Monte Carlo Option Pricer – Model

What is Monte Carlo Simulation?

Monte Carlo is a broad class of computer simulated algorithms that rely on repeated random experiments, or simulations to estimate an uncertain event, or a complex function. We could argue that there may exist some deterministic formula for the estimation of the price of an option, given parameters (e.g. Black Scholes), however, it is highly unlikely to be accurate 100% of the time. The underlying concept behind Monte Carlo simulation is to use randomness to solve a problem that might be deterministic. The MC simulation relies on the law of large numbers, where we can approximate the expected value of a complex probability distribution by obtaining a large enough sample and taking the average of it.

How do we apply it to option pricing?

Random, but possible paths for the option are simulated in a computer environment, then are averaged out to obtain a deterministic value for the most likely price of the option. However, what exactly are the parameters that are used to determine the simulation? Usually, for more simple models, we stick to just using the underlying stock price as the dependent, which is usually just modelled using the Geometric Brownian Distribution with constant drift (mu) and constant volatility.

Model/Distribution Selection?

Monte Carlo simulations often rely on the Geometric Brownian Motion (GBM) model, which assumes a random walk to simulate stock price movements. While GBM is effective for modeling general trends, it lacks the ability to capture the dynamics of more volatile market conditions. To address this limitation, we incorporate Maximum Likelihood Estimation (MLE) to calibrate the parameters specific to each stock, ensuring the GBM accurately reflects individual price behaviors.

However, during periods of heightened volatility, the Heston Volatility Model provides a more realistic alternative by accounting for stochastic volatility, enabling a better representation of market dynamics under extreme conditions.

To further enhance the simulation's accuracy, Hidden Markov Models (HMMs) are employed to predict market regimes. By identifying these regimes based on historical stock price data, the model can dynamically determine whether to apply GBM or the Heston Volatility Model, ensuring an adaptive approach to stock price simulation.

Lastly, to capture abrupt and significant financial events, the Poisson process is used to simulate market shocks. This allows the model to integrate rare but impactful events, improving the robustness of the simulation in replicating real-world market behavior.

This multi-layered modeling approach leverages the strengths of each technique, creating a comprehensive framework for analyzing stock price dynamics under varying market conditions.

Project Outline:

- Objective: Develop a monte Carlo simulation-based tool to price options (American Options)
- Mathematical Concepts
    - Brownian Motion (Random Walk)
    - Geometric Brownian Motion
    - Risk-neutral valuation
- Deliverables:
    - Jonathan makes a github repository and sets up the environment.
    - Gather 5 years of data (2017 - 2022), Use 2023 as testing year.
        - Choose universe of options to trade from (list of companies, ~100 - 200 companies)
    - MC Simulation Framework (1):
        - Generate multiple simulated paths for the underlying asset using Geometric Brownian Motion. Estimate expected payoffs (Average of all the simulated paths) under risk neutral measures.
    - Option Pricing Outputs (2):
        - Estimate the price for the option. Convergence analysis. What does the option price converge on as the number of simulations increases?
    - Black Scholes Model (3):
        - Estimate parameters for the Black Scholes Model (implied vol, stddev, mean, etc.).
        - Input into BSM, then can average out with the option price calculated from Monte Carlo Simulation
    - Further (4):
        - Can visualize price paths
        - Support for different price dynamics (stochastic volatility) (Markov Chain to model price dynamics)
        - Potentially backtest this live, then graph the returns to show legitimacy

    - Further (Optional) (5):
        - Optimize using *antithetic variables*
        - Incorporate stochastic volatility
        - Speed up simulations using parallel computing (TensorFlow)
- Finalize and compile together.
    - The final output of the project will be a Pandas DataFrame for a single long position in both the call and put option for the respective stock, that is a percentage away from the current price, in T time.

