ISSN 2320 - 2602
Volume 14 No. 6, June 2025
International Journal of Advances in Computer Science and Technology
Available Online at http://www.warse.org/IJACST/static/pdf/file/ijacst021462025.pdf
https://doi.org/10.30534/ijacst/2025/021462025

Monte Carlo Calculations Supporting Automatic Verification

Ile Dimitrievski[1], Stefan Trajanoski[2], Rasim Salkoski[3]
[1]Assistant Professor at Faculty of Computer Science and Engineering, University of Information Science and Technology "St. Paul the Apostle", Republic of North Macedonia, ile.dimitrievski@uist.edu.mk
[2]Master student at Faculty of Communication Networks and Security, University of Information Science and Technology "St. Paul the Apostle", Republic of North Macedonia, stefan.trajanoski@cse.uist.edu.mk
[3]Associated Professor at Faculty of Information systems Visualization Multimedia and Animation, University of Information Science and Technology "St. Paul the Apostle", Republic of North Macedonia, rasim.salkoski@uist.edu.mk

Received Date : April 24 , 2025 Accepted Date : May 30, 2025 Published Date : June 07, 2025

ABSTRACT

Today programming languages are very popular and using them for additional analysis in the process of investing our funds it could be from big plus for better and improved decisions. Different methods, different strategies as well as different analysis can bring us into different solutions and ideas, for smart and sustainable investing strategies Python programming language is sophisticated well-known software tool which provides solutions in terms of programs as part of financial analysis. Monte Carlo simulations use random sampling to solve complex problems that are too difficult for exact solutions. Distributed systems, which consist of many interconnected computers working together, can benefit greatly from Monte Carlo methods. This article explains how Monte Carlo simulations can be applied to distributed systems, highlighting their benefits, how to implement them, and some practical uses.

Key words : Investment, Monte Carlo, Finances, Python, Simulations, Synergy

1. INTRODUCTION

Monte Carlo Monte Carlo simulation is a powerful statistical technique widely used in automatic verification, particularly for analyzing and verifying systems with inherent complexity or uncertainty. Automatic verification involves systematically checking whether a system satisfies specified properties or behaves as intended. However, for many real-world systems, exhaustive analysis becomes infeasible due to the vast state spaces, probabilistic behaviors, or the presence of stochastic elements. Monte Carlo simulation addresses these challenges by employing random sampling to estimate system behaviors and performance metrics.[1][2]

This approach is especially effective in scenarios where traditional methods, such as exhaustive model checking or formal proofs, are computationally prohibitive. Monte Carlo simulation can approximate system properties by repeatedly generating random inputs or scenarios based on a defined probability distribution. By analyzing the results of these simulations, engineers and researchers can assess system reliability, detect potential failures, and evaluate performance under varying conditions.[3]

Applications of Monte Carlo simulation in automatic verification span diverse domains, including probabilistic model checking, rare-event detection, and performance analysis. For instance, it is instrumental in verifying distributed systems, autonomous vehicles, and safety-critical applications like medical devices or aerospace systems. Despite its scalability and flexibility, Monte Carlo simulation faces challenges such as balancing computational cost with accuracy and ensuring adequate coverage of critical system behaviors.[4][6]
By combining Monte Carlo techniques with other verification methods, modern systems can achieve higher levels of reliability and robustness, making it an indispensable tool in complex system verification.

2. MONTE CARLO SIMULATION FUNDAMENTALS

The Monte Carlo simulation is a statistical method that relies on random sampling to estimate numerical results for complex problems. It is particularly useful in scenarios where deterministic solutions are difficult to obtain due to uncertainty, large state spaces, or stochastic behaviors. The technique is widely applied in fields such as finance, engineering, risk assessment, and system verification. By repeatedly simulating possible outcomes based on probability distributions, Monte Carlo methods provide insights into the likely behavior of a system under varying conditions.

