Monte Carlo Simulation of Option Pricing with Jump-Diffusion Model

Neerja Kasture 22110165
Ashmit Chhoker 22110040
Anura Mantri 22110144
Harshita Singh 22110299
Kaveri Visavadiya 22110114

October 3, 2023

Abstract

Our project explores the Monte Carlo simulation technique to estimate option prices within a jump-diffusion framework. The primary objective is to develop a comprehensive understanding of the mathematical and computational aspects underlying this sophisticated financial modeling approach.

The model employed in this study combines two fundamental components: continuous diffusion, modeled by the Geometric Brownian Motion (GBM), and occasional jumps in the underlying asset's price, modeled as a Poisson process. The mathematical foundation for these components is discussed, and their integration into a coherent option pricing framework is explained.

Key aspects of the project include defining the parameters that influence option pricing, such as volatility, risk-free interest rates, and jump intensity, as well as the practical implementation of the simulation. The Monte Carlo simulation methodology is applied to generate numerous possible future price paths for the underlying asset. Asset price movements are simulated over discrete time intervals, option payoffs at maturity are calculated, and the average payoff across multiple simulations is computed.

1 Introduction

1.1 Options : Overview

Options are financial derivatives that give the holder the right, but not the obligation, to buy (call option) or sell (put option) a specific underlying asset, such as a stock or commodity, at a predetermined price (strike price) within a specified period (expiration date). Options provide investors with opportunities for hedging, speculating, and managing risk in financial markets. Options trading involves paying a premium to acquire the option, and the potential profit or loss depends on the underlying asset's price movement.

Options trading presents various challenges, including sensitivity to market volatility, the erosion of value due to time decay, and the need to select appropriate strike prices to achieve profitable outcomes. Market liquidity can vary, impacting transaction costs and execution quality. Traders must make accurate directional predictions and manage complex Greek values, such as Theta (price decay due to time) and Vega(to navigate options effectively. There's also the cost of hedging and the risk of assignment, particularly with American-style options. Unforeseen market events and the overall complexity of options further add to the challenges.

The multitude of challenges inherent in options trading poses difficulties for both traders and fin-tech companies. To address these issues effectively, they seek to analyze and understand the behavioral patterns of options markets. By gaining insights into these patterns, they can develop tailored trading strategies that account for factors such as volatility, time decay, and market liquidity. This data-driven approach is crucial for mitigating risks and optimizing returns in the complex world of options trading.

1.2 Problem Statement

To uncover the behavioral patterns of options we will be applying Monte Carlo numerical method on jump diffusion model. This approach allows them to simulate and analyze various scenarios, taking into account abrupt market movements and changes in asset prices. By running extensive simulations, we can better understand how options react to these dynamics and refine their trading strategies accordingly.

2 Assumptions and parameters

2.1 Assumptions

The Jump-Diffusion Model makes the following assumptions:

1. Asset prices can experience sudden, unpredictable jumps in addition to continuous random price movements.
2. The occurrence of jumps follows a Poisson process, with the jump size and jump frequency being random variables.
3. Volatility can vary over time, allowing for a more realistic representation of asset price dynamics.

We constraint our parameters to the following bounds:
$0 < \sigma < \infty$
$0 < m < 2$
$0 < v < \infty$
$0 \leq \lambda < 5$

This is because:
The volatility should be strictly positive
The mean of the jump shouldn't be more/less than +-100%.
The variance is strictly positive.
The intensity of the jump is a small number. This is because a higher intensity of jumps would imply a more erratic stock price, which is not realistic in most cases. Here the choice of 5 is somewhat arbitrary.

2.2 Parameters

We will use the following parameters:

$$
\begin{aligned}
S &: \text{Current Stock Price} \\
K &: \text{Strike Price} \\
T &: \text{Time to maturity in years} \\
\sigma &: \text{Annual Volatility} \\
m &: \text{Mean of Jump Size} \\
v &: \text{Standard Deviation of Jump Size} \\
\lambda &: \text{Jump Intensity} \\
dW(t) &: \text{Wiener Process} \\
N(t) &: \text{Compound Poisson Process}
\end{aligned}
$$

Current Stock Price: Market price of the underlying asset (typically a stock) at a specific point in time.

