Proceedings of CONF-APMM 2026 Symposium: Applied Mathematics and Quantitative Modeling
DOI: 10.54254/2753-8818/2026.PU36482

Monte Carlo Simulation for Estimating π: Algorithm Principle and Error Analysis

Jinhao Guo

New College, University of Toronto, Toronto, Canada
caesar.guo@mail.utoronto.ca

Abstract. Monte Carlo simulation is a stochastic numerical method that approximates solutions to mathematical problems via repeated random sampling. Owing to its conceptual simplicity and broad applicability, it has become a foundational computational tool across physics, finance, engineering, and data science. This paper examines the algorithmic principle of Monte Carlo π estimation and systematically analyzes factors governing simulation accuracy. A Python-based numerical simulation is implemented: uniformly distributed random points are generated within a unit square, and the value of π is estimated by counting points falling inside the inscribed quarter circle. Experiments with varying sample sizes are conducted to assess their impacts on estimation accuracy and convergence behavior, and the influence of pseudo-random number generation on simulation outcomes is also discussed. The results confirm that estimation stability improves with increasing sample size, and the simulation error decays approximately at a rate proportional to the inverse square root of the sample size. This study elucidates the mathematical fundamentals underlying Monte Carlo simulation and offers an intuitive case for understanding the properties, strengths, and limitations of stochastic numerical methods. The findings deepen understanding of the relationship between sample size and simulation accuracy, and provide theoretical and practical support for extending Monte Carlo methods to more complex numerical simulation and uncertainty analysis tasks.

Keywords: Monte Carlo simulation, π estimation, Random sampling, Simulation error, Python

1. Introduction

Monte Carlo simulation is a class of stochastic numerical methods that approximate solutions to mathematical problems through repeated random sampling. Since its formal introduction by Metropolis and Ulam in 1949, it has emerged as one of the most widely adopted computational techniques in scientific research, valued for its conceptual simplicity, structural flexibility, and applicability to complex systems [1]. Monte Carlo methods have been extensively deployed in physics, engineering, finance, and statistics—domains where analytical solutions are often intractable or unattainable [2, 3]. Recent research has further enhanced the efficiency and accuracy of Monte Carlo simulation via advanced sampling strategies and computational optimization techniques [4, 5]. Among its numerous canonical applications, π estimation based on geometric

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2753-8818/2026.PU36482

probability is regarded as one of the most classic and intuitive examples, offering a straightforward demonstration of the core principles of stochastic simulation [6].

This paper focuses on the Monte Carlo algorithm for π estimation and investigates the key factors determining simulation accuracy. A Python-based simulation is constructed to generate uniformly distributed random points within a unit square, and multiple sample sizes are compared to evaluate their effects on estimation accuracy and convergence patterns. In addition, the impacts of pseudo-random number generation and repeated sampling on simulation outcomes are briefly discussed [3, 4]. This study aims to verify the quantitative relationship between sample size and estimation error, and to characterize the convergence behavior of the Monte Carlo estimator as sample size grows. The findings help illustrate the mathematical fundamentals and convergence properties of Monte Carlo simulation, while providing an accessible reference for learners to understand stochastic simulation methods and their applications in scientific computing [2, 5].

2. Mathematical principles of Monte Carlo π estimation

This section introduces the theoretical foundation of the Monte Carlo algorithm for estimating π. First, the mathematical model based on geometric probability is established, and the corresponding Monte Carlo estimator is derived. Subsequently, the implementation procedure adopted in this study is presented, including the simulation workflow and computational framework. These theoretical and computational foundations provide the basis for the experimental design and performance evaluation described in the following sections.

2.1. Basic mathematical theory of Monte Carlo simulation

Monte Carlo simulation is a stochastic numerical method that approximates the solution of mathematical problems through repeated random sampling. Unlike deterministic numerical algorithms, Monte Carlo methods estimate unknown quantities by exploiting probabilistic principles, making them particularly suitable for problems whose analytical solutions are difficult or impossible to obtain [1, 2]. In this study, the estimation of π is formulated as a geometric probability problem based on the area ratio between a quarter circle and its circumscribed unit square.

