Risk Assessment in Financial Markets: Monte Carlo and Heston Model Simulations in Rust

Christopher Lee

March 2025

Contents

1 Introduction

DISCLAIMER AND BACKGROUND: I am in no way shape or form an expert in this topic. The brunt of my knowledge is purely qualitative, and required reading and borrowing other people's knowledge to implement (all of whom are cited and listed in the references). As for my code, you can find it on my github page: HERE (it's a hyperlink). I tried my best to program in the most blazing-fast way possible, but I'm still learning! If I get any information wrong (or you have any cool optimizations for my code) please let me know by emailing me at jleechris06@gmail.com.

Understanding risk and predicting stock price movements are key parts of financial modeling. Since markets are unpredictable, we use simulations to model how prices and volatility might behave over time. Two popular ways to do this are Monte Carlo simulations and the Heston stochastic volatility model. These models help investors, traders, and financial analysts estimate future price movements and assess risks, making them essential tools in quantitative finance.

Monte Carlo simulations are a great way to estimate where stock prices might go by running a bunch of random scenarios based on probability. They're widely used for pricing options, managing risk, and optimizing portfolios. The idea is simple: generate many possible future stock price paths based on assumed probabilities and then analyze the results to get an idea of expected returns and potential risks. However, one downside is that traditional Monte Carlo models assume volatility stays the same, which isn't very realistic in actual markets. In reality, volatility tends to change over time due to various economic and market factors, making basic Monte Carlo models less accurate in capturing market behavior.

The Heston model fixes this by making volatility random instead of constant, which makes simulations more accurate. Since real markets tend to have volatility that changes over time, this model better captures patterns like sudden jumps in volatility or how stocks and volatility are related. The Heston model introduces a stochastic process for volatility, meaning it follows its own random path rather than staying fixed. This allows for more realistic modeling of financial markets, helping analysts better understand market fluctuations and risks associated with different investment strategies.

This paper goes over how Monte Carlo and Heston model simulations work for stock price modeling and risk assessment. The goal is to break down the theory behind these models, show how they were coded in Rust, and analyze how they help assess financial risk. By simulating stock price and volatility paths, we can measure risk using things like Value at Risk (VaR) and better understand market uncertainty. VaR is an important risk metric that helps quantify potential losses in a given time frame with a certain level of confidence, making it a useful tool for portfolio managers and risk analysts.

Beyond just looking at theory, this paper will also dive into the practical implementation of these models using Rust. Rust is a fast and memory-efficient programming language, making it well-suited for handling large-scale simulations. The implementation will focus on optimizing computations and leveraging parallel processing to handle thousands of stock price and volatility simulations efficiently. Additionally, the paper will explore how these simulations can be visualized to provide better insights into market behavior.

Here's how the paper is structured: Section 2 explains the theory behind Monte Carlo simulations and the Heston model, including their mathematical foundations and key assumptions. Section 3 covers how they were implemented in Rust, detailing the coding approach, optimizations, and challenges faced. Section 4 discusses key takeaways, limitations of the models, and potential future improvements, such as exploring alternative stochastic models or improving computational efficiency.

2 Theoretical Knowledge

2.1 Financial Modeling Using Monte Carlo Simulations

Monte Carlo simulations are a statistical method used to model uncertainty in various fields, including finance. The core idea behind Monte Carlo methods is to use random sampling to generate a range of possible outcomes for a given financial variable, such as stock prices or option values. This approach is widely used in risk assessment, derivative pricing, and portfolio optimization (Glasserman, 2004).

The mathematical foundation of Monte Carlo simulations lies in the repeated random sampling of variables to approximate expected values. In finance, this involves using stochastic differential equations (SDEs) such as the Geometric Brownian Motion (GBM) model, which assumes that stock prices follow the equation:

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

Where S_t represents stock price at time t, μ represents expected return, σ represents volatility, and W_t represents random fluctuations. (Hull, 2017)

Monte Carlo methods are useful because they allow for the estimation of financial metrics such as Value at Risk (VaR), which helps in understanding potential losses under uncertain market conditions. However, a major limitation of basic Monte Carlo models is the assumption that volatility remains constant over time, which is not realistic in financial markets (Boyle, Broadie, & Glasserman, 1997).