The process of Monte Carlo simulation generally follows four key steps. First, the problem domain is defined by identifying relevant variables and their associated probability distributions. These distributions represent the uncertainty and variability inherent in the system. Second, random samples are generated using pseudo-random number generators, ensuring a diverse range of potential scenarios. Third, each scenario is simulated, and the resulting outcomes are recorded for analysis. Finally, the results are aggregated to estimate probability distributions and derive statistical insights, such as expected values, confidence intervals, and risk assessments.

Monte Carlo methods are particularly effective in cases where traditional deterministic approaches fall short. For example, in financial modeling, Monte Carlo simulations help assess investment risks and predict stock price movements. In engineering, they support reliability analysis by estimating system failure probabilities under uncertain conditions. Additionally, Monte Carlo techniques play a crucial role in probabilistic model checking, where they approximate system properties in scenarios where exhaustive verification is computationally infeasible. The adaptability and scalability of Monte Carlo simulation make it an invaluable tool for analyzing complex, uncertain systems.

3. AUTOMATIC VERIFICATION OVERVIEW

Automatic verification is a crucial process in ensuring that complex systems behave as expected and meet predefined specifications. It involves the use of computational techniques to systematically check a system's properties, eliminating the need for manual inspection, which is often impractical for large-scale or highly intricate systems. By automating the verification process, errors can be detected early in development, reducing costs and improving system reliability. Automatic verification is widely applied in domains such as software engineering, hardware design, distributed systems, and safety-critical industries like aerospace and healthcare.

There are several key approaches to automatic verification, each with its strengths and limitations. Model checking is one of the most widely used techniques, where a formal model of the system is constructed and systematically explored to verify if it satisfies specified properties, typically expressed in temporal logic. This method is highly effective for detecting logical errors, deadlocks, or safety violations in reactive systems. Another approach is theorem proving, which uses mathematical logic to prove that a system adheres to its specifications. Although theorem proving provides rigorous correctness guarantees, it often requires significant expertise and manual intervention.

Other verification techniques include static analysis and simulation-based testing. Static analysis examines the system's structure and code without executing it, identifying potential issues such as unreachable code, memory leaks, or race conditions. On the other hand, simulation-based testing executes the system under various conditions to observe its behavior in different scenarios. While it does not provide exhaustive verification, it is a practical method for uncovering faults in complex systems.

Overall, automatic verification is essential for ensuring system correctness and reliability, particularly in environments where failures can have severe consequences. By integrating various verification techniques and leveraging scalable approaches such as Monte Carlo simulations, engineers can improve the efficiency and accuracy of system validation, making automatic verification an indispensable tool in modern computing and engineering.

4. SYNERGY BETWEEN MONTE CARLO SIMULATIONS AND AUTOMATIC VERIFICATION

The combination of Monte Carlo simulations and automatic verification creates a powerful synergy for analyzing and verifying complex systems. Automatic verification provides systematic methods for ensuring that a system adheres to its specified properties, while Monte Carlo simulations offer statistical techniques to handle uncertainty, randomness, and vast state spaces effectively. Together, they address the limitations of each approach, enabling more comprehensive and practical verification solutions.

4.1 Leveraging Monte Carlo in Verification

Monte Carlo simulations complement traditional verification methods by focusing on probabilistic and stochastic systems. For example, in probabilistic model checking, Monte Carlo simulations approximate the likelihood of certain outcomes by randomly sampling scenarios. This is particularly useful in verifying systems with infinite or intractable state spaces, where exhaustive exploration is computationally infeasible.

4.2 Rare Event Detection

Rare-event simulation is a significant area where this synergy is evident. Automatic verification techniques identify critical regions in a system, and Monte Carlo methods focus computational resources on these regions. This approach efficiently estimates the probability of rare but critical events, such as system failures in safety-critical applications.

4.3 Performance Verification

Monte Carlo simulations enhance automatic verification in evaluating performance metrics, such as latency, reliability, or throughput. By generating realistic inputs and simulating operational scenarios, Monte Carlo methods provide insights into system behavior under uncertain conditions.

4.4 Practical Benefits

Scalability: Monte Carlo reduces the computational overhead of exhaustive verification.

Flexibility: It adapts to various probabilistic models and uncertain environments.