Strike Price: Predetermined price at which a buyer of an option (whether call or put) can purchase (in the case of a call) or sell (in the case of a put) the underlying asset upon exercise of the option.

Time to Maturity: "Time of maturity" refers to the future date when the option will expire. It's the point in time up to which the holder of the option has the right, but not the obligation, to exercise the option.

Volatility: This is the continuous component of the price movement and is analogous to the volatility parameter in the Black-Scholes model. It captures the regular, small-scale fluctuations in the price of the underlying asset and is typically denoted as σ or a similar notation.

Mean Jump Size: This is the expected (or average) value of the log-normal distribution which describes the magnitude of jumps when they occur. It gives an indication of the typical size of the jump one might expect when an unexpected event impacts the stock price.

Standard Deviation of Jump Size: The standard deviation of the jump size (also referred to as the jump volatility) represents the dispersion of the jump sizes around the mean jump size. It's a measure of the variability in the size of the jumps that occur in the stock price.

Jump Intensity: It refers to the frequency of large price jumps. Here it is the number of jumps in a year.

Wiener Process: The Wiener process accounts for the continuous price evolution. It's often used to model the unpredictable continuous movement of stock prices.

Compound Poisson Process: N(t) represents a Poisson process with the probability of k jumps occurring over the life of the option given by the Poisson probability mass function:

$$P(N(t) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!} \tag{1}$$

3 Modeling Methods : Jump Diffusion model

The Merton Jump diffusion model is a result of Robert C. Merton's 1979 paper 'Option Pricing When Underlying Stock Returns Are Discontinuous'. It is an extension of the Black Scholes model that account for its limitations.

In the Black-Scholes model, it's assumed that stock prices follow a geometric Brownian Motion. However, real-world stock prices often experience sudden 'jumps' due to significant events like takeover announcements or legal decisions. The Jump Diffusion model integrates the Black Scholes diffusion mechanism with an additional jump component. This jump component models the stock price leaps with log-normal jumps that are influenced by a Poisson process.

3.1 Continuous Diffusion Component

The core of the model is the Geometric Brownian Motion (GBM), which describes the continuous, stochastic evolution of the underlying asset's price, S_t. GBM is defined by the following stochastic differential equation:

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

- dS_t is the infinitesimal change in the asset price over a small time interval dt.
- r is the risk-free interest rate.
- σ is the volatility of the asset's returns.
- dW_t is a Wiener process (Brownian motion), representing random, normally distributed shocks.

3.2 Jump Component

In addition to the continuous price movement described by GBM, the model incorporates occasional jumps in the asset price. These jumps are modeled as a Poisson process with intensity λ, which means the number of jumps that occur in a time interval dt follows a Poisson distribution with mean λdt.

Each jump has a random size, which is drawn from a normal distribution with mean 0 and standard deviation σ_j. The jump size is the sum of these random draws. The jump component is added to the price at each time step where a jump occurs.

3.3 Equations

The Jump Diffusion Stochastic Differential Equation can be written as follows:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + (J-1)dN(t) \tag{2}$$

According to Black-Scholes model, the stock price satisfies the stochastic differential equation given by

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{3}$$

The SDE is easily solved using Itô's lemma. Let $Y_t = \log S_t$,

$$\begin{aligned} dY_t &= \frac{\partial Y_t}{\partial S_t} dS_t + \frac{1}{2} \frac{\partial^2 Y_t}{\partial S_t^2} (dS_t)^2 \\ &= \frac{1}{S_t} dS_t - \frac{1}{2S_t^2} (dS_t)^2. \end{aligned} \tag{4}$$

We have

$$\begin{aligned} (dS_t)^2 &= \mu^2 S_t^2 dt^2 + 2\mu\sigma S_t^2 dt\, dW_t + \sigma^2 S_t^2 dW_t^2 \\ &= \sigma^2 S_t^2 dt. \end{aligned} \tag{5}$$

Note that $\mathcal{O}(dt^2)$ and $\mathcal{O}(dt^{\frac{3}{2}})$ terms go to zero and $(dW_t)^2 = \mathcal{O}(dt)$.

