Comparison of Different Techniques in European Options Pricing

Kgosietsile Thuso Ntsono Nathanial Petersen
Yamkelani Socamangashe Shumani Raludzingana

September 22, 2024

Abstract

This project explores the pricing techniques for European options, focusing on the Binomial Tree (BN) model and the Black-Scholes (BS) model. Both models are widely used in financial markets for option pricing; however, it has been found that the simulation of the Binomial Tree model is unstable under high volatility and stable under low volatility, whereas the simulation of the Black-Scholes model is unaffected by volatility. The study that Group 5 will conduct compares these techniques in terms of accuracy, efficiency, and applicability under varying market conditions, allowing us to see their strengths and limitations.

1 Introduction

The investment market is currently expanding and developing at an ever-increasing rate, the growing number of alternative investment tools available to investors serves as evidence of this. In mathematical finance, the pricing of financial derivatives plays an important role in global markets, especially options. The primary dynamic segment of the security market is represented by options, a type of derivative that grants the holder the right, but not the obligation, to purchase an underlying asset [4]. Based on the type of time period implementation, options are separated into two categories: American and European options. The European type option is executed at maturity, whereas the American type option is executed at or before maturity [4]. Options are mainly utilized for speculation and hedging, and the two types of option rights are the purchase options (also known as the call option) and the selling options (also known as a put option). Corporations use options to guard agains losses from fluctuations in foreign exchange rates, and by consumers to manage risks [2]. Since the price of an option is a crucial component of investment strategy, economists, option providers, and consumers have always had to address the issue of how to price options correctly. In this paper, two option pricing methods will be compared.

In the financial market, the Binomial Tree Model and the Black Scholes Model are the two well-known models. The Black Scholes Model is an option that was devolped for European style options without dividends which allowed investors to approximate a price [2]. This model made the assumptions that there will not be any dividend payments made during the option period, that there will not be any transaction fees, that the interest rates will be risk-free and stable over time, and that the stock price will move in a

Geometric Brown motion. The Binomial Tree Model's approach shows that there are two possible ways that stock prices can move in the market, that is up or down [2]. Although the goal of both models is to produce precise option price estimates, their underlying presumptions and computational strategies are completely different, as a result both methods have their strengths and limitations. This study compares the two models with emphasis on their real world applications, theoretical underpinnings, and their accuracy when it comes to pricing in various market scenarios. This research contributes to the clarification of which model is more appropriate for particular market environments by examining the practical implications of each model.

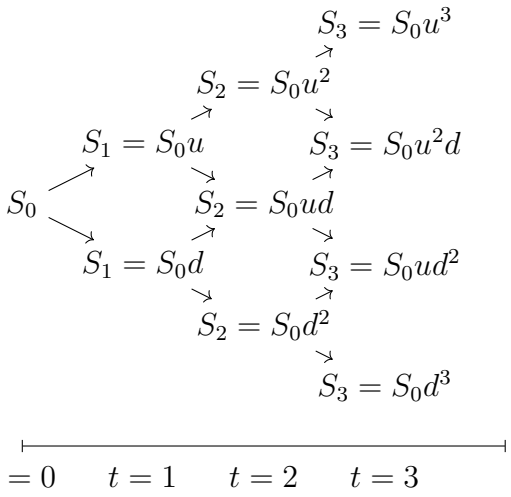
The benefits, drawbacks, and the uses of the Black Scholes and the Binomial Tree Models in the context of option pricing have been extensively studied in previous and relevant studies. According to a He survey, the Binomial Tree and the Black Scholes model have the same tendency when pricing European options. However, the volatility of the model affects the binomial tree's accuracy [4]. Zeyuan Shao's study on the pricing technique for european option and application state that by computing the expectation, the Binomial Tree Model provides a comprehensive list of all potential future carbon credits and efficiently determines the necessary date price. The outcome Shao shows that when it comes to pricing European, the Binomial Tree model can function just as well as the Black Scholes model. Despite the Binomial Tree model's steady efficiency at low volatility, the step was close to 2000. More actions, such as 1.5 sigmas, will be required to keep the accuracy level the same under higher volatility.

Even though there has been much research on the comparison between the Black Scholes and the Binomial Tree Models, this paper is unique in that it focuses on using these models in the South African context, addressing the particular difficulties and market conditions of the area. The rest of the project is organised as follows:

The rest of the project is organised as follows: section 2 covers the Binomial Tree Model, section 3 covers the derivation of the Black Scholes Model, section 4 covers the result comparisons between the two models, section 5 is the conclusion, and the last section is the reference list.

