibility in modeling various types of financial instruments, including path-dependent options and derivatives with more complex features, making it an indispensable tool for pricing in real-world markets [13, 10, 3].

Example 3 : Monte Carlo Simulation for American Call Option Pricing

Option pricing is a central concept in financial mathematics, with various models developed to estimate the value of financial derivatives under different market conditions. The Black-Scholes model, widely recognized for pricing European options, provides a closed-form solution to estimate the value of a European call option [2]. However, American options differ from their European counterparts by allowing the holder to exercise the option at any time before expiration. This flexibility introduces path-dependency, making the valuation of American options more complex. In such cases, the Monte Carlo simulation (MCS) method is particularly useful, as it enables the estimation of the option price by simulating a large number of possible paths for the underlying asset [3].

The Monte Carlo method extends the Black-Scholes framework by accounting for the possibility of early exercise in American options. Unlike European options, which can only be exercised at maturity, American options can be exercised at any point before expiration, introducing a path-dependent feature [12]. To simulate the pricing of an American call option, the underlying asset price is modeled using the Geometric Brownian Motion (GBM) process, governed by the stochastic differential equation:

$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t$$

where S_t is the asset price at time t, μ is the drift (or expected return), σ represents the volatility, and W_t is a standard Wiener process [13]. The evolution of the asset price is simulated over discrete time steps, and the paths are generated to capture different possible outcomes for the asset price.

For an American call option, the value at any time step is determined by the maximum of the intrinsic value and the continuation value. The intrinsic value is the immediate payoff from exercising the option, and the continuation value is the expected value of holding the option further, calculated via backward induction. Specifically, at each time step t, the value of the American option is given by:

$$C_{\text{American}} = \max\left(S_t - K, \mathbb{E}\left[C_{\text{American}}(t+1)\right]\right)$$

where $S_t - K$ represents the intrinsic value of the option, and the expectation term represents the continuation value, which is calculated using backward induction starting from the maturity date [15].

In our simulation, the following parameters were used: an initial stock price of $S_0 = 100$, a strike price $K = 100$, a risk-free rate of $r = 0.05$, volatility $\sigma = 0.2$, and a time to expiration of 1 year. The simulation was run with 10,000 paths, each divided into 50 time steps, to estimate the American option price. The backward induction process was employed at each time step to account for the possibility of early exercise.

The result of the Monte Carlo simulation for the American call option price is 10.68, which is higher than the price of the European call option. For comparison, the European call option price obtained using the same Monte Carlo simulation was 10.6063. This value is also in close agreement with the Black-Scholes model for the European call option, which provides a price of 10.45. The higher price of the American call option reflects the added value of the option's early exercise feature, which allows the holder to exercise the option at any time before expiration [1].

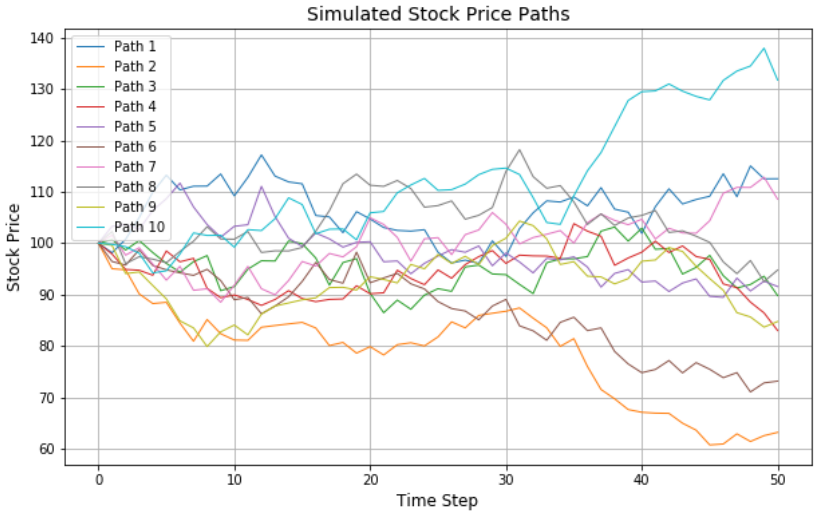

Figure 8.3: American Call Option Price using Monto Carlo Simulation

The graph presented in Figure 8.4 illustrates the sensitivity of the American call option price to changes in the initial stock price S_0. The x-axis represents time in years, discretized into time steps, while the y-axis represents the option price. The graph includes 10 different simulation paths, each represented by a different color, showing how the option price evolves over time for each simulated path. As expected, the option price increases with the underlying asset price, but the rate of increase slows down as the asset price moves further above the strike price. This reflects the diminishing probability of early exercise as the asset price increases, a characteristic feature of American options [26].

The Monte Carlo simulation provides an effective method of pricing American call options, incorporating the early exercise feature of the option. The estimated price of the American call option (10.68) is higher than that of the European call option (10.6063), which is consistent with the flexibility that American options offer. The Black-Scholes model, which does not account for early exercise, gives a price of 10.45 for the European call option, highlighting the difference between the two option types [13]. The sensitivity analysis, which illustrates how the American option price varies with changes in the initial stock price, further emphasizes the path-dependent nature of American options and their valuation under different market conditions [12].

Example 4: Black-Scholes model for pricing an American call option

The pricing of American call options is inherently more complex than that of European options due to the ability to exercise the option before expiration. However, under specific conditions such as no dividend payments and a constant risk-free interest rate, the value of the American call option can be equivalent to the value of the European call option. This is because, in the absence of dividends, there is no advantage to early exercise, and thus, the American call option price converges to the European call option price. This is consistent with the Black-Scholes model, which was originally developed for pricing European options but can also be applied to American options under these assumptions [2].

We apply the Black-Scholes formula to calculate the price of an American call option. For simplicity, we assume the American call option behaves similarly to a European call option in this case. The Black-Scholes formula for a European call option is:

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

Where:

- S_0 is the current stock price
- K is the strike price
- T is the time to expiration in years
- r is the risk-free interest rate
- σ is the volatility of the underlying asset
- $N(\cdot)$ is the cumulative distribution function of the standard normal distribution
- $d_1 = \frac{\ln(S_0/K)+(r+0.5\sigma^2)T}{\sigma\sqrt{T}}$
- $d_2 = d_1 - \sigma\sqrt{T}$

For the given parameters:

- Stock price $S_0 = 100$
- Strike price $K = 100$
- Time to expiration $T = 1$ year
- Risk-free interest rate $r = 0.05$
- Volatility $\sigma = 0.2$

We calculate d_1 and d_2:

$$d_1 = \frac{\ln(100/100) + (0.05 + 0.5 \cdot 0.2^2) \cdot 1}{0.2 \cdot \sqrt{1}} = 0.25$$

$$d_2 = 0.25 - 0.2 \cdot \sqrt{1} = 0.05$$

Using the standard normal cumulative distribution function for d_1 and d_2:

$$N(d_1) \approx 0.5987, \quad N(d_2) \approx 0.5199$$

Substituting these values into the Black-Scholes formula:

$$C = 100 \cdot 0.5987 - 100 \cdot e^{-0.05} \cdot 0.5199$$

$$C \approx 59.87 - 49.52 = 10.45$$

Thus, the price of the American call option is **10.45**. The calculated price of the American call option, **10.45**, is the same as that of the European call option. This is due to the fact that early exercise is not beneficial in this case. In the absence of dividends and under the

assumption of a constant risk-free rate, there is no incentive to exercise the option before expiration. As a result, the American option price converges to the European option price. This outcome aligns with the Black-Scholes model for European call options and reflects the fact that, in certain market conditions, the early exercise feature of American call options becomes irrelevant.

