# Mid Term Report
Option Pricing Models
FINSEARCH Competition

Pushp Raj Choudhary 23B2152
Avishkar Bahirwar 23B0765
Ronak Gupta 23B2148
Aditya Anand Gupta 23B2144

July 2025

## Contents

## 1 Introduction

Options are fundamental derivatives in financial markets, providing the right, but not the obligation, to buy or sell an asset at a specified price before or at a certain date. This mid-term report outlines our progress in the FINSEARCH competition, focusing on the theoretical foundation, model implementation, and preliminary analysis of option pricing models.

Options enable portfolio diversification and risk mitigation. They also serve as tools for speculation and strategic investment under varying market scenarios. Through a structured understanding of mathematical and statistical principles, our project aims to compare traditional and computational approaches to pricing options and draw insights from historical data.

## 2 Overview of Options

### 2.1 Types of Options

- **Call Option**: Right to buy the underlying asset.
- **Put Option**: Right to sell the underlying asset.

### 2.2 Option Styles

- **American Option**: Exercisable at any time before expiry.
- **European Option**: Exercisable only at expiry.
- **Exotic Options**: Include barrier, Asian, and lookback options with non-standard features.

### 2.3 Key Terminology

- **Strike Price (K)**: The agreed-upon price for the transaction.
- **Spot Price ($S_0$)**: Current market price of the underlying asset.
- **Maturity (T)**: Time until the option expires.
- **Volatility ($\sigma$)**: Measure of asset price fluctuation.
- **Risk-Free Rate (r)**: Theoretical rate of return of a riskless investment.

## 3 Option Payoff and Profit

- **Call Option Payoff:** $\max(0, S_T - K)$
- **Put Option Payoff:** $\max(0, K - S_T)$

- **Profit:** Payoff $-$ Premium paid

The payoff structure allows investors to limit losses and potentially gain substantial returns depending on market movement.

## 4 Mathematical Foundations

### 4.1 No-Arbitrage Principle

Option prices must adhere to no-arbitrage bounds to prevent riskless profit opportunities. For European options on non-dividend stocks:

$$\max(0, S_0 - Ke^{-rT}) \leq C_0 \leq S_0$$
$$\max(0, Ke^{-rT} - S_0) \leq P_0 \leq Ke^{-rT}$$

### 4.2 Put-Call Parity

For European options:

$$C_0 - P_0 = S_0 - Ke^{-rT}$$

This ensures pricing consistency and allows derivation of one price from the other.

## 5 Option Pricing Models

### 5.1 Binomial Option Pricing Model

A discrete-time approach using a recombining price tree. It models multiple price paths and is useful for American options.

**Steps:**

1. Time to expiration $T$ divided into $n$ intervals of $\Delta t = T/n$.
2. Calculate up/down factors:
$$u = e^{\sigma\sqrt{\Delta t}}, \quad d = e^{-\sigma\sqrt{\Delta t}}$$
3. Risk-neutral probability:
$$p = \frac{e^{r\Delta t} - d}{u - d}$$
4. Build tree, compute payoff at maturity, apply backward induction:
$$C_0 = e^{-r\Delta t}(pC_u + (1-p)C_d)$$

### 5.2 Black-Scholes Model

A continuous-time model for European options on non-dividend paying stocks.

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

$$d_1 = \frac{\ln(S_0/K) + (r + 0.5\sigma^2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}$$

Where $N(\cdot)$ is the cumulative normal distribution.

### 5.3 Monte Carlo Simulation

Simulate multiple paths for $S_T$ using geometric Brownian motion:

$$S_T = S_0 \exp\left((r - 0.5\sigma^2)T + \sigma\sqrt{T}Z\right), \quad Z \sim N(0,1)$$

Estimate option price as average of discounted payoffs:

$$C \approx e^{-rT} \cdot \frac{1}{N} \sum_{i=1}^{N} \max(S_T^{(i)} - K, 0)$$

# 6 Implementation Details

## 6.1 Parameter Selection for Case Study

- $S_0 = 100$, $K = 95$, $T = 0.25$ years
- $r = 8\%$, $\sigma = 20\%$ annually

## 6.2 Sample Calculations

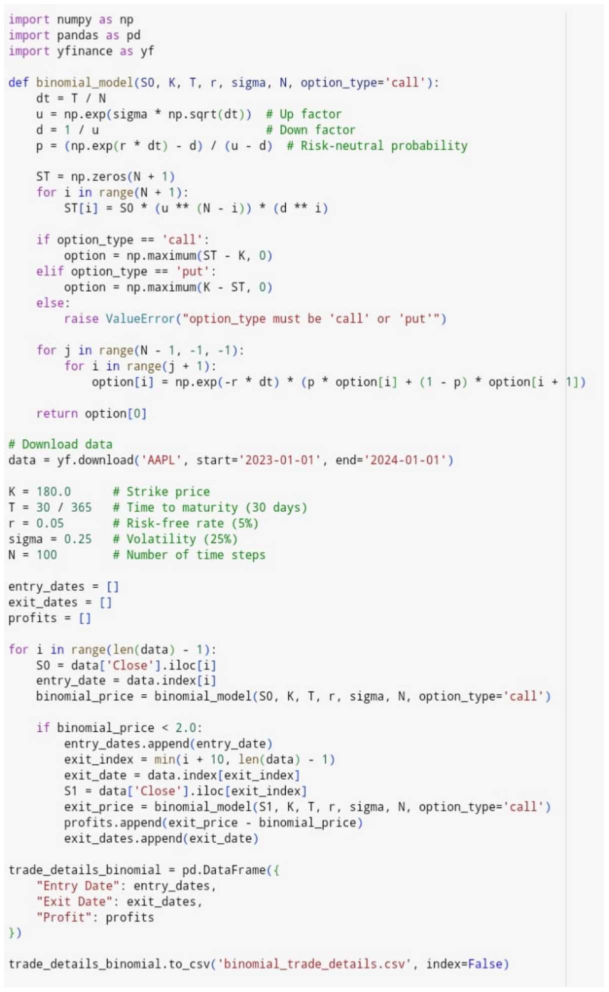
### 6.2.1 Black-Scholes Model

$$\begin{aligned}
d_1 &= \frac{\ln(100/95) + (0.08 + 0.5 \cdot 0.2^2) \cdot 0.25}{0.2 \cdot \sqrt{0.25}} \approx 0.76 \\
d_2 &= d_1 - 0.2\sqrt{0.25} \approx 0.66 \\
C &= 100 \cdot N(0.76) - 95 \cdot e^{-0.08 \cdot 0.25} \cdot N(0.66) \approx 8.23
\end{aligned}$$

### 6.2.2 Binomial Model (1-step Example)

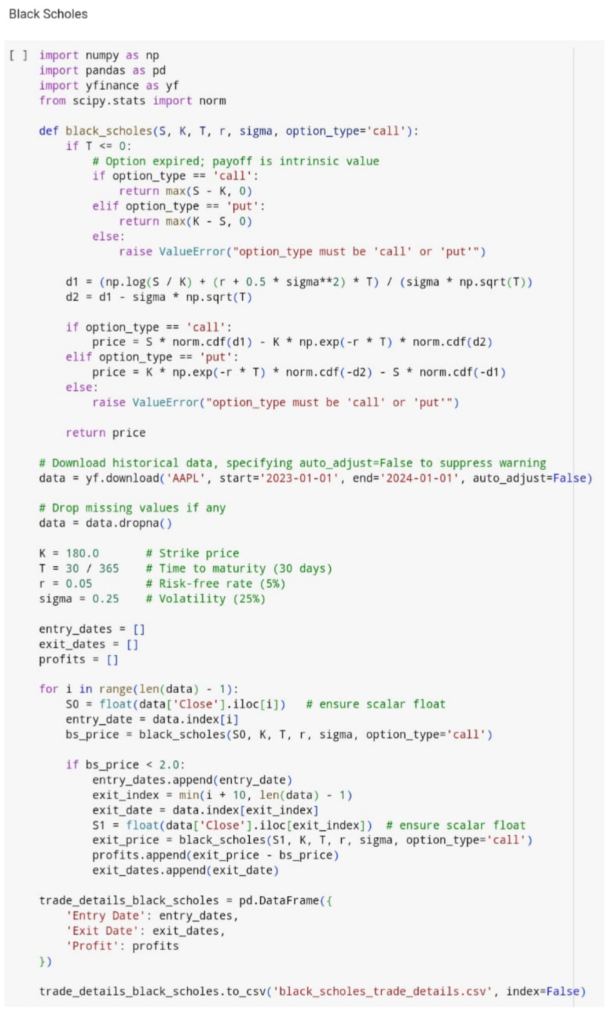
$$\begin{aligned}
u &= e^{0.2 \cdot \sqrt{0.25}} = 1.1, \quad d = 1/u = 0.9 \\
p &= \frac{e^{0.08 \cdot 0.25} - 0.9}{1.1 - 0.9} = 0.6 \\
&\text{Option Value} \approx 8.82
\end{aligned}$$