Integration: Automatic verification guides Monte Carlo sampling by identifying relevant system properties and constraints.

4.5 Flow Chart of Monte Carlo Simulation

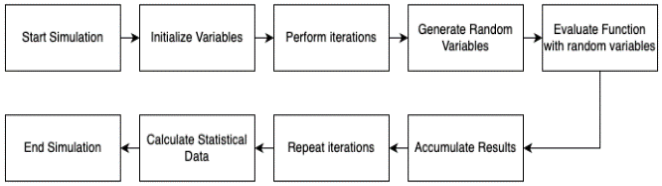

Figure 1: Flow Chart of Monte Carlo Simulation process

On Figure 1 is shown Flow Chart of Monte Carlo Simulation process. The Monte Carlo simulation process begins with the Start Simulation step, where the simulation environment is initialized as shown on Figure 1. This step signifies the beginning of the computational process, ensuring that all necessary configurations are set up correctly. Following this, in the Initialize Variables step, all required variables, parameters, and probability distributions are defined. These variables typically include input parameters, boundary conditions, and probability functions that will be used in the random sampling process. Proper initialization is crucial, as it lays the foundation for generating meaningful simulation results.

Once the variables are set, the Perform Iterations step starts. Monte Carlo simulations rely on repetitive trials to estimate possible outcomes. Each iteration represents an independent experiment where different random inputs are used. Within each iteration, the Generate Random Variables step takes place, where random numbers are drawn from predefined probability distributions. These random values represent uncertain factors in the system, allowing the simulation to model real-world variability.

After generating the random variables, the simulation moves to the Evaluate Function with Random Variables step. Here, the mathematical model or function governing the system is computed using the randomly generated values. This function could represent anything from financial risk models to physical system behaviors. Once the function is evaluated, the results are stored in the Accumulate Results step, where data from multiple iterations is collected for further statistical analysis.

The Repeat Iterations step ensures that the process continues for a predefined number of trials or until a convergence criterion is met. Running multiple iterations increases the accuracy of the simulation by providing a more comprehensive statistical representation of possible outcomes. Once all iterations are completed, the simulation enters the Calculate Statistical Data phase, where results from all iterations are analyzed. Key statistical measures such as mean, variance, confidence intervals, and probability distributions are calculated to extract meaningful insights.
Finally, the process concludes with the End Simulation step. At this point, the collected statistical data is used to interpret results, make decisions, or refine models. Monte Carlo simulations provide valuable probabilistic insights, making them useful for risk assessment, decision-making, and system optimization in various fields, including finance, engineering, and scientific research.

5. CASE STUDY: FORECASTING STOCK PRICES USING MONTE CARLO SIMULATION

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

For this case study about Monte Carlo Simulation, I will use Python programming language to get the desired results in the chart. In this part, I will start with explaining the programing part or the code of the program, and after the explanation of the code we will analyze companies. Shortly, about Monte Carlo Simulation, it is technique that allows certain people to estimate the possible outcomes of an uncertain event. Monte Carlo Simulation is used many areas like finance, engineering etc.

Figure 2 shows installation of the required Python libraries. Installing required libraries for the program to be able to make calculation and presentation of the desired data is shown on Figure 2. Python offers a rich ecosystem of libraries that make it a powerful language for a wide range of applications. Libraries like NumPy and SciPy provide robust tools for numerical and scientific computing, enabling efficient handling of large datasets and complex mathematical operations. Pandas is essential for data manipulation and analysis, offering data structures and functions designed to make data cleaning and preparation straightforward. For machine learning, libraries such as TensorFlow, Keras, and Scikit-learn are widely used, providing comprehensive tools for building and training models. Matplotlib and Seaborn are popular for data visualization, allowing users to create detailed and informative plots. Additionally, libraries like Flask and Django simplify web development, while BeautifulSoup and Scrappy are invaluable for web scraping tasks. These libraries, among many others, enhance Python’s versatility and make it a preferred choice for developers in various fields.

```
import numpy as np
import pandas as pd
from pandas_datareader import data as wb
import matplotlib.pyplot as plt
import yfinance as yf
from scipy.stats import norm
```