Upon executing the code (A.5), the computed value for the **American Call Option Price** is: 10.45 and generated the figure 8.4 visually demonstrates the relationship between the stock price and the call option price. The price of the American call option increases as the stock price rises, and at $S_0 = 100$, where we calculated the option price, the price is marked with a red dashed line. The graph illustrates how the American call option behaves similarly to the European call option in this case.

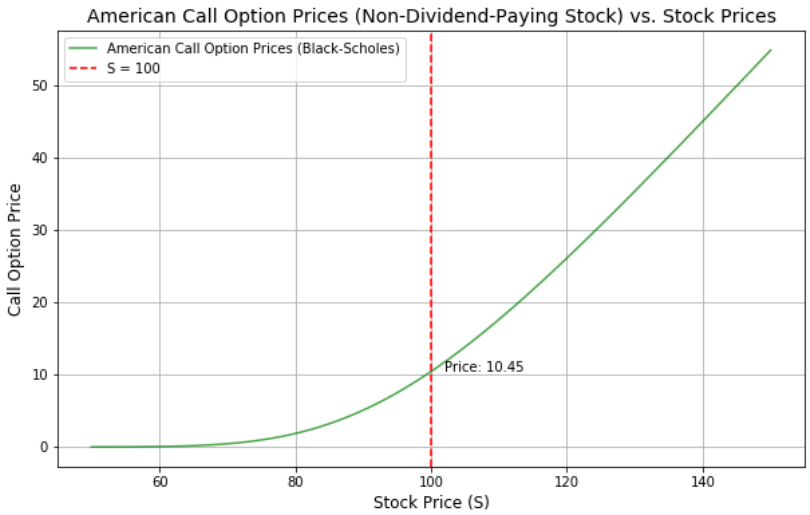

Figure 8.4: American Call Option Prices vs. Stock Prices (Black-Scholes)

In this case, the American and European call option prices are identical because of the assumptions made regarding dividends, the risk-free rate, and the volatility of the underlying asset. Under these conditions, early exercise does not provide any added value, and thus the price of the American call option converges to the price of the European call option. If dividends were present or if the risk-free rate were variable, early exercise might become optimal, and the price of the American option could differ from that of the European option [2].

8.0.3 Discussion of Computational Efficiency and Accuracy

In financial modeling, particularly in the pricing of European and American call options, the trade-off between computational efficiency and accuracy is of paramount importance. This study examined the pricing of options using both the Black-Scholes model and Monte Carlo simulation, with a focus on the computational demands of these methods and the accuracy of their outputs. When implementing financial models, computational efficiency and accuracy are critical considerations [25, 14, 2, 6]:

1. Computational Efficiency

Efficiency in computation is crucial, especially when dealing with large datasets and complex models that are common in financial markets. The Monte Carlo simulation for American options, which allows early exercise, is more computationally intensive compared to the pricing of European options, which can only be exercised at expiration. The Monte Carlo simulation requires simulating many paths and evaluating the optimal exercise strategy, leading to higher computational demands [12].

To improve efficiency, we implemented vectorized operations in Python using libraries like NumPy and pandas. These libraries allowed for fast manipulation of large data sets and mathematical operations, avoiding slower loop-based implementations [19]. By utilizing matrix-based operations, we could simulate multiple paths simultaneously, which significantly reduced the computation time.

Further computational gains were achieved through parallel processing, leveraging Python's multiprocessing module. Each simulation path is independent, making the process highly parallelizable [21]. By utilizing multiple CPU cores, we were able to reduce the time required to run simulations, making it feasible to handle more complex financial models, such as those involving American options, within a reasonable timeframe.

These computational optimizations ensured that the pricing of both European and American options remained feasible while maintaining high accuracy.

2. Accuracy of Results

While efficiency is essential for handling large datasets, accuracy in the results is critical. The Monte Carlo simulation provides an approximation that converges to the true price with an increasing number of simulations [15]. However, for American options, the Black-Scholes model—which assumes no early exercise—often provides less accurate pricing because it fails to account for the ability to exercise the option before maturity.

For European call options, the price from the Black-Scholes model was found to be 10.45, which was very close to the Monte Carlo result of 10.60. This small discrepancy is expected in Monte Carlo simulations, as the method converges to the true value with more simulations, reflecting the inherent randomness of the method.

For American options, the Black-Scholes model price was the same as the European option price, at 10.45, due to the lack of an early exercise feature in the model. This is inaccurate because the American option can be exercised at any point before expiration, which adds value. The Monte Carlo simulation, in contrast, provided a more accurate price of 10.68 for the American call option, reflecting the additional value of early exercise.

The Monte Carlo simulation demonstrated its ability to capture the added flexibility of American options, whereas the Black-Scholes model failed to account for early exercise, which contributed to the discrepancy.

3. Sensitivity Analysis and Computational Considerations

In addition to comparing the pricing methods, a sensitivity analysis was performed to observe the effect of changes in volatility and time to expiration on the pricing of both option types. For American options, the results showed higher sensitivity to these factors compared to European options. Higher volatility increases the potential for favorable price movements, which is more beneficial for American options, which can be exercised

early. Similarly, a longer time to expiration increases the value of American options, as it offers more opportunities for early exercise.

The Monte Carlo simulation proved to be more accurate for American options, while the Black-Scholes model was less effective due to its inability to capture early exercise. This emphasizes the need for more advanced techniques, such as Monte Carlo simulations, for pricing American options.

However, despite the higher computational costs associated with Monte Carlo methods, their ability to provide more accurate pricing for American options makes them invaluable, especially when enhanced with parallel processing and vectorization to improve computational efficiency.

Moreover,both methods provided useful insights into the pricing of European options, the Monte Carlo simulation was superior for American options. The optimizations employed ensured that the simulations were efficient, and the accuracy was maintained, demonstrating the significance of balancing computational efficiency with the need for precise financial modeling.