Consider a unit square containing an inscribed quarter circle of radius one. Since the areas of the square and the quarter circle are 1 and π/4, respectively, a point uniformly sampled from the square falls inside the quarter circle with probability π/4. Let (Xi, Yi), i = 1, . . ., N, denote independently and uniformly distributed random points in the square. Whether a sampled point lies inside the quarter circle is determined by the following indicator function:

$$f(x) = \begin{cases} 1, & X_i^2 + Y_i^2 \le 1 \\ 0, & X_i^2 + Y_i^2 > 1 \end{cases} \tag{1}$$

The total number of points located inside the quarter circle is then expressed as

$$N_{inside} = \sum_{i=1}^{N} I_i \tag{2}$$

According to the geometric probability relationship,

$$\frac{N_{inside}}{N} \approx \frac{\pi}{4} \tag{3}$$

DOI: 10.54254/2753-8818/2026.PU36482

which leads to the Monte Carlo estimator

$$\hat{\pi} = 4\frac{N_{inside}}{N} \tag{4}$$

The estimator $\hat{\pi}$ approximates the true value of π rather than its exact value. According to the Law of Large Numbers, the sample proportion converges almost surely to the corresponding probability as the sample size increases [3]. Consequently,

$$\lim_{N\to\infty} \hat{\pi} = \pi \tag{5}$$

Furthermore, the convergence rate of the classical Monte Carlo estimator is proportional to N−1/2, indicating that a fourfold increase in sample size is generally required to reduce the standard error by approximately one half [7, 8].

$$O(N^{\frac{-1}{2}}) \tag{6}$$

indicating that the estimation error decreases proportionally to the inverse square root of the sample size [4]. Although this convergence rate is slower than that of many deterministic numerical methods, Monte Carlo simulation possesses excellent scalability and is largely independent of problem dimensionality, which explains its widespread application in scientific computing, uncertainty quantification, financial engineering, and statistical inference [2, 5]. Therefore, the mathematical model established above provides the theoretical foundation for the numerical experiments conducted in this study.

2.2. Python implementation process of point-casting simulation

Based on the mathematical model presented above, a numerical simulation was implemented in Python to estimate the value of π using Monte Carlo sampling. The NumPy library was employed to generate uniformly distributed pseudo-random numbers, while Matplotlib was used to visualize the simulation results. The reliability of Monte Carlo simulation partly depends on the statistical quality of the pseudo-random number generator. In this study, the MT19937 generator was adopted because of its long period and favorable equidistributional properties [9]. A fixed seed was specified to ensure the reproducibility of the numerical experiments.

In each simulation, pairs of independent random numbers were uniformly generated over [0, 1]. Points satisfying $X_i^2 + Y_i^2 \leq 1$ were classified as interior points, after which π was estimated using Eq. (2). Vectorized array operations were used to process the sampled points efficiently. The detailed sample-size settings, repetition scheme, and evaluation criteria are presented separately in Section 3.

To investigate the influence of sample size on estimation accuracy, five different sample sizes were selected, namely $N = 10^2$, 10^3, 10^4, 10^5, and 10^6. Each experiment was independently repeated thirty times in order to reduce the influence of random fluctuations and to obtain statistically reliable results. The average estimated value, absolute error, relative error, and computational time were recorded for each sample size. This experimental design enables a comprehensive evaluation of both the convergence behaviour and computational efficiency of the Monte Carlo algorithm.

To quantitatively assess the simulation performance, three evaluation metrics were adopted. The relative error is defined as:

DOI: 10.54254/2753-8818/2026.PU36482

$$\mathrm{RE} = \frac{\left|\hat{\pi} - \pi\right|}{\pi} \times 100\% \tag{7}$$

allowing comparisons among different experimental settings. In addition, the computational time required for each simulation was recorded to analyse the trade-off between estimation accuracy and computational cost. These performance indicators provide an objective basis for evaluating the effectiveness of the proposed simulation framework and facilitate the subsequent analysis presented in Section 3.