- Necessary Libraries:
    - NumPy, Pandas, (TensorFlow), (SciKit learn), SciPy, Matplotlib
    - HMMLearn: For hidden markov models

Progress Documentation:

Part I – Model Selection & Parameter Estimation:

To begin simulating using the Monte Carlo method, we first need to be able to simulate the stock prices. To do this, we decided to use the Geometric Brownian Motion with dynamic variables to simulate a more realistic environment. The dynamic variables will have parameters calculated by related models, which come from real stock data. In a nutshell, the volatility will be calculated using Heston's model, with parameters being estimated using the Maximum Likelihood Estimation method. (insert how more parameters will be determined)

Geometric Brownian Motion:

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

There are many parameters that need to be calculated (or estimated, to be more precise). Many of the parameters used in Geometric Brownian Motion and Heston's model are shared, however, one such parameter that is unique to the GBM is the drift, mu. To estimate this, we will apply the Maximum Likelihood Estimation technique, which is as follows:

- We know the returns of the Geometric Brownian Motion should theoretically follow the Normal Distribution. Hence, we are just maximising the Normal distribution.

1. Log Returns Distribution:
   $R_t = \ln(S_t/S_{t-1})$, and: $R_t \sim N(\mu_{\text{daily}}, \sigma^2_{daily})$
2. Likelihood Function:

$$L(\mu, \sigma) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(R_t - \mu)^2}{2\sigma^2}\right)$$

3. Taking the log to get the Log Likelihood:

$$\ln L\,(\mu, \sigma) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{\sum_{t=1}^{n}(R_t - \mu)^2}{2\sigma^2}$$

4. Maximum Likelihood Estimator for mu:

$$\hat{\mu} = \frac{1}{n}\sum_{t=1}^{n} R_t$$

Which is simply the sample mean. The sample standard deviation is also the MLE estimator for the StdDev.

Heston's Model:

$$dv_t = \kappa(\theta - v_t)\ , dt + \sigma_v \sqrt{v_t}\ , dW_t, \quad v_t > 0$$

Unfortunately, there are no closed form solutions for the Likelihood Function for Heston's model. To estimate the parameters kappa, theta, sigma_v and initial variance, we will utilize the Euler – Maruyama discretization method, which aims to approximate the numerical solution of a stochastic differential equation. This works by approximating the changes at each time increment dt, until we have reached T. Since approximations are small and at each day, they become more accurate than approximating it. A potential step by step may look like:

1. Variance Process:

$$\left[dv_t = \kappa(\theta - v_t)dt + \sigma_{v\sqrt{v_t}}, dW_t^v\right]$$

   Where:

- v_t is the variance at time t
- dt Is the incremental time
- κ is the mean reversion rate
- σ_v is the volatility of the volatility
- W_t^v is the Wiener process for Variance

2. Price Process:

$$S_{k+1} = S_k + \mu S_k \Delta t + \sqrt{\max(v_k, 0)}\, S_k \Delta W_k^S$$

Where:

- $\Delta W_k^S = \rho \Delta W_k^v + \sqrt{1-\rho^2}\Delta Z_k$, and
- $\Delta Z_k \sim N(0, \Delta t)$

3.

I) Given initial conditions S_0 and v_0

Generate two independent normal variables at each step

- $\Delta W_k^v \sim N(0, \Delta t)$
- $\Delta Z_k \sim N(0, \Delta t)$

II) Compute Correlation Adjusted Wiener Increment

$$\Delta W_k^S = \rho \Delta W_k^v + \sqrt{1-\rho^2}\Delta Z_k.$$

III) Update the Variance

$$v_{k+1} = v_k + \kappa(\theta - v_k)\Delta t + \sigma_{v\sqrt{\max(v_k,0)}} \Delta W_k^v$$

IV) Update the Price

$$S_{k+1} = S_k + \mu S_k \Delta t + \sqrt{\max(v_k, 0)}\, S_k \Delta W_k^S.$$

However, this is quite tedious and difficult to implement, and with the existence of SciPy's "optimize" library, we do not need to go through the entire Euler Maruyama Discretization Process. We will use SciPy's "optimize" library to do much of the grunt work for us by efficiently optimizing these variables to find the Maximum Likelihood Parameters.