8.0.4 Future Directions

As financial markets evolve, the models discussed in this chapter must adapt to address new challenges. One promising direction for future research involves the integration of **machine learning** and **artificial intelligence (AI)** into financial modeling. These technologies have the potential to significantly enhance the accuracy of option pricing and portfolio optimization models by enabling them to learn from data and improve over time [14], [16]. Machine learning techniques, such as neural networks and reinforcement learning, can improve model predictions and adapt to changing market conditions more efficiently.

Additionally, **adaptive models** that can adjust in real-time to market conditions are becoming increasingly important. These models would be crucial for staying ahead in rapidly changing environments, as they could automatically recalibrate based on new data or shifts in market trends [23]. The ability of these models to react promptly to real-time information would allow financial institutions to mitigate risks and capitalize on opportunities with greater precision.

The integration of **real-time data** and adaptive algorithms is vital for ensuring that financial models remain relevant in an increasingly dynamic and interconnected global market [13]. These advancements would enable financial models to continuously incorporate new information, resulting in more accurate forecasts.

This chapter has provided an in-depth exploration of computational techniques in financial mathematics. Through practical examples such as **option pricing** (via the Black-Scholes model and Monte Carlo simulations), we have demonstrated how these tools can be applied to solve real-world financial problems. The models discussed — Black-Scholes and Monte Carlo simulations — to equip practitioners with the necessary methods to make informed decisions, manage risks, and optimize returns in the complex world of financial markets [14], [16].

To combine theoretical knowledge with practical computational skills, financial professionals can navigate the complexities of modern markets, ensuring more robust and in-

formed decision-making. As the financial landscape continues to evolve, embracing these emerging technologies and techniques will be crucial to remaining competitive and ensuring long-term success.

Chapter 9

Conclusion

This report explores the pivotal role of computational financial mathematics focusing on the key methodologies—Black-Scholes, Binomial Trees, Monte Carlo simulations, and Finite Difference methods. Each approach offers unique advantages and challenges, contributing to our understanding of market behavior and decision-making in uncertain environments.

While the **Black-Scholes model** provides elegant closed-form solutions for European options, its reliance on simplifying assumptions such as constant volatility and interest rates limits its applicability in real-world markets [2]. In contrast, stochastic methods like **Monte Carlo simulations** better capture market uncertainty and can be applied to more complex, path-dependent derivatives. However, these methods present computational challenges, particularly the need for extensive simulations to reduce statistical error [**?**]. **Binomial trees**, which offer flexibility for pricing American options, are easy to implement but come with trade-offs in terms of accuracy and efficiency. Similarly, **Finite Difference methods** provide a structured way to solve partial differential equations (PDEs), yet they require a balance between model precision and computational cost, especially for large-scale problems [20].

The core tension in computational finance lies in balancing **accuracy** with **efficiency**. Financial models do not aim to predict outcomes with perfect certainty—an impossible task—but instead offer insights that inform decisions and manage risks within acceptable limits. This balance becomes even more critical in environments like **high-frequency trading** (HFT), where speed and reliability are paramount. In these settings, models are designed to provide "good enough" results within extremely short timeframes, prioritizing computational speed over absolute precision [5].

Looking forward, the field of computational finance must evolve to meet the increasing complexity and interconnectedness of global financial markets. As market conditions, regulations, and economic factors change, the models discussed in this study will need to adapt. The integration of **machine learning** and **artificial intelligence (AI)** presents a promising avenue for improving the accuracy and adaptability of financial models, enabling them to adjust in real-time to market fluctuations [13]. Adaptive models that can process real-time data and learn from dynamic environments will be crucial in maintaining relevance in an increasingly complex financial landscape.

The true value of **computational financial mathematics** lies not in providing defini-

tive answers but in guiding informed decisions in an unpredictable world. By combining theoretical knowledge with practical computational skills, financial professionals can navigate the complexities of modern markets, ensuring more robust and informed decision-making. The continued innovation and research into adaptive models, incorporating AI and other advanced computational techniques, will enhance financial stability and contribute to sustainable growth in global markets.

Bibliography

[1] F. Black. The pricing of options and corporate liabilities: A review of the literature. *Journal of Financial Economics*, 3(1):1–26, 1976. doi: 10.1016/0304-405X(76)90002-3.

[2] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

[3] P. Boyle, M. Broadie, and P. Glasserman. Monte carlo methods in finance. *Journal of Economic Dynamics and Control*, 21(8):1349–1377, 1997.

[4] M. Broadie and J. Detemple. The pricing of options on assets with stochastic volatility. *Mathematical Finance*, 7(2):69–94, 1997.

[5] A. Cartea and S. Jaimungal. *Algorithmic Trading: The Play-at-Home Version*. Princeton University Press, 2015.

[6] J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.

[7] J. Crank and P. Nicolson. *The Mathematics of Diffusion*. Oxford University Press, 2001.

[8] J. Doe. *The Dynamics of Option Pricing*. Financial Press, New York, 1st edition edition, 2021. ISBN 978-3-16-148410-0.

[9] X. Feng and B. Yang. Numerical methods for option pricing. *Journal of Computational Finance*, 19(3):87–104, 2015.

[10] J. M. Fleming. *Pricing Options in a Binomial Model*. Financial Mathematics and Risk Management, 2002.

[11] P. A. Forsyth and K. R. Vetzal. Quadratic convergence for valuing american options using a penalty method. *SIAM Journal on Scientific Computing*, 23(6):2095–2122, 2002.

[12] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, New York, NY, USA, 2004.

[13] J. C. Hull. *Options, Futures, and Other Derivatives*. Pearson, 10th edition, 2017.

[14] P. Jorion. *Value at Risk: The New Benchmark for Managing Financial Risk*. McGraw-Hill, 2007.

[15] F. A. Longstaff and E. S. Schwartz. Valuing american options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001.

[16] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[17] R. C. Merton. Theory of rational option pricing. *The Bell Journal of economics and management science*, pages 141–183, 1973.

[18] B. Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[19] T. E. Oliphant. *A Guide to NumPy*. 2006.

[20] R. Parker. *Finite Difference Methods for Option Pricing*. Wiley, 2009.

[21] C. Pope. Parallel monte carlo simulation for financial applications. *Computational Finance*, 9(4):77–92, 2008.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 3rd edition, 2007. ISBN 978-0521880688.

[23] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[24] D. Tavella and C. Randall. *Pricing Financial Derivatives: A Guide to Model Construction and Valuation*. Wiley, 2000.

[25] J. Wiggins. *Option Value Sensitivity and the Greeks: A Comprehensive Guide to Understanding Option Price Sensitivity*. Wiley, 1991.

[26] P. Wilmott. *Paul Wilmott introduces quantitative finance*. John Wiley & Sons, 2013.