2.2 Stochastic Volatility Using a Heston Model

The Heston model improves upon traditional models by incorporating stochastic volatility, meaning that the volatility of a stock price is also a random process rather than a fixed value. This makes it more suitable for capturing real-world market dynamics, such as volatility clustering and the leverage effect (Heston, 1993).

The Heston model is defined by the following system of stochastic differential equations:

$$dS_t = \mu S_t dt + \sqrt{v_t S_t} dW_t^1$$

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t} dW_t^2$$

Where v_t represents random variance, κ represents the rate that the variance tends to the long term mean θ, ξ is the volatility of variance (depends on how much variance can change over time), and W_t^1 and W_t^2 are two Wiener processes with correlation ρ

Definition: A sequence of random variables $B(t)$ is a Wiener process if $B(0) = 0$, and for all t, s such that $s < t$, $B(t) - B(s)$ is normally distributed with variance $t - s$ and the distribution of $B(t) - B(s)$ is independent of $B(r)$ for $r \leq s$ (quantstart, n.d.). The correlation ρ between the W_t^1 and W_t^2 allows the model to capture the leverage effect, where asset returns and changes in volatility are inversely related (Algorithm Trading Library, n.d.).

The introduction of a random variance term allows the Heston model to better reflect market behavior and improve risk assessment strategies, particularly in pricing derivatives and handling portfolio risks (Gatheral, 2006).

2.3 Comparing Monte Carlo and Heston Models in Risk Assessment

Both Monte Carlo simulations and the Heston model are useful for financial modeling, but they serve different purposes. Monte Carlo methods provide a flexible approach for pricing and risk management, especially when dealing with complex financial instruments. However, the assumption of constant volatility can limit its accuracy in certain market conditions.

The Heston model, on the other hand, improves accuracy by modeling volatility as a stochastic process, making it more effective for capturing market uncertainties. This is particularly useful in estimating financial risk metrics such as VaR and Conditional Value at Risk (CVaR), which depend on the accurate representation of volatility dynamics (Jäckel, 2002).

In the next section, we will discuss how these models were implemented in Rust, leveraging parallel computing techniques for efficient simulations.

3 Implementation and Methadology

3.1 Overview of Approach

The implementation of Monte Carlo and Heston model simulations in this "paper" was done using Rust, a systems programming language known for its speed, safety, and concurrency capabilities (Klabnik & Nichols, 2019). Rust's ownership model and built-in concurrency features make it particularly well suited for handling large-scale simulations efficiently (also because I really like rust and refuse to use Python).

Monte Carlo simulations rely on generating random samples to model different possible future states of stock prices. This requires using a random number generator to simulate the stochastic differential equation (SDE) determining price movements (this was stupid hard figuring out). Therefore I had to use a crate like rand which allows for generating normally distributed pseudo-random numbers.

The Heston model introduces an additional random process for volatility, requiring the simulation of two correlated Wiener processes. To manage these computations, I used Rust's tokio crate, which allows for async functions and parallelization. This is especially important when running thousands of simulations to estimate financial risk metrics accurately (I also wanted an excuse to learn async Rust even though it's bad).

3.2 Monte Carlo Simulation Implementation (Also Known as The Hard Part)

Monte Carlo simulations were implemented by approximating the Geometric Brownian Motion (GBM) equation using the Euler-Maruyama method. The stock price at each time step is updated using the formula (I DON'T UNDERSTAND THIS MATH BUT IT WORKS):

$$S_{t+1} = S_t \times e^{(\mu - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z_t}$$

Where Z_t is a standard normal random variable.

The way that I implemented this formula into my program is by using the sample function in the rand_distr crate. For each task that we create in our program (see lines 42-50 in main.rs) a stock price path which is represented as a vector contains all of the computations from $t = 1$ to the number of steps we take.

This approach is widely used in financial modeling for option pricing and risk assessment (Glasserman, 2004). The Rust implementation creates multiple simulated stock price paths using the tokio::spawn function to distribute computation across multiple threads, significantly improving performance.