## 6.3 Python Implementation Snippet



```python
import numpy as np
import pandas as pd
import yfinance as yf

def binomial_model(S0, K, T, r, sigma, N, option_type='call'):
    dt = T / N
    u = np.exp(sigma * np.sqrt(dt))  # Up factor
    d = 1 / u                        # Down factor
    p = (np.exp(r * dt) - d) / (u - d)  # Risk-neutral probability

    ST = np.zeros(N + 1)
    for i in range(N + 1):
        ST[i] = S0 * (u ** (N - i)) * (d ** i)

    if option_type == 'call':
        option = np.maximum(ST - K, 0)
    elif option_type == 'put':
        option = np.maximum(K - ST, 0)
    else:
        raise ValueError("option_type must be 'call' or 'put'")

    for j in range(N - 1, -1, -1):
        for i in range(j + 1):
            option[i] = np.exp(-r * dt) * (p * option[i] + (1 - p) * option[i + 1])

    return option[0]

# Download data
data = yf.download('AAPL', start='2023-01-01', end='2024-01-01')

K = 180.0      # Strike price
T = 30 / 365   # Time to maturity (30 days)
r = 0.05       # Risk-free rate (5%)
sigma = 0.25   # Volatility (25%)
N = 100        # Number of time steps

entry_dates = []
exit_dates = []
profits = []

for i in range(len(data) - 1):
    S0 = data['Close'].iloc[i]
    entry_date = data.index[i]
    binomial_price = binomial_model(S0, K, T, r, sigma, N, option_type='call')

    if binomial_price < 2.0:
        entry_dates.append(entry_date)
        exit_index = min(i + 10, len(data) - 1)
        exit_date = data.index[exit_index]
        S1 = data['Close'].iloc[exit_index]
        exit_price = binomial_model(S1, K, T, r, sigma, N, option_type='call')
        profits.append(exit_price - binomial_price)
        exit_dates.append(exit_date)

trade_details_binomial = pd.DataFrame({
    "Entry Date": entry_dates,
    "Exit Date": exit_dates,
    "Profit": profits
})

trade_details_binomial.to_csv('binomial_trade_details.csv', index=False)
```

(a) Code Snippet 1



Black Scholes

```python
[ ] import numpy as np
    import pandas as pd
    import yfinance as yf
    from scipy.stats import norm

    def black_scholes(S, K, T, r, sigma, option_type='call'):
        if T <= 0:
            # Option expired; payoff is intrinsic value
            if option_type == 'call':
                return max(S - K, 0)
            elif option_type == 'put':
                return max(K - S, 0)
            else:
                raise ValueError("option_type must be 'call' or 'put'")

        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)
        else:
            raise ValueError("option_type must be 'call' or 'put'")

        return price

    # Download historical data, specifying auto_adjust=False to suppress warning
    data = yf.download('AAPL', start='2023-01-01', end='2024-01-01', auto_adjust=False)

    # Drop missing values if any
    data = data.dropna()

    K = 180.0       # Strike price
    T = 30 / 365    # Time to maturity (30 days)
    r = 0.05        # Risk-free rate (5%)
    sigma = 0.25    # Volatility (25%)

    entry_dates = []
    exit_dates = []
    profits = []

    for i in range(len(data) - 1):
        S0 = float(data['Close'].iloc[i])   # ensure scalar float
        entry_date = data.index[i]
        bs_price = black_scholes(S0, K, T, r, sigma, option_type='call')

        if bs_price < 2.0:
            entry_dates.append(entry_date)
            exit_index = min(i + 10, len(data) - 1)
            exit_date = data.index[exit_index]
            S1 = float(data['Close'].iloc[exit_index])  # ensure scalar float
            exit_price = black_scholes(S1, K, T, r, sigma, option_type='call')
            profits.append(exit_price - bs_price)
            exit_dates.append(exit_date)

    trade_details_black_scholes = pd.DataFrame({
        'Entry Date': entry_dates,
        'Exit Date': exit_dates,
        'Profit': profits
    })

    trade_details_black_scholes.to_csv('black_scholes_trade_details.csv', index=False)
```

(b) Code Snippet 2

Figure 1: Python implementation snippets

# 7 Empirical Analysis

## 7.1 Data Collection

Collected historical stock data from Yahoo Finance and Quandl for firms like Apple Inc. and indices like NIFTY 50.

## 7.2 Preliminary Results

- Compared theoretical vs market option prices.
- Noted deviations due to dividends, early exercise, volatility skew.

### 7.3 Challenges Faced

- Handling missing values and noise in data
- Estimating implied volatility
- Modeling American options

## 8 Comparison Table: Model Features

| **Model** | **Underlying Assumptions** | **Option Types** | **Pros/Cons** |
|---|---|---|---|
| Binomial | Discrete steps with recombining tree; assumes known volatility | American/European | Intuitive and easy to understand, but computationally slow for large $n$ |
| Black-Scholes | Log-normal asset returns, constant volatility and interest rate | European | Fast and analytical; lacks flexibility for early exercise or path dependency |
| Monte Carlo | Stochastic process (e.g., Geometric Brownian Motion); randomness in pricing paths | Complex/Exotic | Highly flexible; however, computationally intensive and slower to converge |

## 9 Next Steps

- Extend to American and exotic options
- Incorporate dividends, stochastic volatility
- Analyze sensitivity using Greeks: $\Delta, \Gamma, \Theta, \rho, \nu$
- Build real-time data pipeline
- Summarize findings in final report

## 10 Conclusion

We have established a comprehensive theoretical and practical foundation for option pricing, implemented key models, and initiated empirical validation. Future efforts will focus on advanced modeling, real-time data integration, and evaluating model robustness against real market behavior.

## 11 Code

You can access the code here: Finsearch Code Notebook.