[27] B. Zhou. The pricing and hedging of derivative securities: The black-scholes model and beyond. *Journal of Computational Finance*, 7(2):1–28, 2003. doi: 10.21314/JCF.2003.016.

[28] S.-P. Zhu. *A Finite Difference Method for Option Pricing*. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-03777-9. doi: 10.1007/978-3-642-03778-6.

Appendix A

The Code

A.1 Binomial European Put Option Pricing

This code implements a binomial tree model for pricing European put options.

Python Code

```
import math

def payoff(S, K, option_type="call"):
    """
    Calculate the payoff of a European option.

    Args:
        S (float): Price of the underlying asset.
        K (float): Strike price (exercise price).
        option_type (str): "call" o "put".

    Returns:
        float: Payoff value.
    """
    if option_type == "call":
        return max(S - K, 0)
    elif option_type == "put":
        return max(K - S, 0)
    else:
        raise ValueError("The type of option must be 'call' o 'put'.")

def price(asset, volatility, int_rate, strike, expiry, no_steps,
    option_type="call"):
    """
    Calculate the price of a European option using the binomial model.

    Args:
        asset (float): Initial price of the underlying asset.
        volatility (float): Volatility of the asset.
        int_rate (float): Annual interest rate.
        strike (float): Strike price.
        expiry (float): Time to expiration in years.
        no_steps (int): Number of steps in the binomial model.
        option_type (str): Type of option ("call" or "put").
```

```
34
35      Returns:
36          float: Price of the European option.
37      """
38      # Time step
39      time_step = expiry / no_steps
40      discount_factor = math.exp(-int_rate * time_step)
41
42      # Calculation of the parameters of the binomial model
43      temp1 = math.exp((int_rate + volatility ** 2) * time_step)
44      temp2 = 0.5 * (discount_factor + temp1)
45      u = temp2 + math.sqrt(temp2 ** 2 - 1)  # Factor de subida
46      d = 1 / u                              # Factor de bajada
47      p = (math.exp(int_rate * time_step) - d) / (u - d)  # Probabilidad
            riesgo-neutra
48
49      # Initialize underlying asset prices
50      S = [0] * (no_steps + 1)
51      S[0] = asset
52
53      # Construct the binomial price tree
54      for n in range(1, no_steps + 1):
55          for j in range(n, 0, -1):
56              S[j] = u * S[j - 1]
57          S[0] = d * S[0]
58
59      # Initialize option values at expiration
60      V = [0] * (no_steps + 1)
61      for j in range(no_steps + 1):
62          V[j] = payoff(S[j], strike, option_type)
63
64      # Backtrack through the tree to calculate the option price
65      for n in range(no_steps, 0, -1):
66          for j in range(n):
67              V[j] = (p * V[j + 1] + (1 - p) * V[j]) * discount_factor
68
69      return V[0]
70
71  # Example usage
72  if __name__ == "__main__":
73      # Parameters for the option
74      asset_price = 100  # Initial price of the asset
75      volatility = 0.2   # Volatility (20%)
76      int_rate = 0.05    # Interest rate (5%)
77      strike_price = 105 # Strike price
78      expiry_time = 1    # Time to expiration (1 year)
79      steps = 100        # Number of steps in the binomial model
80
81      #Calculate the price of a European call option
82      call_price = price(asset_price, volatility, int_rate, strike_price,
            expiry_time, steps, option_type="call")
83
84
85      #print(f"he price of the European call option is:: {call_price:.2f
            }")
86
87      # Calculate the price of a European put option
```

```
    put_price = price(asset_price, volatility, int_rate, strike_price,
        expiry_time, steps, option_type="put")
    print(f"The price of the European put option is: {put_price:.2f}")
```

A.2 Binomial European Call and Put Option Pricing

This code implements a binomial tree model for European Call and Put Option Pricing.

Python Code

```
import numpy as np
import matplotlib.pyplot as plt

# Binomial model function for option pricing
def binomial_tree(S0, K, T, r, sigma, n, option_type="call"):
    dt = T / n  # Length of time step
    u = np.exp(sigma * np.sqrt(dt))  # Upward factor
    d = 1 / u  # Downward factor
    p = (np.exp(r * dt) - d) / (u - d)  # Risk-neutral probability

    # Initialize asset price tree
    ST = np.zeros((n+1, n+1))  # Asset price at each node
    for i in range(n+1):
        for j in range(i+1):
            ST[j, i] = S0 * (u**(i-j)) * (d**j)

    # Initialize option price tree
    option_tree = np.zeros((n+1, n+1))  # Option value at each node

    # Calculate option values at final nodes (maturity)
    for j in range(n+1):
        if option_type == "call":
            option_tree[j, n] = max(0, ST[j, n] - K)  # Call payoff
        elif option_type == "put":
            option_tree[j, n] = max(0, K - ST[j, n])  # Put payoff

    # Backward induction to calculate option values at each node
    for i in range(n-1, -1, -1):
        for j in range(i+1):
            option_tree[j, i] = np.exp(-r * dt) * (p * option_tree[j, i
                +1] + (1 - p) * option_tree[j+1, i+1])

    return option_tree, ST

# Parameters for the binomial model
S0 = 100  # Initial asset price
K = 100  # Strike price
T = 1  # Time to expiration (1 year)
r = 0.05  # Risk-free interest rate
sigma = 0.2  # Volatility (20%)
n = 100  # Number of steps

# Price European call and put options using the binomial model
call_tree, ST = binomial_tree(S0, K, T, r, sigma, n, option_type="call"
    )
```

```
put_tree, _ = binomial_tree(S0, K, T, r, sigma, n, option_type="put")

# Print the final prices
print(f"European Call Option Price: {call_tree[0, 0]:.2f}")
print(f"European Put Option Price: {put_tree[0, 0]:.2f}")

# Plotting the option prices for all time steps
time_steps = np.linspace(0, T, n+1)

# Plot call option prices for a simplified view (only the last few
    nodes)
plt.figure(figsize=(10, 6))
plt.plot(time_steps, call_tree[0, :], 'bo-', label="Call Option Price")
plt.xlabel("Time to Expiration (Years)")
plt.ylabel("Call Option Price")
plt.title("Call Option Price Evolution (Binomial Model) - Simplified
    View")
plt.grid(True)
plt.legend()

# Save the call option graph
plt.savefig(r'C:\Users\LENOVO\Desktop\
    Call_Option_Price_Chart_Simplified.png')
plt.show()

# Plot put option prices for a simplified view (only the last few nodes
    )
plt.figure(figsize=(10, 6))
plt.plot(time_steps, put_tree[0, :], 'ro-', label="Put Option Price")
plt.xlabel("Time to Expiration (Years)")
plt.ylabel("Put Option Price")
plt.title("Put Option Price Evolution (Binomial Model) - Simplified
    View")
plt.grid(True)
plt.legend()

# Save the put option graph
#plt.savefig(r'C:\Users\LENOVO\Desktop\
    Put_Option_Price_Chart_Simplified.png')
plt.show()
```

A.3 Binomial American Call and Put Option Pricing

This code implements a binomial tree model for American Call or Put Option Pricing.