A critical aspect of Monte Carlo methods is the ability to estimate Value at Risk (VaR). By analyzing the distribution of simulated stock price returns, we calculate the 5th percentile as the 95% VaR, providing insight into potential downside risk (Jäckel, 2002). This was implemented in the calculate_var_95 function in our program. We first calculate returns at each point in time throughout our simulation (stored as a vector). After sorting the returns, we can get the 5th percentile by indexing through the vector.

3.3 Heston Model Simulation Implementation (It Never Gets Easier)

The Heston model requires simulating both stock prices and stochastic volatility. This is achieved using the system of SDEs we previously talked about.

This is implemented in lines 84-98 of our program. For each simulation we perform, we calculate our correlated standard normal variables using our corelation variable ρ. After, we calculate volatility by using a mean reverting square-root process (CIR Process). This is an equivalent approximation of our Heston model, defined in 2.2. Finally, we model the log-price of our data in order to avoid any negative values. Looking at line 98, price_path is modeled by our GBM eqation in 3.2.

Additionally, the plotters crate was used to visualize simulation results, generating graphs of stock price paths and volatility movements.

3.4 Challenges and Optimization Strategies

One major challenge in implementing financial simulations is balancing accuracy and computational efficiency. Stochastic differential equations require many numerical approximation, and errors can accumulate over time if step sizes are too large (Higham, 2001). The implementation in this paper carefully selects time step sizes.

Another challenge is handling memory usage in large-scale simulations. Rust's ownership model and memory safety features help prevent excessive memory consumption, ensuring that simulations run efficiently without memory leaks (Klabnik & Nichols, 2019).

4 Conclusion

4.1 Key Takeaways

This "paper" explored the implementation of Monte Carlo and Heston model simulations for stock price dynamics and financial risk assessment. The Monte Carlo simulation provided a simple yet effective method for estimating future stock prices based on Geometric Brownian Motion (GBM). However, its assumption of constant volatility limited its ability to capture actual market behaviors. However, the Heston model introduced random volatility, allowing for a more realistic representation of financial markets by incorporating volatility clustering and mean reversion effects (Gatheral, 2006).

A key finding from our simulations was that risk assessment methods such as Value at Risk (VaR) produce different estimates depending on the volatility model used. The Monte Carlo simulation, due to its constant volatility assumption, may underestimate extreme market movements, while the Heston model captures a wider range of outcomes, making it more suitable for pricing options and managing risk in uncertain conditions (Jorion, 2007).

4.2 Limitations

While both models provided useful insights into price behavior, a few key limitations were found. First, the accuracy of Monte Carlo methods depends on the number of simulations performed. A higher number of simulations improves accuracy but needs more computational power. Second, the Heston model requires additional parameters, such as mean reversion speed and volatility of volatility, which need to be estimated carefully. In my program, these were set to constants and do not accurately reflect a real stock's behaviors. Incorrect parameter selection can lead to misleading results (Hull, 2017).

Another limitation is the assumption of normality in the Monte Carlo simulation, which may not fully capture extreme events like financial crashes. Alternative models, such as jump-diffusion processes, could be explored to better account for sudden market shocks (Gatheral, 2006) (I hope to explore more models in the future).

4.3 Future Improvements

Future work can focus on optimizing the implementation of these simulations to improve efficiency. Techniques such as variance reduction methods can significantly enhance performance (Jäckel, 2002). Additionally, incorporating machine learning techniques into financial modeling could provide more adaptive and data-driven approaches to risk assessment (Everybody LOVES AI).

Another area of improvement is model calibration (which I exempted myself from doing). The accuracy of the Heston model heavily depends on estimating its parameters from historical data (which I did not do). Implementing advanced calibration techniques, such as maximum likelihood estimation (MLE) or Bayesian inference, could improve model reliability (Gatheral, 2006) (another improvement I hope to explore!).

