Option Pricing Model Project Report

1. Introduction & Objectives

This self-initiated project aims to implement and analyze the **Black-Scholes option pricing model** for European call and put options. The primary objective is to understand option valuation mathematics, develop a working model in Python, and explore how key factors like volatility and time to expiry impact option prices.

2. Model Selection & Theory

We selected the **Black-Scholes formula** due to its widespread use in financial markets for pricing European-style options (no early exercise). The model relies on several input parameters:

- Current stock price (S_0)
- Strike price (K)
- Time to expiry (T), in years
- Risk-free interest rate (r)
- Annualized volatility (σ)

The Black-Scholes formula is given by:

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

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

where:

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

$N(x)$ represents the cumulative distribution function (CDF) of the standard normal distribution.

3. Implementation Details

- The project is coded in **Python** using `numpy` and `scipy.stats`.

- Input parameters (stock price, strike price, time to expiry, volatility, risk-free rate) are defined in the script.
- The code computes the call and put prices using all combinations of parameter values.
- The results are outputted as a CSV file (`option_price_table.csv`) for easy analysis.

Key Code Snippet

```
import numpy as np
from scipy.stats import norm
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 = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call
def black_scholes_put(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)
    put = K * np.exp(-r * T) * norm.cdf(-d2) - S * norm.cdf(-d1)
    return put
```

4. Results

The generated CSV table (`option_price_table.csv`) displays how option prices vary as stock price, time to expiry, and volatility change. Key insights observed:

- **Higher volatility** increases both call and put prices.
- **Longer time to expiry** usually increases price, especially for calls.
- **Stock price above the strike** favors call options; below, favors puts.

You may visualize these results using graphs for deeper understanding.

5. Limitations & Extensions

- Model works for **European options** (no early exercise), not American style.
- No dividend adjustments included; these can affect real-world pricing.
- Parameters were hardcoded for demonstration but can easily be replaced with live market data or read from files.
- Potential extensions: Implement binomial tree or Monte Carlo methods, build an Excel version, or apply to Indian markets using real data.

6. Files Included

- `black_scholes.py` — Python script implementing the model
- `option_price_table.csv` — Output table of computed prices
- `README.md` — Project overview and usage guide

7. References

- Black, F., & Scholes, M. (1973). "The Pricing of Options and Corporate Liabilities", Journal of Political Economy.

Monte Carlo Option Pricing Model Project Report

1. Introduction & Objectives

This project implements a **Monte Carlo simulation** approach for pricing European call and put options. The objective is to understand the simulation method for option valuation, implement it in Python, and analyze how option prices vary with key parameters such as volatility, time to expiry, and stock price.

2. Model Selection & Theory

Monte Carlo simulation is a numerical method used to estimate the expected payoff of options by simulating many possible future stock price paths based on stochastic processes.

The simulation assumes the stock price follows a **Geometric Brownian Motion**:

$$S_T = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}Z\right)$$

where $Z \sim N(0,1)$ is a standard normal random variable.

The option payoffs at maturity T are:

$$\text{Call payoff} = \max(S_T - K, 0)$$

$$\text{Put payoff} = \max(K - S_T, 0)$$

The option price is the discounted average payoff over many simulations:

$$\text{Option Price} = e^{-rT}\mathbb{E}[\text{payoff}]$$

3. Implementation Details

- Written in **Python**, using `numpy` for simulations and `pandas` for data handling.
- Conducts 10,000 random simulations (seeds set for reproducibility).
- Computes call and put prices for combinations of stock prices, strike prices, volatility, and time to expiry.
- Results are saved in a CSV file for further analysis.

Key Python Code Snippet

```
def monte_carlo_call(S, K, T, r, sigma, simulations=10000):
    np.random.seed(42)
    Z = np.random.standard_normal(simulations)
    ST = S * np.exp((r - 0.5 * sigma ** 2)*T + sigma * np.sqrt(T) * Z)
    payoffs = np.maximum(ST - K, 0)
    return np.exp(-r * T)*np.mean(payoffs)
```

4. Results and Observations

- The generated CSV (`monte_carlo_option_price_table.csv`) displays option prices across multiple scenarios.
- Option prices increase with volatility and time to expiry.
- Prices behave logically relative to the position of stock price compared to strike.
- Monte Carlo prices closely approximate theoretical values given a large number of simulations.

5. Limitations & Future Work

- The simulation approach is computationally intensive compared to closed-form solutions like Black-Scholes.
- Only prices European options with no early exercise.
- Does not incorporate dividends or dynamic interest rates.
- Can be extended to price American options, path-dependent options, or include variance reduction techniques for faster convergence.

6. Files Included

- `monte_carlo_option_pricing.py` — Python simulation code
- `monte_carlo_option_price_table.csv` — Output results
- `README.md` — Documentation and usage instructions

7. References

- Glasserman, P. (2004). Monte Carlo Methods in Financial Engineering.
- Hull, J. C. (2012). Options, Futures, and Other Derivatives.

Binomial Option Pricing Model Project Report

1. Introduction & Objectives

This project implements the **Binomial Option Pricing Model** to price European call and put options. The primary objective is to understand option valuation through a discrete-time lattice approach and to observe how varying parameters affect option prices.

2. Theory

The binomial model divides the time to expiry into a number of discrete steps. At each step, the underlying stock price can either move up by a factor u or down by a factor d. The option price is calculated by working backward from maturity to the present using risk-neutral probabilities.

Key parameters and formulas:

- Time step: $\Delta t = \frac{T}{N}$
- Up factor: $u = e^{\sigma\sqrt{\Delta t}}$
- Down factor: $d = \frac{1}{u}$
- Risk-neutral probability: $p = \frac{e^{r\Delta t} - d}{u - d}$

At maturity, option payoffs are:

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

The option value at earlier nodes is the discounted expected value of future option prices.

3. Implementation Details

- Developed in Python using `numpy` and `pandas` libraries.
- Uses 50 discrete steps for the binomial tree for a balance between accuracy and computation time.
- Evaluates option prices for varying stock prices, strike prices, volatilities, and times to expiry.
- Outputs results in a CSV file for ease of analysis.

4. Results

- Option prices logically increase with greater volatility and longer times to expiry.
- Call and put prices reflect expected relationships with the underlying stock and strike price.
- The binomial model's prices serve as a discrete approximation to continuous-time models (e.g., Black-Scholes) and converge as the number of steps increases.

5. Limitations and Extensions

- Limited to European options; American options require additional logic for early exercise.
- Computational cost rises with increased steps.
- Assumes constant volatility and interest rates; does not account for dividends.
- Future work could involve implementing American option pricing or calibrating the model to market data.

6. Files Included

- `binomial_option_pricing.py`: Source code for the binomial pricing algorithm.
- `binomial_option_price_table.csv`: CSV with computed option prices.
- `README.md`: Documentation and usage instructions.

7. References

- Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). "Option pricing: A simplified approach," Journal of Financial Economics.