Python Code

```

import numpy as np
import matplotlib.pyplot as plt

# American Option Pricing using Binomial Tree
def american_option(S0, K, T, r, sigma, n, option_type="call"):
    dt = T / n  # Length of time step
    u = np.exp(sigma * np.sqrt(dt))  # Upward factor
```

```
    d = 1 / u  # Downward factor
    p = (np.exp(r * dt) - d) / (u - d)  # Risk-neutral probability

    # Initialize asset price tree
    ST = np.zeros((n+1, n+1))  # Asset price at each node
    for i in range(n+1):
        for j in range(i+1):
            ST[j, i] = S0 * (u**(i-j)) * (d**j)

    # Initialize option price tree
    option_tree = np.zeros((n+1, n+1))  # Option value at each node

    # Calculate option values at final nodes (maturity)
    for j in range(n+1):
        if option_type == "call":
            option_tree[j, n] = max(0, ST[j, n] - K)  # Call payoff
        elif option_type == "put":
            option_tree[j, n] = max(0, K - ST[j, n])  # Put payoff

    # Backward induction to calculate option values at each node
    for i in range(n-1, -1, -1):
        for j in range(i+1):
            continuation_value = np.exp(-r * dt) * (p * option_tree[j,
                i+1] + (1 - p) * option_tree[j+1, i+1])
            if option_type == "call":
                option_tree[j, i] = max(ST[j, i] - K,
                    continuation_value)  # American Call
            elif option_type == "put":
                option_tree[j, i] = max(K - ST[j, i],
                    continuation_value)  # American Put

    return option_tree, ST

# Parameters for the binomial model
S0 = 100  # Initial asset price
K = 100  # Strike price
T = 1  # Time to expiration (1 year)
r = 0.05  # Risk-free interest rate
sigma = 0.2  # Volatility (20%)
n = 100  # Number of steps

# Price American call and put options using the binomial model
call_tree, ST = american_option(S0, K, T, r, sigma, n, option_type="
    call")
put_tree, _ = american_option(S0, K, T, r, sigma, n, option_type="put")

# Print the final prices
print(f"American Call Option Price: {call_tree[0, 0]:.2f}")
print(f"American Put Option Price: {put_tree[0, 0]:.2f}")

# Plotting the American option prices for all time steps
time_steps = np.linspace(0, T, n+1)

# Plot American Call Option prices (only the last few nodes for
    simplicity)
plt.figure(figsize=(10, 6))
plt.plot(time_steps, call_tree[0, :], 'bo-', label="American Call
    Option Price")
```

```
plt.xlabel("Time to Expiration (Years)")
plt.ylabel("American Call Option Price")
plt.title("American Call Option Price Evolution (Binomial Model)")
plt.grid(True)
plt.legend()

# Save the call option graph
#plt.savefig(r'C:\Users\LENOVO\Desktop\American_Call_Option_Price_Chart.png')
plt.show()

# Plot American Put Option prices (only the last few nodes for simplicity)
plt.figure(figsize=(10, 6))
plt.plot(time_steps, put_tree[0, :], 'ro-', label="American Put Option Price")
plt.xlabel("Time to Expiration (Years)")
plt.ylabel("American Put Option Price")
plt.title("American Put Option Price Evolution (Binomial Model)")
plt.grid(True)
plt.legend()

# Save the put option graph
#plt.savefig(r'C:\Users\LENOVO\Desktop\American_Put_Option_Price_Chart.png')
plt.show()
```

A.4 European Call Option Price using Black-Scholes Model

The Python code used for calculating and plotting the European call option price using the Black-Scholes model is as follows:

```
# Import necessary libraries
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

# Define the Black-Scholes function
def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price

# Parameters for the Black-Scholes model
S = np.linspace(50, 150, 100)  # Stock price range for plotting
K = 100  # Strike price
T = 1    # Time to expiration in years
r = 0.05 # Risk-free interest rate
sigma = 0.2 # Volatility

# Calculate call prices for all stock prices
call_prices = black_scholes_call(S, K, T, r, sigma)
```

```
22
23 # Specific calculation for S = 100
24 S_specific = 100
25 call_price_specific = black_scholes_call(S_specific, K, T, r, sigma)
26
27 # Display the calculated price
28 print(f"European Call Option Price for S={S_specific}: {
       call_price_specific}")
29
30 # Plot the graph
31 plt.figure(figsize=(10, 6))
32 plt.plot(S, call_prices, label="Call Option Prices (Black-Scholes)",
       color="blue", alpha=0.7)
33 plt.axvline(S_specific, color="red", linestyle="--", label=f"S = {
       S_specific}")
34 plt.text(S_specific + 2, call_price_specific, f"Price: {
       call_price_specific:.2f}", color="black", fontsize=10)
35 plt.xlabel("Stock Price (S)", fontsize=12)
36 plt.ylabel("Call Option Price", fontsize=12)
37 plt.title("European Call Option Prices vs. Stock Prices (Black-Scholes)
       ", fontsize=14)
38 plt.legend()
39 plt.grid()
40 plt.show()
```

A.5 American Call Option Price using Black-Scholes Model

The Python code used for calculating and plotting the American call option price using the Black-Scholes model is as follows:

```
1
2 # Import necessary libraries
3 import numpy as np
4 from scipy.stats import norm
5 import matplotlib.pyplot as plt
6
7 # Define the Black-Scholes function for a call option
8 def black_scholes_call(S, K, T, r, sigma):
9     d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(
          T))
10     d2 = d1 - sigma * np.sqrt(T)
11     call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
12     return call_price
13
14 # Parameters for the Black-Scholes model
15 S = np.linspace(50, 150, 100)  # Stock price range for plotting
16 K = 100  # Strike price
17 T = 1    # Time to expiration in years
18 r = 0.05 # Risk-free interest rate
19 sigma = 0.2 # Volatility
20
21 # Calculate call prices for all stock prices
22 call_prices = black_scholes_call(S, K, T, r, sigma)
23
```

```
# Specific calculation for S = 100
S_specific = 100
call_price_specific = black_scholes_call(S_specific, K, T, r, sigma)

# Display the calculated price
print(f"American Call Option Price for S={S_specific} (non-dividend-paying): {call_price_specific}")

# Plot the graph
plt.figure(figsize=(10, 6))
plt.plot(S, call_prices, label="American Call Option Prices (Black-Scholes)", color="green", alpha=0.7)
plt.axvline(S_specific, color="red", linestyle="--", label=f"S = {S_specific}")
plt.text(S_specific + 2, call_price_specific, f"Price: {call_price_specific:.2f}", color="black", fontsize=10)
plt.xlabel("Stock Price (S)", fontsize=12)
plt.ylabel("Call Option Price", fontsize=12)
plt.title("American Call Option Prices (Non-Dividend-Paying Stock) vs. Stock Prices", fontsize=14)
plt.legend()
plt.grid()
plt.show()
```