Figure 2: Installing the required Python libraries

```
def forecast(stock):
    ticker = stock
    data = pd.DataFrame()
    data = yf.download(ticker,'2016-01-01', today)['Adj Close']
```

Figure 3: Forecast function for getting data for the stock

```
    log_returns = np.log(1 + data.pct_change())
    u = log_returns.mean()
    var = log_returns.var()


    drift = u - (0.5 * var)
    stdev = log_returns.std()



    np.array(drift)



    t_intervals = 250
    iterations = 500

    daily_returns = np.exp(drift+ stdev * norm.ppf(\
        np.random.rand(t_intervals, iterations)))
    s0 = data.iloc[-1]
```

Figure 4: Formulas used for calculating of the logarithmic returns in the simulation

```
    price_list = np.zeros_like(daily_returns)
    price_list[0] = s0

    for t in range(1, t_intervals):
        price_list[t] = price_list[t - 1] * daily_returns[t]

    price_DF = pd.DataFrame.from_dict(price_list)
    price_DF = price_DF.T

    price_forecast = {}

    for a in range(1, t_intervals):
        price_forecast[a] = {}
        price_forecast[a] = price_DF[a].mean()
```

Figure 5: Creating the price list and the for loop for the intervals

```
    plt.figure(figsize=(10, 6))
    plt.xlabel('Days from today')
    plt.ylabel('Stock Price')
    plt.plot(price_list)
    plt.show()
    plt.savefig('myfig.jpeg')
```

Figure 6: Function for drawing the plots

Figure 3 shows the forecast function and the process of getting data for the stock. Firstly, on the Figure 3 it is shown defining of the function, after the defining of the function in the Figure 3 we are getting the data about the stock till today. With the help of the yfiance library we are getting the data from yahoo finance and we are taking Adj Close price.

Figure 4 shows mathematical chunk of code with formulas that has purpose of calculating logarithmic returns.
Logarithmic return is a way or method for calculating return, in this case logarithmic return is used to calculate the return on a daily basis.[5][7]. Np.log short for numpy.log is part of the numpy library and is a mathematical function that helps user to calculate natural logarithm. In the Figure 3 is simply the natural log of 1 plus the arithmetic return or percent change. Second line is for calculating the average of the logarithmic return. For the formula I used the letter U because is the most similar to the Greek letter micro which is symbol for mean in statistics. Finally, with. Var () function we are calculating the variance of the stock prices. Variance essentially is the measure of how far the stock price is spread out from the mean.

Figure 5 shows the process of creating the price list and the for loop for the intervals.[7]. The price list is created also with the help of the numpy library. Price list basically is array of zeros, firstly it will be filled with zeros in the array but after it will be swapped with the daily returns with actual values. Next what we are creating is for loop to go through the intervals, and the formula in the for loop is for the price list to be filled with the values of calculations. If we simplify this, it from programming and mathematical aspect this part of the code it will go to the day which have at the moment and it will go in the next day price t and it will calculate.

Everything till now it was the calculation for Monte Carlo simulation, to be understandable for the person who will read the data on the computer we must to write the code to show the data to be readable for everyone. Mathplotlib library is the library for presentation the data on the graph, firstly with this part of the code we are drawing the plot with size of 10-inch width and 6-inch height. Next function it will be the name on the X – axis accompanying to that next function it will be name on Y – axis. The data in the plot will be presented with the data frame of price list which we created previously. And this plot

to be visible for the user we are using plt. show and pl. savfig to save plot as image. Figure 6 shows the function for drawing the plots, and it will be an output of the image that will show how does this plot will look when it generated.

5.1 Monte Carlo Simulation scenario for Apple INC (AAPL)

Monte Carlo simulation was carried about company Apple INC referred as (AAPL) with the data given in the Table 1.