2 Binomial Tree Model

The binomial option pricing model is a deceptively simple approach to addressing a complex problem. In this model, stock prices can only move up or down at each step, branching out like a tree. This basic assumption allows the model to be remarkably accurate and flexible in pricing options [1].



The nodal value $S(t_k)$ represents the price of the security at time t_k. At each time step, the price can either go up or down, leading to the following relationship:

$$S(t_{k+1}) = \begin{cases} u \cdot S(t_k) & \text{with probability } p \\ d \cdot S(t_k) & \text{with probability } 1-p \end{cases} \tag{1}$$

Where:

$$u = e^{\sigma\sqrt{\frac{t}{n}}}, \quad d = e^{-\sigma\sqrt{\frac{t}{n}}}, \quad p = \frac{e^{r\frac{t}{n}} - d}{u - d} \tag{2}$$

To understand the model, note that if i of the first k price movements are increases and $k - i$ are decreases, then the price at time t_k will be:

$$S(t_k) = u^i d^{k-i} S_0 \tag{3}$$

This implies that there are $k + 1$ possible prices of the security at time t_k, since i can take on values from 0 to k.

Now, let $V_k(i)$ denote the expected return from the put option at time t_k. To calculate $V_0(0)$, the present value of owning the put, we work backwards. First, we determine $V_n(i)$ for each of its $n+1$ possible values of i; then we compute $V_{n-1}(i)$ for each of its n possible values, continuing this process step by step [3].

At expiration time t_n, the value of the option is:

$$V_n(i) = \max(K - u^i d^{n-i} S_0, 0) \tag{4}$$

where K is the strike price.

Remark: Equation (4) is used when we want to calculate the European option using the Binomial tree method.

Let $\beta = e^{-r\frac{t}{n}}$, where r is the interest rate, t is the option's expiration time, and n is the number of periods. Suppose we are at time t_k, and the put option has not yet been exercised. The stock price is $u^i d^{k-i} S_0$, where i is the number of upward movements, and $k - i$ is the number of downward movements.

If we exercise the option at time t_k, we will receive $K - u^i d^{k-i} S_0$. If we do not exercise the option, the stock price at t_{k+1} will either increase to $u^{i+1} d^{k-i} S_0$ with probability p, or decrease to $u^i d^{k-i+1} S_0$ with probability $1 - p$.

The expected return at time t_k if we do not exercise the option is given by:

$$p\beta V_{k+1}(i+1) + (1-p)\beta V_{k+1}(i) \tag{5}$$

The maximal expected return at time t_k is the larger of this value and the immediate payoff from exercising the option:

$$V_k(i) = \max\left(K - u^i d^{k-i} S_0,\ \beta\left(pV_{k+1}(i+1) + (1-p)V_{k+1}(i)\right)\right) \tag{6}$$

for $k = 0, 1, \ldots, n-1$.

To compute the price of the option, we work backwards through the binomial tree. First, we calculate the value of the option at the final time step using the payoff formula $V_n(i) = \max(K - u^i d^{n-i} S_0, 0)$ (see equation (4)). Then, we apply the recurrence relation for each preceding time step until we determine $V_0(0)$, which gives us the present value of the option. Though this process may seem tedious, it can be easily implemented programmatically or in a spreadsheet.

Remark: When we have to implement the American Option, we use equation (6) which will give us all the possible values for every possible price at any point that we would like to exercise at any time prior to expiration.

2.1 Python Code Implementation for European Options

The following Python code implements the binomial tree model for calculating European call and put option prices:

Listing 1: European Option Pricing using Binomial Model

```
import math

# Parameters
s = 15600            # Initial stock price
K = 15700          # Strike price
t = 0.25         # Time to maturity (in years)
sigma = 0.1846    # Volatility
r = 0.065         # Risk-free interest rate
n = 100           # Number of steps

# Calculations
dt = t / n
u = math.exp(sigma * math.sqrt(dt))           # Up factor
d = math.exp(-sigma * math.sqrt(dt))          # Down
   factor
p = (math.exp(r * dt) - d) / (u - d)          # Risk-
   neutral probability
beta = math.exp(-r * dt)                      # Discount
   factor

# Step 1: Compute stock prices at maturity
stock_prices = [s * (u ** (n - i)) * (d ** i) for i in
   range(n + 1)]

# Step 2: Compute call option values at maturity
call_option_values = [max(price - K, 0) for price in
   stock_prices]
```

```
# Step 3: Compute put option values at maturity
put_option_values = [max(K - price, 0) for price in
    stock_prices]

# Step 4: Backward induction to calculate the call option
    value at each node
for i in range(n - 1, -1, -1):
    for j in range(i + 1):
        call_option_values[j] = beta * (p *
            call_option_values[j] + (1 - p) *
            call_option_values[j + 1])

# Step 5: Backward induction to calculate the put option
    value at each node
for i in range(n - 1, -1, -1):
    for j in range(i + 1):
        put_option_values[j] = beta * (p *
            put_option_values[j] + (1 - p) *
            put_option_values[j + 1])

# Call and put option prices at time 0
call_option_price = call_option_values[0]
put_option_price = put_option_values[0]

# Print the results
print(f"European Call Option Price: {call_option_price:}"
    )
print(f"European Put Option Price: {put_option_price:}")
```

This Python code calculates the price of European call and put options using the binomial model. It defines the key parameters, builds the binomial tree, and then uses backward induction to compute the option prices at time 0.

3 Options Pricing Using the Black-Scholes Model

The Black-Scholes model is a mathematical model used for options pricing. In this project, we will make use of the Black-Scholes model, as well as computational methods, to simulate the pricing of options using relevant data from various articles that served as the basis for the project. The Black-Scholes model has the following assumptions associated with it [3]:

1. The price/value of an asset follows a Geometric Brownian Motion (GBM), i.e., the current price/value of an asset is independent of the asset's past price/value.
2. Arbitrage opportunity does not exist in the market.
3. There are no taxes, i.e., you don't pay tax if you choose to sell an asset (i.e., option).
4. The risk-free interest rate, r, and the volatility, σ, are constant.

Firstly, in order to formulate the Black-Scholes model, we have to define the cumulative density function for a standard normal distribution, $\Phi(x)$. We do this by recalling Standard Normal Random Variables and Continuous Probability. We have the cumulative density function for a standard normal distribution, $\Phi(x)$, defined as follows [3]:

$$P\{Z \leq x\} = \Phi(x) \tag{7}$$

where

$$Z = \frac{X - \mu}{\sigma} \tag{8}$$

We have that

$$P\{Z < -x\} = P\{Z > x\} \tag{9}$$

$$\Rightarrow P\{Z < -x\} = 1 - P\{Z \leq x\}$$

$$\Rightarrow \Phi(-x) = 1 - \Phi(x) \tag{10}$$

We can approximate the function, $\Phi(x)$, as follows:

$$\Phi(x) \approx 1 - \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \left(a_1 y + a_2 y^2 + a_3 y^3 + a_4 y^4 + a_5 y^5\right) \tag{11}$$

where

$$y = \frac{1}{1 + 0.2316419x} \tag{12}$$

$$a_1 = 0.319381530, \quad a_2 = -0.356563782, \quad a_3 = 1.781477937, \tag{13}$$

$$a_4 = -1.821255978, \quad a_5 = 1.330274429 \tag{14}$$

Now that we've defined the function, $\Phi(x)$, we can define the equation for a European Call option, as follows:

$$C = S(0)\Phi(\omega) - Ke^{-rt}\Phi(\omega - \sigma\sqrt{t}) \tag{15}$$

where $S(0)$ is the initial stock price, K is the strike price, r is the interest rate, t is the time period, and σ is the volatility. We have that

$$\omega = \frac{rt + \frac{\sigma^2 t}{2} - \ln\left(\frac{K}{S(0)}\right)}{\sigma\sqrt{t}} \tag{16}$$

As a side note, we know that the Black-Scholes model is unaffected by high volatility, whereas the Binomial Tree model is affected by high volatility; particularly, after about 2000 steps, the accuracy of the model declines.