A.6 Finite Difference Method for European Call Option Pricing

The following Python code implements the Finite Difference Method for pricing a European Call Option: European Call Option Price using Finite Difference: 10.85 European Call Option Price using Black-Scholes: 10.45

```
The following Python code implements the finite difference method with an early exercise condition for American call options:

\begin{lstlisting}[language=Python, caption=Pricing European Call Option using Finite Difference Method]

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Finite Difference method for European Call Option with full solution history

def finite_difference_european_call_graph(S_max, K, T, r, sigma, M, N):
    # Create asset price grid and time discretization
    S = np.linspace(0, S_max, M + 1)  # Asset price grid (0 to S_max)
    dt = T / N                        # Time step size
    dS = S_max / M                    # Asset price step size

    # Initialize option price grid at maturity: V[:, N] holds the payoff at T
    V = np.maximum(S - K, 0)          # Payoff condition at maturity for a call option
```

```
    V_history = np.zeros((M + 1, N + 1))  # To store V at each time step
    V_history[:, N] = V                   # Set terminal condition

    # Finite difference coefficients (explicit scheme)
    alpha = 0.5 * dt * (sigma**2 * (S / dS)**2 - r * (S / dS))
    beta = 1 - dt * (sigma**2 * (S / dS)**2 + r)
    gamma = 0.5 * dt * (sigma**2 * (S / dS)**2 + r * (S / dS))

    # Apply boundary conditions for all time steps:
    # At S = 0, option price is 0; at S = S_max, option price is the intrinsic value.
    V[0] = 0
    V[M] = max(S_max - K, 0)

    # Backward time-stepping loop: from maturity back to t = 0
    for j in range(N - 1, -1, -1):
        for i in range(1, M):
            V[i] = alpha[i] * V[i - 1] + beta[i] * V[i] + gamma[i] * V[i + 1]
            # Safeguard: if an overflow or invalid value is encountered, set it to 0
            if np.isnan(V[i]) or np.isinf(V[i]):
                V[i] = 0
        # Reapply boundary conditions at each time step
        V[0] = 0
        V[M] = max(S_max - K, 0)
        V_history[:, j] = V  # Store the computed option values for this time step

    return S, V_history

# Black-Scholes formula for European Call Option (for comparison)
def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)

# Parameters for the option and finite difference model
S_max = 200       # Maximum asset price
K = 100           # Strike price
T = 1             # Time to expiration (1 year)
r = 0.05          # Risk-free interest rate
sigma = 0.2       # Volatility (20%)
M = 50            # Number of asset price steps (adjust for stability)
N = 200           # Number of time steps (adjust for stability)

# Compute the full finite difference solution (option price evolution)
S, V_history = finite_difference_european_call_graph(S_max, K, T, r, sigma, M, N)

# Plot the evolution of the option price at several time levels
# Choose 5 time levels from t=0 to t=T
time_indices = [0, int(N/4), int(N/2), int(3*N/4), N]
plt.figure(figsize=(10, 6))
for j in time_indices:
    t = float(j * T / N)  # Convert time to float for proper formatting
```

```
    plt.plot(S, V_history[:, j], label=f"t = {t:.2f} years")
plt.xlabel("Asset Price (S)")
plt.ylabel("Option Price (V)")
plt.title("European Call Option Price Evolution (Finite Difference)")
plt.legend()
plt.grid(True)
plt.show()

# For comparison, calculate the Black-Scholes price at S = K:
call_price_bs = black_scholes_call(K, K, T, r, sigma)
print(f"European Call Option Price using Black-Scholes: {call_price_bs
    :.2f}")

# Also, print the finite difference price at S = K:
closest_index = np.abs(S - K).argmin()  # Find index where S is closest
     to K
print(f"European Call Option Price using Finite Difference: {V_history[
    closest_index, 0]:.2f}")
```

A.7 Finite Difference Method for American Call Option Pricing

The following Python code implements the finite difference method with an early exercise condition for American call options: American Call Option Price using Finite Difference: 10.449780040917219

Listing A.1: Pricing American Call Option using Finite Difference Method

```

import numpy as np

def finite_difference_american_call(S_max, K, T, r, sigma, M, N):
    # Asset price grid
    S = np.linspace(0, S_max, M + 1)  # Asset price grid
    dt = T / N  # Time step
    dS = S_max / M  # Asset price step
    V = np.maximum(S - K, 0)  # Payoff condition at maturity (at time T
        )

    # Finite difference coefficients (exclude the boundaries)
    alpha = 0.5 * dt * (sigma**2 * (S[1:-1] / dS)**2 - r * (S[1:-1] /
        dS))
    beta = 1 - dt * (sigma**2 * (S[1:-1] / dS)**2 + r)
    gamma = 0.5 * dt * (sigma**2 * (S[1:-1] / dS)**2 + r * (S[1:-1] /
        dS))

    # Time-stepping loop with early exercise condition
    for j in range(N - 1, -1, -1):  # Backward time loop
        V[1:-1] = alpha * V[0:-2] + beta * V[1:-1] + gamma * V[2:]
        # Apply early exercise condition
        V[1:-1] = np.maximum(V[1:-1], S[1:-1] - K)

    # Boundary conditions at S=0 and S=S_max
    V[0] = 0  # Option price at S=0 (for call option)
    V[-1] = max(S_max - K, 0)  # Option price at S_max (corrected
        boundary condition)
```

```
25
26      return V[M // 2]  # Return the price at the middle asset point (
            close to the strike price)
27
28  # Example usage
29  S_max = 200  # Maximum asset price
30  K = 100  # Strike price
31  T = 1  # Time to expiration in years
32  r = 0.05  # Risk-free interest rate
33  sigma = 0.2  # Volatility
34  M = 500  # Asset price steps (fine grid)
35  N = 10000  # Time steps (increased for better precision)
36
37  call_price = finite_difference_american_call(S_max, K, T, r, sigma, M,
        N)
38  print(f"American Call Option Price using Finite Difference: {call_price
        }")
```

A.8 Monte Carlo Simulation for European Call Option Pricing

Below is the Python implementation of the Monte Carlo simulation for pricing a European call option:

Listing A.2: Monte Carlo simulation for pricing a European call option

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   def monte_carlo_european_call(S0, K, T, r, sigma, num_simulations):
5       dt = T  # Single time step (maturity)
6       prices = np.zeros(num_simulations)  # Array to store payoffs
7       stock_prices = np.zeros(num_simulations)  # Array to store stock
            prices at maturity
8
9       for i in range(num_simulations):
10          Z = np.random.normal(0, 1)  # Standard normal random variable
11          ST = S0 * np.exp((r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt
                ) * Z)  # Stock price at maturity
12          stock_prices[i] = ST
13          prices[i] = max(ST - K, 0)  # Payoff for call option
14
15      # Calculate discounted expected payoff
16      call_price = np.exp(-r * T) * np.mean(prices)
17
18      # Plot scatter graph
19      plt.figure(figsize=(10, 6))
20      plt.scatter(stock_prices, prices, alpha=0.6, color="blue", label="
            Simulated Payoffs")
21      plt.axhline(0, color="red", linestyle="--", label="Payoff = 0")
22      plt.xlabel("Stock Price at Maturity (S_T)", fontsize=12)
23      plt.ylabel("Payoff", fontsize=12)
24      plt.title("Scatter Plot of Stock Prices vs. Payoffs", fontsize=14)
25      plt.legend()
26      plt.grid(True)
27      plt.show()
```

```
28
29      return call_price
30
31  # Example usage
32  S0 = 100  # Initial stock price
33  K = 100   # Strike price
34  T = 1     # Time to expiration in years
35  r = 0.05  # Risk-free interest rate
36  sigma = 0.2  # Volatility
37  num_simulations = 10000  # Number of simulations
38
39  call_price = monte_carlo_european_call(S0, K, T, r, sigma,
        num_simulations)
40  print(f"European Call Option Price using Monte Carlo Simulation: {
        call_price:.2f}")
```

A.9 Pricing American Call Options Using Monte Carlo Simulation

The following Python code implements the Monte Carlo method for pricing American call options:

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   %matplotlib inline
4   def monte_carlo_american_call(S0, K, T, r, sigma, num_simulations,
        num_timesteps):
5       dt = T / num_timesteps  # Time step
6       prices = np.zeros((num_simulations, num_timesteps + 1))  #
            Simulated paths
7       payoffs = np.zeros(num_simulations)
8
9       # Simulate price paths
10      for i in range(num_simulations):
11          prices[i, 0] = S0
12          for t in range(1, num_timesteps + 1):
13              Z = np.random.normal(0, 1)  # Standard normal random
                    variable
14              prices[i, t] = prices[i, t - 1] * np.exp((r - 0.5 * sigma
                    **2) * dt + sigma * np.sqrt(dt) * Z)
15
16      # Calculate intrinsic values at maturity
17      payoffs = np.maximum(prices[:, -1] - K, 0)
18
19      # Backward induction
20      for t in range(num_timesteps - 1, 0, -1):
21          in_the_money = prices[:, t] > K
22          regression = np.polyfit(prices[in_the_money, t], payoffs[
                in_the_money] * np.exp(-r * dt), 2)
23          continuation_value = np.polyval(regression, prices[in_the_money
                , t])
24
25          # Exercise decision: intrinsic value vs continuation value
26          exercise_value = prices[in_the_money, t] - K
27          payoffs[in_the_money] = np.maximum(exercise_value,
```

```
            continuation_value)

    # Discount the remaining payoffs
    option_price = np.mean(payoffs * np.exp(-r * dt))

    # Plot stock price paths (only a few paths for clarity)
    plt.figure(figsize=(10, 6))
    for i in range(min(10, num_simulations)):  # Display only a few
        paths for clarity
        plt.plot(prices[i, :], lw=1)
    plt.xlabel("Time Step", fontsize=12)
    plt.ylabel("Stock Price", fontsize=12)
    plt.title("Simulated Stock Price Paths", fontsize=14)

    # Add legend for the first few paths
    plt.legend([f"Path {i+1}" for i in range(min(10, num_simulations))
        ], loc='upper left', fontsize=10)

    plt.grid(True)
    plt.show()  # Ensure plot displays in Jupyter Notebook

    return option_price

# Example usage
S0 = 100  # Initial stock price
K = 100   # Strike price
T = 1     # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma = 0.2  # Volatility
num_simulations = 10000  # Number of simulations
num_timesteps = 50  # Number of timesteps

american_call_price = monte_carlo_american_call(S0, K, T, r, sigma,
    num_simulations, num_timesteps)
print(f"American Call Option Price using Monte Carlo Simulation: {
    american_call_price:.2f}")
```

A.10 Analysis of Greeks for European Call Option using Black-Scholes Model

The following Python code computes the Greeks for a European call option using the Black-Scholes model and plots a scatter graph of the Greeks against stock prices.

Listing A.3: Python Implementation of Black-Scholes Greeks

```
# Ensure this is placed at the top of the notebook to enable inline
    plotting
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import os

# Define the option parameters
S = np.array([50.00, 70.20, 90.40, 110.61, 130.81])  # Stock prices
```

```
K = 100  # Strike price
T = 1  # Time to maturity (in years)
r = 0.05  # Risk-free interest rate
sigma = 0.2  # Volatility (20% annual volatility)
option_type = 'call'  # Option type ('call' or 'put')

# Define Black-Scholes formula for European call options
def black_scholes(S, K, T, r, sigma, option_type='call'):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(
        T))
    d2 = d1 - sigma * np.sqrt(T)

    if option_type == 'call':
        price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    elif option_type == 'put':
        price = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)

    # Greeks
    delta = norm.cdf(d1)
    gamma = norm.pdf(d1) / (S * sigma * np.sqrt(T))
    theta = -(S * norm.pdf(d1) * sigma) / (2 * np.sqrt(T)) - r * K * np
        .exp(-r * T) * norm.cdf(d2)
    vega = S * norm.pdf(d1) * np.sqrt(T)
    rho = K * T * np.exp(-r * T) * norm.cdf(d2)

    return price, delta, gamma, theta, vega, rho

# Store Greeks and prices
deltas, gammas, thetas, vegas, rhos = [], [], [], [], []

for stock_price in S:
    _, delta, gamma, theta, vega, rho = black_scholes(stock_price, K, T
        , r, sigma, option_type)
    deltas.append(delta)
    gammas.append(gamma)
    thetas.append(theta)
    vegas.append(vega)
    rhos.append(rho)

# Define the path to save the files
desktop_path = r'C:\Users\LENOVO\Desktop'

# Delta vs. Stock Price
plt.figure(figsize=(6, 4))
plt.plot(S, deltas, label='Delta', color='blue', marker='o')
plt.title('Delta vs. Stock Price')
plt.xlabel('Stock Price')
plt.ylabel('Delta')
plt.grid(True)
plt.legend(loc='best')
# Display the figure
plt.show()
# Save the figure to the desktop
plt.savefig(os.path.join(desktop_path, 'Delta1.png'))
plt.close()

# Gamma vs. Stock Price
plt.figure(figsize=(6, 4))
```

```
plt.plot(S, gammas, label='Gamma', color='green', marker='o')
plt.title('Gamma vs. Stock Price')
plt.xlabel('Stock Price')
plt.ylabel('Gamma')
plt.grid(True)
plt.legend(loc='best')
# Display the figure
plt.show()
# Save the figure to the desktop
plt.savefig(os.path.join(desktop_path, 'Gamma1.png'))
plt.close()

# Theta vs. Stock Price
plt.figure(figsize=(6, 4))
plt.plot(S, thetas, label='Theta', color='red', marker='o')
plt.title('Theta vs. Stock Price')
plt.xlabel('Stock Price')
plt.ylabel('Theta')
plt.grid(True)
plt.legend(loc='best')
# Display the figure
plt.show()
# Save the figure to the desktop
plt.savefig(os.path.join(desktop_path, 'Theta1.png'))
plt.close()

# Vega vs. Stock Price
plt.figure(figsize=(6, 4))
plt.plot(S, vegas, label='Vega', color='purple', marker='o')
plt.title('Vega vs. Stock Price')
plt.xlabel('Stock Price')
plt.ylabel('Vega')
plt.grid(True)
plt.legend(loc='best')
# Display the figure
plt.show()
# Save the figure to the desktop
plt.savefig(os.path.join(desktop_path, 'Vega1.png'))
plt.close()

# Rho vs. Stock Price
plt.figure(figsize=(6, 4))
plt.plot(S, rhos, label='Rho', color='orange', marker='o')
plt.title('Rho vs. Stock Price')
plt.xlabel('Stock Price')
plt.ylabel('Rho')
plt.grid(True)
plt.legend(loc='best')
# Display the figure
plt.show()
# Save the figure to the desktop
plt.savefig(os.path.join(desktop_path, 'Rho1.png'))
plt.close()
```