We begin by finding the Log Likelihood contribution for each step, which is:

$$\mu_t = v_{t-1} + \kappa(\theta - v_{t-1})\Delta t$$

Where the variance of the simulation is:

$$\text{Variance}_t = \sigma_v^2 v_{t-1} \Delta t$$

When we put this together, the combined log-likelihood function for the Heston Volatility Model is:

$$\ell = -\frac{1}{2}\sum_{t=1}^{n}\left(\ln(2\pi \cdot \text{Variance}_t) + \frac{\left(v_{\text{observed},t} - \mu_t\right)^2}{\text{Variance}_t}\right)$$

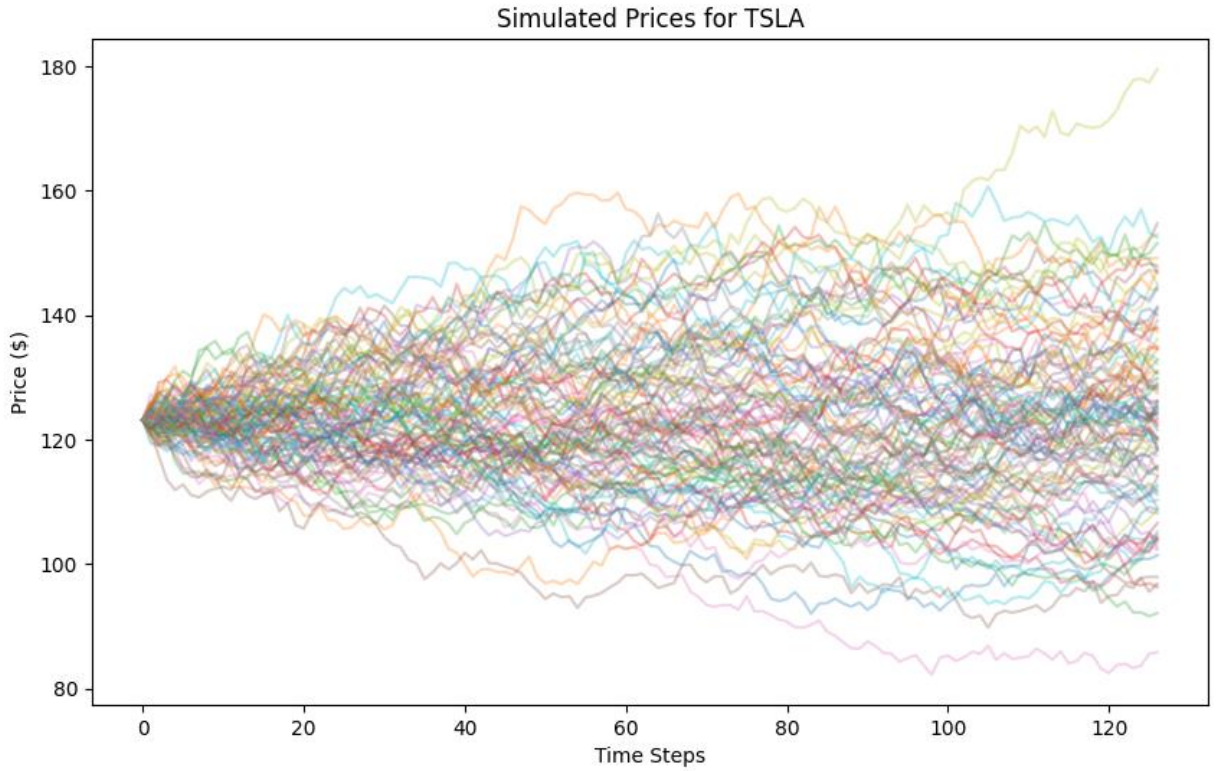
Python will then optimize this equation to figure out the maximum likelihood estimator for each parameter of the Heston Volatility Model.