Therefore, we have

$$\begin{aligned} dY_t &= \frac{1}{S_t}(\mu S_t dt + \sigma S_t dW_t) - \frac{1}{2S_t^2}(\sigma^2 S_t^2 dt) \\ &= \left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma dW_t. \end{aligned} \tag{6}$$

Because $dY_t = \log S_t - \log S_0$ and $dW_t = W_t - W_0 = W_t$, we have

$$S_t = S_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right) t + \sigma W_t\right). \tag{7}$$

We can also write this as:

$$n(S_T) = \ln(S) + \int_0^T \left(\mu - \frac{\sigma^2}{2}\right) dt + \int_0^T \sigma dW(t) \tag{8}$$

Then the corresponding jump diffusion equation would be:

$$\ln(S_T) = \ln(S) + \int_0^t \left(\mu - \frac{\sigma^2}{2}\right) dt + \int_0^t \sigma dW(t) + \sum_{j=1}^{N(t)} Q_j - 1 \tag{9}$$

4 Numerical Methods applied

4.1 Monte Carlo Method

How Does the Monte Carlo Simulation Method Work? The Monte Carlo method acknowledges an issue for any simulation technique: the probability of varying outcomes cannot be firmly pinpointed because of random variable interference. Therefore, a Monte Carlo simulation focuses on constantly repeating random samples.

A Monte Carlo simulation takes the variable that has uncertainty and assigns it a random value. The model is then run and a result is provided. This process is repeated again and again while assigning many different values to the variable in question. Once the simulation is complete, the results are averaged to arrive at an estimate.

The 4 Steps in a Monte Carlo Simulation

Step 1: To project one possible price trajectory, use the historical price data of the asset to generate a series of periodic daily returns using the natural logarithm (note that this equation differs from the usual percentage change formula):

$$\text{Periodic Daily Return} = \ln\left(\frac{\text{Day's Price}}{\text{Previous Day's Price}}\right) \tag{10}$$

Step 2: Let r be the series of periodic daily returns. Calculate:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} r_i \quad \text{(This is the average daily return, } \mu\text{)}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(r_i - \mu)^2 \quad \text{(This is the variance, } \sigma^2\text{)}$$

$$\sigma = \sqrt{\sigma^2} \quad \text{(This is the standard deviation)}$$

The drift is then defined as:

$$\text{Drift} = \mu - \frac{\sigma^2}{2} \tag{11}$$

Alternatively, drift can be set to 0; this choice reflects a certain theoretical orientation, but the difference will not be significant, especially for shorter time frames.

Step 3: Obtain a random input:

$$\text{Random Value} = \sigma \times Z \tag{12}$$

where Z is a random value from a standard normal distribution.

The equation for the following day's price is:

$$\text{Next Day's Price} = \text{Today's Price} \times e^{\text{Drift+Random Value}} \tag{13}$$

Step 4: To take e to a given power x, use the exponential function. Repeat this calculation the desired number of times. (Each repetition represents one day.) The outcome is a simulation of the asset's potential future price trajectory.