Now, we define a European Put option, as follows:

$$P = C + Ke^{-rt} - S(0) \tag{17}$$

3.1 Python Code Implementation

Below is the Python code used to compute the Black-Scholes model values:

```
# The following code imports the numpy library as np.
import numpy as np

# The following code creates a variable called, t, and assigns a value to it.
t = 0.25
# The following code creates a variable called, r, and assigns a value to it.
r = 0.065

# The following code creates a variable called, sigma, and assigns a value to
    it.
sigma = 0.1846

# The following code creates a variable called, K, and assigns a value to it.
K = 500

# The following code creates a variable called, S_0, and assigns a value to it.
S_0 = 15675

# The following code creates a function called, phi(), which takes a single
    input argument, x.
def phi(x):
    # The following code creates a variable called, a_1, and assigns a value to
        it.
    a_1 = 0.319381530
    # The following code creates a variable called, a_2, and assigns a value to
        it.
    a_2 = -0.356563782
    # The following code creates a variable called, a_3, and assigns a value to
        it.
    a_3 = 1.781477937
    # The following code creates a variable called, a_4, and assigns a value to
        it.
    a_4 = -1.821255978
    # The following code creates a variable called, a_5, and assigns a value to
        it.
    a_5 = 1.330274429

    # The following code executes whatever is in the "code block" if x is non-
        negative, i.e. x >= 0.
    if x >= 0:
        # The following code creates a variable called, y, and assigns the
            value 1 divided by the sum of 1 and the product of 0.2316419 and the
             input argument, x.
        y = 1 / (1 + (0.2316419 * x))
        # The following code creates a variable called, answer, and assigns to
            it, 1 - ((1 / np.sqrt(2 * np.pi)) * np.exp((-1 * (x ** 2)) / 2) * ((
            a_1 * y) + (a_2 * (y ** 2)) + (a_3 * (y ** 3)) + (a_4 * (y ** 4)) +
            (a_5 * (y ** 5)))).
        answer = 1 - ((1 / np.sqrt(2 * np.pi)) * np.exp((-1 * (x ** 2)) / 2) *
            ((a_1 * y) + (a_2 * (y ** 2)) + (a_3 * (y ** 3)) + (a_4 * (y ** 4))
            + (a_5 * (y ** 5))))
        # The following code returns the value of the variable answer.
        return answer
    # The following code executes whatever is in the "code block" if x is
        negative, i.e. x < 0.
    else:
        # The following code creates a variable called, y, and assigns the
            value 1 divided by the subtraction of 1 and the product of 0.2316419
             and the input argument, x.
        y = 1 / (1 - (0.2316419 * x))
```

```
        # The following code creates a variable called, answer, and assigns to
            it, 1 - ((1 / np.sqrt(2 * np.pi)) * np.exp((-1 * (x ** 2)) / 2) * ((
            a_1 * y) + (a_2 * (y ** 2)) + (a_3 * (y ** 3)) + (a_4 * (y ** 4)) +
            (a_5 * (y ** 5)))).
        answer = 1 - ((1 / np.sqrt(2 * np.pi)) * np.exp((-1 * (x ** 2)) / 2) *
            ((a_1 * y) + (a_2 * (y ** 2)) + (a_3 * (y ** 3)) + (a_4 * (y ** 4))
            + (a_5 * (y ** 5))))
        # The following code returns the value of 1 minus the variable answer.
        return (1 - answer)

# The following code creates a variable called, omega, and assigns to it, ((r *
    t) + (((sigma ** 2) * t) / 2) - np.log(K / S_0)) / (sigma * np.sqrt(t)).
omega = ((r * t) + (((sigma ** 2) * t) / 2) - np.log(K / S_0)) / (sigma * np.
    sqrt(t))

# The following code creates a variable called, C, and assigns to it, (S_0 *
    phi(omega)) - (K * np.exp(-1 * r * t) * phi(omega - (sigma * np.sqrt(t)))).
C = (S_0 * phi(omega)) - (K * np.exp(-1 * r * t) * phi(omega - (sigma * np.sqrt
    (t))))

# The following code creates a variable called, P, and assigns to it, C + (K *
    np.exp(-1 * r * t)) - S_0.
P = C + (K * np.exp(-1 * r * t)) - S_0

# The following code outputs/displays the result of omega.
print("omega␣=␣" + str(omega))
# The following code outputs/displays the result of C.
print("C␣=␣" + str(C))
# The following code outputs/displays the result of P.
print("P␣=␣" + str(P))
```

4 Result Comparisons Between the Two Models

Using data from Riaman (2019) [2], we applied both the Binomial Model (BM) and the Black-Scholes Model (BSM) to calculate the values of European call and put options. This section presents a comparison of the results from both models.

4.1 Parameters

The following table 1 summarizes the key parameters used in the valuation of options. These include the stock price (S), strike price (K), risk-free interest rate (r), volatility (σ), and time to maturity (t). These parameters were essential inputs for both the Binomial and Black-Scholes models.