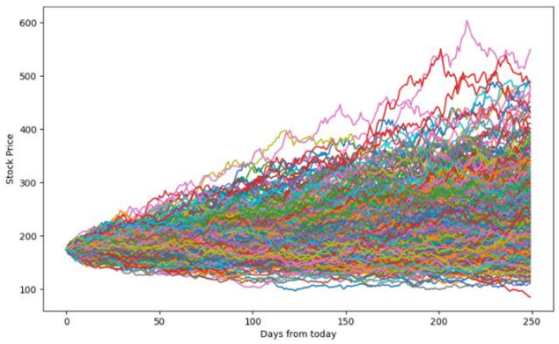

Figure 7: Monte Carlo simulation results for Apple – AAPL

Apple Inc. is a multinational corporation that designs, manufactures, and sells smartphones, laptops, tablets, wearables, and accessories. It also provides a variety of connected services.

Table 1: General information about Apple INC (AAPL)

Sector(s)	Industry	Full Time Employees
Technology	Consumer Electronics	154,000

Figure 7 shows the Monte Carlo simulation graph result of the simulation for the company Apple Inc with ticker name AAPL as commonly presented in the social media. From the chart we can read that it is about time distance of 250 days and 500 possible scenarios for the stock to go up or down. Starting point of the price of the stock will be 175.42 $, at the end of the last day the average price for the stock will be 241.65 $. After some excel calculations the average percentage growth of the stock will be 27%. Also, we can see some extreme scenarios which impact the price to go up to 600 dollars and to fall down below 100 dollars.

6. CONCLUSION

Monte Carlo simulation and automatic verification together form a powerful framework for analyzing and verifying complex systems. While automatic verification provides systematic techniques for ensuring that systems meet their specifications, Monte Carlo simulations address challenges of uncertainty and scalability by employing statistical methods to estimate behaviors and probabilities. This synergy is particularly effective for systems with large or infinite state spaces, probabilistic behaviors, or rare-event scenarios.

Monte Carlo simulations enhance the efficiency and practicality of automatic verification by reducing computational costs, focusing on critical system behaviors, and providing probabilistic guarantees when exhaustive methods are infeasible. Meanwhile, automatic verification techniques guide the simulation process, ensuring that the sampling is directed toward meaningful areas of the system's state space.

Despite the challenges of balancing accuracy, computational cost, and coverage, the integration of these methods has proven indispensable in domains like safety-critical applications, performance analysis, and probabilistic model checking. By combining the strengths of both approaches, engineers and researchers can achieve a deeper understanding of system behavior, verify properties with greater confidence, and ensure reliability in increasingly complex systems.

As systems continue to grow in complexity and uncertainty, the partnership between Monte Carlo simulation and automatic verification will remain a cornerstone of robust system analysis, paving the way for advancements in automation, safety, and efficiency across industries.

REFERENCES

1. S. Köksal. **"Python for Finance #1: Monte Carlo Simulation"**, August 15, 2023, Medium. https://0xsemihkoksal.medium.com/how-to-implement-monte-carlo-simulation-with-python-2509e79104c
2. NASA. JSTAR Monte Carlo Simulation, August 19, 2024, NASA. https://www.nasa.gov/monte-carlo-simulation/
3. J. Stimpel. **"Power of Monte Carlo Simulations in Finance"**, June 10, 2024, PyQuant News. https://www.pyquantnews.com/free-python-resources/power-of-monte-carlo-simulations-in-finance
4. D. Buckley. **"How to Make a Monte Carlo Simulation in Python (Finance)"**, October 5, 2023, DayTrading.com. https://www.daytrading.com/monte-carlo-simulation-python
5. M. Ganchev. **"How to Analyze Financial Data with Monte Carlo Simulation?"**, January 20, 2022, 365 Data Science. https://365datascience.com/tutorials/python-tutorials/monte-carlo-simulation/
6. Z. Oakes. **"Monte Carlo Simulation: Random Sampling, Trading and Python"**, August 30, 2023, QuantInsti. https://blog.quantinsti.com/monte-carlo-simulation/
7. E. Melul. "**finance_portfolio/MontecarlAutomated.ipynb",** 2023, September 12, 2023, GitHub. https://github.com/eliasmelul/finance_portfolio/blob/master/Montecarlo%20Automated.ipynb