Finally, expanding the models to include macroeconomic factors, such as interest rates and economic indicators, could provide a more holistic approach to financial forecasting. By integrating economic data with stochastic models, future research could enhance predictive capabilities and risk management strategies (Jorion, 2007) (I'm not quite sure how I would implement this, but I want to).

4.4 Final Thoughts

Financial modeling plays a crucial role in risk assessment and decision-making. Monte Carlo and Heston simulations provide valuable tools for understanding market behavior and estimating potential losses. While no model is perfect, continuously improving these methodologies through computational advancements and refined parameter estimation will lead to more accurate and robust risk assessment frameworks.

I would like to thank you for reading this pretty lackluster explanation of my code and the math behind it!

References

[1] Jean-Philippe Bouchaud and Marc Potters. *Theory of financial risk and derivative pricing: from statistical physics to risk management*. eng. 2. ed., reprinted, 1. paperback ed. Cambridge: Cambridge Univ. Press, 2011. ISBN: 9780521819169 9780521741866.

[2] Phelim Boyle, Mark Broadie, and Paul Glasserman. "Monte Carlo methods for security pricing". en. In: *Journal of Economic Dynamics and Control* 21.8-9 (June 1997), pp. 1267–1321. ISSN: 01651889. DOI: `10.1016/S0165-1889(97)00028-6`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0165188997000286` (visited on 03/09/2025).

[3] *Brownian motion and the wiener process — quantstart*. URL: `https://www.quantstart.com/articles/Brownian-Motion-and-the-Wiener-Process/` (visited on 03/09/2025).

[4] Jim Gatheral, ed. *The volatility surface: a practitioner's guide*. en. 1st ed. Wiley, Jan. 2012. ISBN: 9780471792512 9781119202073. DOI: `10.1002/9781119202073`. URL: `https://onlinelibrary.wiley.com/doi/book/10.1002/9781119202073` (visited on 03/09/2025).

[5] Paul Glasserman. *Monte carlo methods in financial engineering*. Ed. by B. Rozovskii and M. Yor. Vol. 53. Stochastic Modelling and Applied Probability. New York, NY: Springer New York, 2003. ISBN: 9781441918222 9780387216171. DOI: `10.1007/978-0-387-21617-1`. URL: `http://link.springer.com/10.1007/978-0-387-21617-1` (visited on 03/09/2025).

[6] Steven L. Heston. "A closed-form solution for options with stochastic volatility with applications to bond and currency options". In: *The Review of Financial Studies* 6.2 (1993), pp. 327–343. ISSN: 0893-9454. URL: `https://www.jstor.org/stable/2962057` (visited on 03/09/2025).

[7] *Heston model*. URL: `https://algotradinglib.com/en/pedia/h/heston_model.html` (visited on 03/09/2025).

[8] Desmond J. Higham. "An algorithmic introduction to numerical simulation of stochastic differential equations". en. In: *SIAM Review* 43.3 (Jan. 2001), pp. 525–546. ISSN: 0036-1445, 1095-7200. DOI: `10.1137/S0036144500378302`. URL: `https://epubs.siam.org/doi/10.1137/S0036144500378302` (visited on 03/09/2025).

[9] Peter Jäckel. *Monte Carlo methods in finance*. eng. Repr. Wiley finance series. Chichester Weinheim: Wiley, 2010. ISBN: 9780471497417.

[10] Hull John. *Options, futures, and other derivatives*. English. 10th ed. original-date: Pearson, Jan. 2017.

[11] Steve Klabnik and Carol Nichols. *The Rust programming language*. eng. OCLC: 1162109406. San Francisco: No Starch Press, 2019. ISBN: 9781098122539.

[12] Evert Wipplinger. “Philippe jorion: value at risk – the new benchmark for managing financial risk: 3rd edition, isbn 0-07-146495-6, mcgraw–hill, 2007, 602 pages, approx. 120 chf(Hardcover)”. en. In: *Financial Markets and Portfolio Management* 21.3 (Sept. 2007), pp. 397–398. ISSN: 1934-4554, 2373-8529. DOI: `10.1007/s11408-007-0057-3`. URL: `http://link.springer.com/10.1007/s11408-007-0057-3` (visited on 03/09/2025).