Part II – Volatility and Regime Modelling:

We now have two working models to use to simulate stock prices. Obviously, these stochastic models do not mirror real life stock prices and are just for the sake of estimating potential option prices in the future. To utilize these two models at the same time, the Heston Volatility Model will be used during periods of high volatility and the Geometric Brownian Motion will be used in periods of low volatility *(COULD CONSIDER WEIGHTING).*

To determine when the stock market is in a period of high or low volatility, a Hidden Markov Model (HMM) is used. A Hidden Markov Model can utilize the observable variable (stock prices), to estimate the hidden variable (regime). The model is first trained on ~20 years of S&P 500 data and classifies periods into either 0 or 1. The periods are then grouped together, and the standard deviation of the subgroups are calculated. Whichever period has the higher standard deviation will have the "High Volatility" marker and the lower will have the converse.

The model is now trained, and its transition matrix will now be used to simulate T/dt days into the future, depending on when the option's expiry is. The function responsible will generate a list of days into the future and its corresponding predicted regime. Noting that it is a stochastic model, results may vary per run.



Shown above is one simulation for the TSLA stock with 100 Monte-Carlo simulations using the dynamic Heston Volatility and Geometric Brownian Motion model.

Part III – Implementing the Poisson Process

To capture unexplained shocks in the stock prices, it may be wise to implement the Poisson Process – continuous time process that captures events that happen at a given rate lambda. The time in between these events, or shocks in this context, also known as interarrival times, can be modelled by the exponential distribution. Obviously, for different stocks, the lambda rates will be different, which mostly depends on the stock's volatility and sensitivity to market movements.

To estimate the Poisson lambda rate, the Maximum Likelihood Estimation technique was used, like the other distributions used to construct this model. In summary, the maximum likelihood estimator for the lambda parameter is simply the sample mean of the observations, which is derived below.

$$P(X = x \mid \lambda) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad x = 0,1,2,\dots$$

For a dataset of $\{x_1, x_2, \dots, x_n\}$ independent observations, the likelihood function is:

$$L(\lambda \mid x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$$

We will take the log of this likelihood function to simplify it and convert it to a summation instead.

$$\ell(\lambda) = \ln L(\lambda) = \sum_{i=1}^{n} \ln\left(\frac{\lambda^{x_i} e^{-\lambda}}{x_i!}\right)$$

$$\ell(\lambda) = \sum_{i=1}^{n} (x_i \ln \lambda - \lambda - \ln(x_i!))$$

And since $\ln(x_i!)$ does not depend on λ, we can omit this. It will be differentiated out in the process regardless.

$$\ell(\lambda) = \sum_{i=1}^{n} x_i \ln \lambda - n\lambda$$

Differentiating the log likelihood function with respect to λ:

$$\frac{\partial \ell(\lambda)}{\partial \lambda} = \sum_{i=1}^{n} \frac{x_i}{\lambda} - n$$

Set the equation to 0:

$$\sum_{i=1}^{n} \frac{x_i}{\lambda} - n = 0$$

Simply rearranging the equation for λ gives:

$$\hat{\lambda} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Hence, the maximum likelihood estimator for the Poisson distribution is simply the sample mean of all observations. Real stock observations will be used to calculate the Poisson lambda rate. The extent of the shock will also need to be determined. To calculate how large this shock is, the log-normal distribution will be used. There are a few characteristics of the log-normal distribution $\left[f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}, \quad x > 0\right]$ that make it appropriate for this application:

1. The log-normal distribution is only for positive values, which is suitable, as stock prices cannot be negative.
2. The Geometric Brownian Motion is commonly used for the modelling of stock prices, which assumes a log-normal distribution for the price.
3. There is also empirical evidence that the returns of the financial markets often follow a log-normal distribution, often over extended periods of time. Price shocks tend to reflect a combination of compounding effects that align well with the log-normal distribution's properties.

Hence, the Poisson distribution will determine when the shocks occur and how many shocks will occur for this given stock.