A.11 Finite Difference Method for American Call Option Pricing

Finite Difference Method for American Call Option Pricing for graph:

Listing A.4: Python Finite Difference Method for American Call Option Pricing of Results

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
S_max = 200  # Maximum asset price
K = 100  # Strike price
T = 1  # Time to expiration in years
r = 0.05  # Risk-free interest rate
sigma = 0.2  # Volatility
M = 200  # Asset price steps
N = 1000  # Time steps

# Asset price grid
S = np.linspace(0, S_max, M + 1)

# Time step and asset price step
dt = T / N
dS = S_max / M

# Initialize option value at maturity
V = np.maximum(S - K, 0)

# Finite difference coefficients
alpha = 0.5 * dt * (sigma ** 2 * (S / dS) ** 2 - r * (S / dS))
beta = 1 - dt * (sigma ** 2 * (S / dS) ** 2 + r)
gamma = 0.5 * dt * (sigma ** 2 * (S / dS) ** 2 + r * (S / dS))

# Time-stepping loop with early exercise condition
for j in range(N - 1, -1, -1):
    V[1:-1] = alpha[1:-1] * V[0:-2] + beta[1:-1] * V[1:-1] + gamma
        [1:-1] * V[2:]
    V[1:-1] = np.maximum(V[1:-1], S[1:-1] - K)  # Early exercise
        condition

# Plot the option value as a function of asset price at maturity
plt.plot(S, V)
plt.xlabel('Asset Price (S)')
plt.ylabel('Option Price')
plt.title('American Call Option Price vs Asset Price')
plt.grid(True)
plt.savefig(r"C:\Users\LENOVO\Desktop\pde_american_finite_difference.
    png")
plt.show()
# Assuming your plot is stored in a variable named `p`
#plt.savefig(r"C:\Users\LENOVO\Desktop\pde_american_finite_difference.
    png")
```

A.12 European call option and hedging portfolio

The following Python code is used to European Call option and hedging:

Listing A.5: Python Implementation of European Call option and hedging of Results

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

# Define the Black-Scholes function for European call option price
def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(
       T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price

# Define the Black-Scholes function for Delta (Hedging Portfolio)
def black_scholes_delta(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(
       T))
    delta = norm.cdf(d1)  # Delta is the derivative of the option price
        with respect to S
    return delta

# Parameters for the Black-Scholes model
S = np.linspace(50, 150, 100)  # Stock price range for plotting
K = 100  # Strike price
T = 1    # Time to expiration in years
r = 0.05 # Risk-free interest rate
sigma = 0.2 # Volatility

# Calculate call prices for all stock prices
call_prices = black_scholes_call(S, K, T, r, sigma)

# Calculate Delta (Hedging portfolio ratio) for all stock prices
delta_values = black_scholes_delta(S, K, T, r, sigma)

# Calculate Hedging Portfolio Value (Delta * Stock Price)
hedging_portfolio_values = delta_values * S

# Specific calculation for S = 100
S_specific = 100
call_price_specific = black_scholes_call(S_specific, K, T, r, sigma)
delta_specific = black_scholes_delta(S_specific, K, T, r, sigma)
hedging_portfolio_specific = delta_specific * S_specific

# Display the calculated prices and values
print(f"European Call Option Price for S={S_specific}: {
   call_price_specific}")
print(f"Delta for Hedging Portfolio at S={S_specific}: {delta_specific}
   ")
print(f"Hedging Portfolio Value for S={S_specific}: {
   hedging_portfolio_specific}")

# Plot the graph
plt.figure(figsize=(10, 6))

```

```
# Plot the European Call Option Prices
plt.plot(S, call_prices, label="Call Option Prices (Black-Scholes)",
    color="blue", alpha=0.7)

# Plot the Hedging Portfolio Values
plt.plot(S, hedging_portfolio_values, label="Hedging Portfolio Value",
    color="green", alpha=0.7)

# Highlight S = 100 with a vertical line
plt.axvline(S_specific, color="red", linestyle="--", label=f"S = {
    S_specific}")
plt.text(S_specific + 2, call_price_specific, f"Price: {
    call_price_specific:.2f}", color="black", fontsize=10)
plt.text(S_specific + 2, hedging_portfolio_specific, f"Portfolio: {
    hedging_portfolio_specific:.2f}", color="black", fontsize=10)

# Labels and title
plt.xlabel("Stock Price (S)", fontsize=12)
plt.ylabel("Price / Portfolio Value", fontsize=12)
plt.title("European Call Option Prices and Hedging Portfolio Values (
    Black-Scholes)", fontsize=14)

# Display the legend and grid
plt.legend()
plt.grid(True)
plt.savefig(r"C:\Users\LENOVO\Desktop\European_call_option_and_hedging.
    png", format='png')
# Show the plot
plt.show()
```