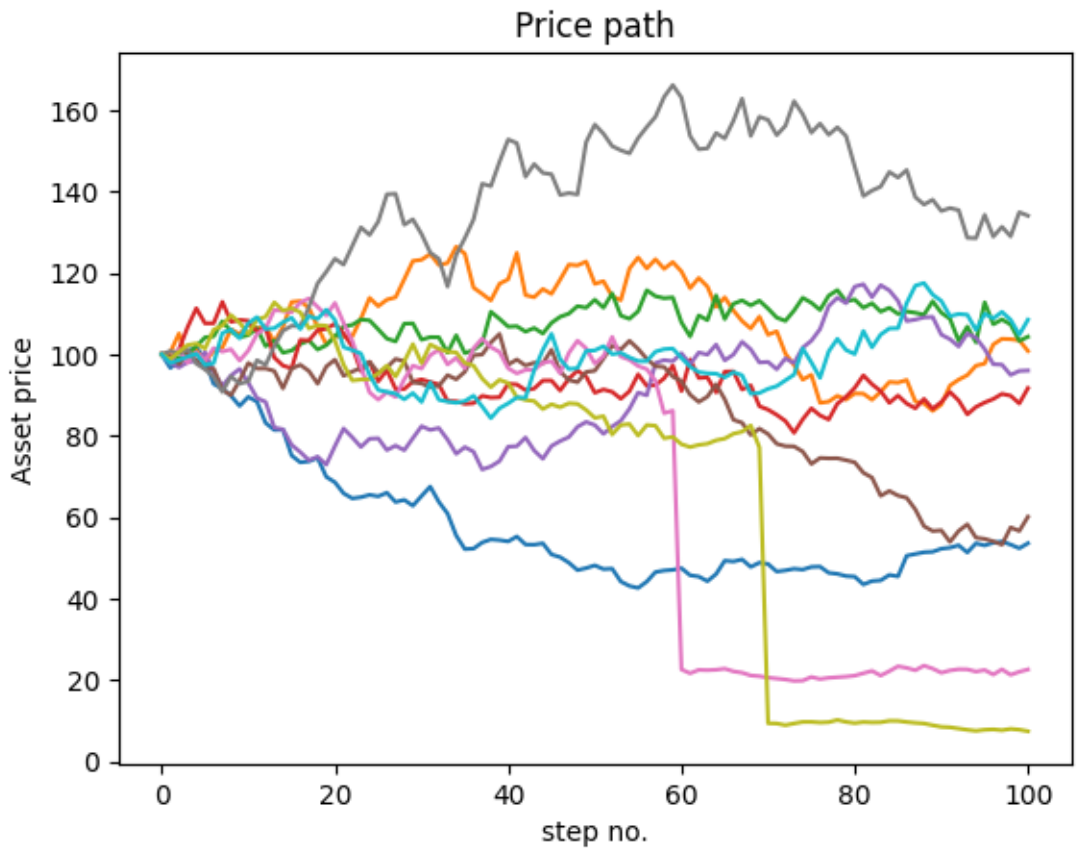
By generating an arbitrary number of simulations, you can gauge the probability that a security's price will follow a given trajectory.

5 Application of Monte Carlo on Option Pricing

5.1 Monte Carlo Simulation

To estimate the option price, a Monte Carlo simulation is performed. It involves the following steps:

1. Initialize the stock price at S_0 and define the option payoff at maturity.
2. Divide the time to maturity (T) into n_{steps} small time intervals of length $dt = \frac{T}{n_{\text{steps}}}$.
3. For each simulation ($n_{\text{simulations}}$), simulate the price path of the underlying asset from time $t = 0$ to $t = T$.
   (a) At each time step, calculate the next price using the GBM equation.
   (b) Determine the number of jumps in this time step using a Poisson process.
   (c) Calculate the size of each jump as the sum of random draws from a normal distribution.
   (d) Add the jump component to the price.
4. Calculate the option payoff at maturity (e.g., European call option payoff as $\max(S_T - K, 0)$).
5. Compute the average of the option payoffs across all simulations.
6. Discount the average payoff back to present value using the risk-free interest rate to obtain the estimated option price.



The above graph gives a visual representation of 20 simulations for an asset having price:100, volatility:0.3 and jump intensity:0.2