However, we will need to determine whether these shocks are positive or negative. The model is already complicated as it is, so further complication via utilizing another Hidden Markov Model may slow down computations. To simplify this somewhat, a simple probabilistic model based off on the stocks past performance is used. The polarity of the shock is modelled by a random choice, either a 0 or 1 with probabilities trained on past performance. Over large samples/simulations, the expected value of the shocks should accurately and adequately be reflected in the monte-carlo simulations.

Part IV – Calculating Option Price:

To figure out the option pricing from these stock simulations, we first need to compile a list of strike prices for the given option. To simplify, we have decided to go with a 10% and 20% discount from the current price, as well as a 10% and

20% increase from the current price in 3-month increments, all the way to one year. To figure out the time value of this option, there will also be an option which strike price is the same as the current price.

A simplified process to calculate the option premium may look like:

1. Determine strike prices (10% & 20% in both directions from the current price)
2. We will determine the value of the long call and the long put.
    a. To find this, we will need the option profiles for both, which are:
    b. Long Call: $\text{Payoff} = \max(S_t - K, 0)$
    c. Long Put: $\text{Payoff} = \max(K - S_t, 0)$
3. The option will then need to be discounted, which follows the equation:

$$e^{-rT} \cdot \text{Payoffs}$$

4. The estimated option price will then be the geometric mean of all the simulated option prices.

Where:

- r : Is the risk-free rate at the time

The risk-free rate of return is obtained from the US treasury 10-year bond.

Finally, we will need to account for the time value of the option. A simplistic model for the time value of the option is used: $\sigma \cdot \sqrt{T} \cdot k$, where:

- σ: is the volatility of the underlying stock
- T : is the time to expiration
- k : is a scaling factor, such that it is proportional to the price of the underlying stock

Through experimentation, the model's k value was 0.25. This was done on repeated trials of comparing the time value of options to real options listed on Yahoo.

Thus, we have completed the model of simulating the option premium of a stock. We will now also calculate the put option premium for the same strike price, along with the same parameters. For a sanity check, we will also feed this into the Black Scholes Formula, to ensure that the model's pricing is somewhat like that of the Black Scholes Model.

Areas for Improvement:

The model somewhat undervalues some options when compared with the Black Scholes Equation, especially for NVDA, potentially due to some parameters being estimated inaccurately. Due to the nature of the maximum likelihood method, the sample size may not be large enough to determine that that is the true parameter. The model can be improved by training on larger sets of data, however, that will require further computational resources and may also put the model at risk of overfitting. There is also some instability within the Hidden Markov Model, where running the simulation a few times may produce varied results, as with the nature of stochastic models.

Further, there are obviously many factors that the model has not considered. Given the rapidly changing nature of the stock market and options, it may also be useful to implement a machine learning algorithm that is able to scan the overall sentiment of recent news and to be able to convert that sentiment into quantitative data for analysis in the model. The model can also further apply more advanced machine learning models for regime detection and prediction of the polarity of the shocks, as well as other alternative stochastic models to capture the complex volatility of the stock market.

Conclusion:

This project integrates advanced financial modelling to simulate stock prices and determine option prices with a robust and multi-faceted approach through the utilization of numerous statistical models to capture movements in the markets. By combining models, namely the Geometric Brownian Motion, Heston Volatility Model, Poisson Distribution/Process and the Black Scholes Framework, we have created a dynamic and adaptable model capable of analyzing market behavior and evaluating derivative pricing.

The further incorporation of shocks allows the capture of sudden movements in the market and enhances the realism of the simulation. The model also allows for scalability whilst allowing for seamless transitions and adjustments to model parameters such as stock universes, time horizons, which allows it to remain useful when analysing various market conditions.

While the results demonstrate the effectiveness of market data and the model, there are many areas for improvement as mentioned earlier. However, this project overall provides a solid grasp on the fundamentals of market dynamics and financial decision making, allowing users to derive insights from the complex financial markets.