Data	Stock Price (S)	Strike (K)	Risk-Free (r)	Volatility (σ)	Time (t)
[1]	15600	15700	0.065	0.1846	0.25
[2]	15600	15600	0.060	0.1846	0.25
[3]	15600	15650	0.065	0.01846	0.25

Table 1: Input Parameters Used for Option Valuation

4.2 Value of Call and Put Options in Binomial Model (BM)

The table 2 below presents the calculated values of European call and put options using the Binomial Model (BM). These values were derived using the lattice approach, taking into account the previously defined parameters such as stock price, strike price, risk-free rate, and volatility.

Date	Call Option Value (BM)	Put Option Value (BM)
[1]	652.2933235670748	499.2300315008944
[2]	692.1404197584604	459.88667756615143
[3]	207.183713065437	4.926355050422644

Table 2: Values of Call and Put Options Using the Binomial Model (BM)

4.3 Value of Call and Put Options in Black-Scholes Model (BSM)

The table 3 below shows the values of European call and put options, calculated using the Black-Scholes Model (BSM). These values were computed based on the same input parameters, including stock price, strike price, risk-free rate, and volatility, as in the Binomial Model.

Data	Omega (ω)	Call Option Value (BSM)	Put Option Value (BSM)
[1]	0.15297775624300122	651.1939760142459	498.1306839482386
[2]	0.20866354279523294	693.5806803616651	461.32693816944266
[3]	1.4184825264711973	207.34461764798834	5.087259633146459

Table 3: Values of Call and Put Options Using the Black-Scholes Model (BSM)

5 discussion

This section presents a detailed comparison of the results obtained from the Binomial Model (BM) and the Black-Scholes Model (BSM) for the valuation of European call and put options, based on data provided by Riaman (2019) [2]. Both models use identical key parameters, including stock price (S), strike price (K), risk-free interest rate (r), volatility (σ), and time to maturity (t), which are critical inputs in the valuation process, as outlined in Table reftab:1.

5.1 Binomial Model Results

The values of European call and put options were computed using the Binomial Model (BM) via the lattice approach. The results, shown in Table 2, reflect the defined input parameters. The lattice method divides the time to maturity into discrete intervals, generating a price tree and providing the option values at each step.

5.2 Black-Scholes Model Results

Using the same set of input parameters, the Black-Scholes Model (BSM) was applied to calculate option values, using a closed-form solution for pricing European options. The results, presented in Table 3, include the calculated option values along with omega (ω), which measures the sensitivity of option prices to changes in volatility. The Black-Scholes formula assumes continuous price movements, making it particularly suitable for efficient market conditions.

5.3 Key Observations

- **Call Option Values**: Both the BM and BSM models produced comparable values for call options across all three data sets. For example, in Data [1], the call option values were 652.293 (BM) and 651.194 (BSM), indicating minimal variance between the two models.

- **Put Option Values**: Similarly, the put option values calculated under both models are closely aligned. However, slight discrepancies are observed, particularly in Data [2] and Data [3], where the BSM yielded slightly higher put option values compared to the BM.

- **Sensitivity to Volatility (Omega)**: The omega (ω) values derived from the Black-Scholes Model highlight the sensitivity of option prices to changes in volatility. In Data [3], the omega value is significantly higher (1.418), reflecting greater sensitivity, which aligns with the lower volatility in this data set.

6 conclusion

Both the Binomial Model and Black-Scholes Model provide closely aligned results for the valuation of European call and put options, with minor differences observed due to their differing computational approaches. While the Binomial Model uses a stepwise lattice approach, the Black-Scholes Model applies a continuous-time framework, leading to slight variances in option values, particularly for puts. However, both models remain reliable for option valuation under varying market conditions.

References

[1] Adam Hayes. Understanding the binomial option pricing model, 2024.

[2] Riaman, K. Parmikanti, I. Irianingsih, K. Joebaedi, and S. Supian. Convergence of binomial tree methods to black-scholes model on determining stock option prices. *IOP Conference Series: Materials Science and Engineering*, 567(1):012013, 2019.

[3] Sheldon M. Ross. *Mathematical Finance*. Cambridge University Press, third edition, 2011.

[4] Zeyuan Shao. Pricing technique for european option and application. *Highlights in Business, Economics and Management GBMS 2023*, 14:14–18, 2023. Available online at: `https://doi.org/10.54097/hbem.v14i.8930`.