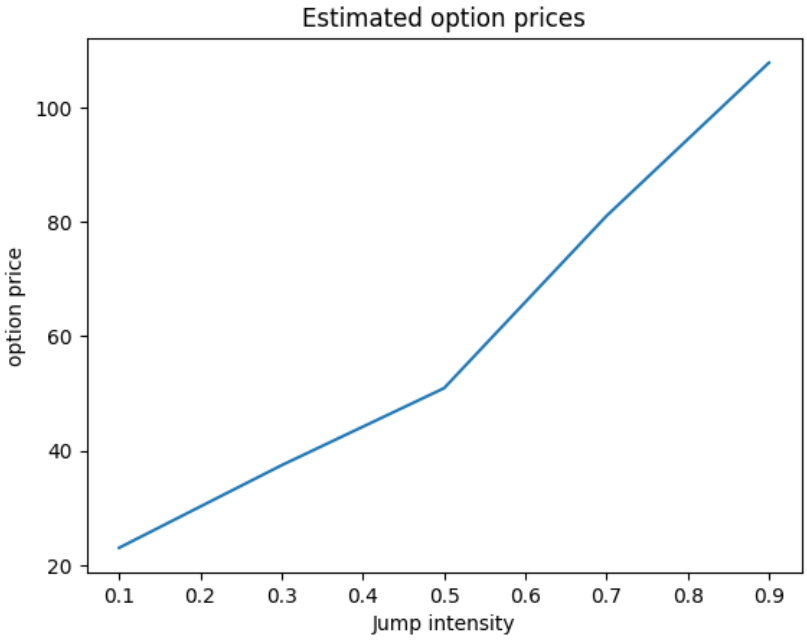
5.2 Simulation Code

```python
import numpy as np
import matplotlib.pyplot as plt
def jump_diffusion_monte_carlo(S0, K, T, r, sigma, lambd, jumps, n_simulations, n_steps):
    dt = T / n_steps
    discount_factor = np.exp(-r * T)

    option_prices = []
    for _ in range(n_simulations):
        price_path = [S0]
        for _ in range(n_steps):
            Z1 = np.random.normal(0, 1)
            N = np.random.poisson(lambd * dt)
            Z2 = np.random.normal(0, 1, N)
            jump = np.sum(Z2)
            next_price = price_path[-1] * np.exp((r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * Z1 + jump)
            price_path.append(next_price)

        option_payoff = max(price_path[-1] - K, 0)  # European call option payoff
        option_prices.append(option_payoff)
    option_price = np.mean(option_prices) * discount_factor
    return option_price
# Example usage:
S0 = 100.0  # Initial stock price
K = 100.0   # Strike price
T = 1.0    # Time to maturity (in years)
r = 0.05   # Risk-free interest rate
sigma = 0.3  # Volatility
lambd = 0.1  # Jump intensity
jumps = 1  # Number of jumps in the model
n_simulations = 1000 # Number of Monte Carlo simulations
n_steps = 100  # Number of time steps
option_price = jump_diffusion_monte_carlo(S0, K, T, r, sigma, lambd, jumps, n_simulations, n_steps)
print(f"Option Price: {option_price:.4f}")
```

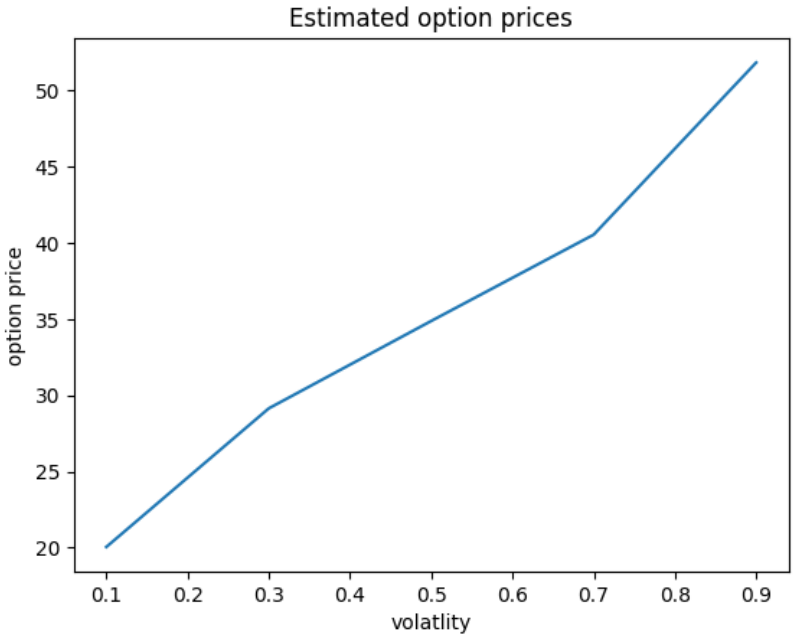
```
Option Price: 23.4525
```

6 Results

For a single simulation:

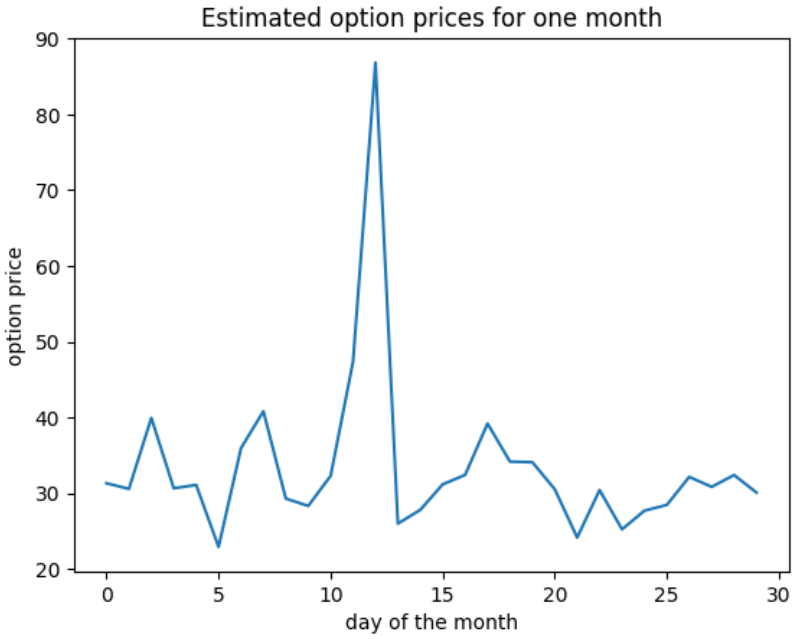


This graph shows that estimated option price increases as jump intensity increases



This graph shows that estimated option price increases as volatility increases
Thus we can infer that more unstable the market, more the price of the option.

Given below is a chart showing option prices estimated for a month using said method:



7 Conclusion

1. Monte Carlo methods are useful numerical methods in situations where analytical solutions are difficult or impossible to obtain due to the complexity of the problem or the presence of numerous variables and uncertainties.

2. They are particularly useful for visual simulations due to their ability to generate a range of potential outcomes by randomly sampling different scenarios. This makes it possible to visualize the distribution of results and understand how different variables interact with each other in a model, whereas this would be computationally difficult with other numerical methods.

3. Monte Carlo methods are versatile and powerful, but their accuracy depends on the number of samples used. Increasing the number of samples generally improves the accuracy of the estimate but also requires more computational resources. Therefore, the trade-off between accuracy and computational cost is an important consideration when using Monte Carlo methods.

Acknowledgement

We like to thank our professors Dilip Sundaram and Akshaa Vatwani for giving us the opportunity to work on this project in our MA203 course.

References

[1] "Option Prices in Merton's Jump Diffusion Model - Wolfram Demonstrations Project," Wolfram.com, 2011.
https://demonstrations.wolfram.com/optionpricesinmertonsjumpdiffusionmodel/

[2] Robin-Guilliou, "Option-Pricing/European Options/Merton Jump Diffusion at main · Robin-Guilliou/Option-Pricing," GitHub, 2021.
https://github.com/Robin-Guilliou/Option-Pricing/tree/main/European%20Options/Merton%20Jump%20Diffusion (accessed Oct. 03, 2023).

[3] "Merton Jump Diffusion Model with Python," www.codearmo.com. https://www.codearmo.com/python-tutorial/merton-jump-diffusion-model-python (accessed Oct. 03, 2023).

[4] W. Kenton, "Monte Carlo Simulation," Investopedia, Mar. 26, 2023.
https://www.investopedia.com/terms/m/montecarlosimulation.asp

[5] K. Agarwal, "What Can The Monte Carlo Simulation Do For Your Portfolio?," Investopedia, 2019.
https://www.investopedia.com/articles/investing/112514/monte-carlo-simulation-basics.asp