3. Simulation experiment design

This section describes the experimental design adopted to evaluate the performance of the Monte Carlo algorithm for estimating the value of π. To ensure the reliability and reproducibility of the experimental results, the computational environment, experimental parameters, and evaluation criteria were carefully controlled. During the simulation, only the sample size was treated as the independent variable, while all other experimental conditions remained unchanged. Such a single-variable experimental design makes it possible to investigate the influence of sample size on estimation accuracy without interference from other factors [4].

3.1. Experimental environment

All numerical simulations were conducted using Python 3.12 on a personal computer equipped with an Intel Core i7 processor and 16 GB of RAM under the Windows 11 operating system. The NumPy library was employed to generate pseudo-random numbers with a uniform distribution over the interval [0,1], while Matplotlib was used to visualize the experimental results. The simulation program was executed under identical software and hardware conditions throughout all experiments to ensure the consistency of the computational environment.

To minimise the influence of irrelevant variables, the radius of the quarter circle and the side length of the square were fixed at one throughout the experiments. The same random number generation strategy and computational procedure were adopted for every experimental group, ensuring that the sample size remained the only variable affecting the estimation results.

3.2. Experimental scheme

The objective of this experiment is to investigate how the sample size influences the estimation accuracy and convergence behaviour of the Monte Carlo algorithm. To examine the effect of sample size on estimation accuracy, five experimental groups with different sample sizes were established. Table 1 summarizes the sample-size settings for each experimental group.

Table 1. Experimental group settings by sample size

Group Classification	Sample Size (N)
Group 1	100
Group 2	1,000
Group 3	10,000
Group 4	100,000

DOI: 10.54254/2753-8818/2026.PU36482

Table 1. (continued)

Group 5	1,000,000

For each experimental group, N random points were independently generated within the unit square. The number of points falling inside the quarter circle was counted, and the estimated value of π was calculated according to:

$$\hat{\pi} = 4\frac{N_{inside}}{N} \tag{8}$$

To reduce the influence of random fluctuations, each experiment was independently repeated 30 times using different random seeds. The average estimated value of π was subsequently calculated for each sample size. This repeated experimental design improves the statistical reliability of the simulation results and provides a more objective basis for subsequent performance evaluation.

3.3. Evaluation criteria

To quantitatively evaluate the performance of the Monte Carlo algorithm, three evaluation indicators were adopted in this study.

The first indicator is the **absolute error**, which measures the absolute deviation between the estimated value and the true value of π,

$$AE = \left|\hat{\pi} - \pi\right| \tag{9}$$

The second indicator is the **relative error**, defined as

$$RE = \frac{\left|\hat{\pi}-\pi\right|}{\pi} \times 100\% \tag{10}$$

This indicator enables comparisons among experiments with different estimation results by expressing the error as a percentage of the true value.

Finally, the **computational time** required for each simulation was recorded to evaluate the computational efficiency of the algorithm. Since Monte Carlo simulation relies on repeated random sampling, increasing the sample size is expected to improve estimation accuracy while simultaneously increasing computational cost. Therefore, both estimation accuracy and computational efficiency should be considered when evaluating the performance of the algorithm.

The experimental results obtained according to the above experimental scheme will be presented and analysed in the following section.

4. Experimental results and error analysis

This section presents the numerical results obtained according to the experimental scheme described in Section 3. The estimated value of π, absolute error, relative error, standard deviation, and computational time under different sample sizes are compared. The convergence behaviour and error characteristics of the Monte Carlo estimator are then discussed based on the experimental data.

DOI: 10.54254/2753-8818/2026.PU36482

4.1. Experimental data

Each sample-size setting was independently simulated 30 times, and the mean value of the 30 estimates was used as the final result. Table 2 summarizes the estimated values of π, absolute errors, relative errors, standard deviations, and mean runtimes obtained under different sample sizes.

Table 2. Monte Carlo simulation results under different sample sizes

Sample Size (N)	Mean Estimated π	Absolute Error	Relative Error (%)	Standard Deviation	Mean Runtime (s)
100	3.10533333	0.03625932	1.154155	0.18148324	0.000664
1,000	3.14160000	0.00000735	0.000234	0.04515163	0.000040
10,000	3.14032000	0.00127265	0.040510	0.01923303	0.000184
100,000	3.14157333	0.00001932	0.000615	0.00586417	0.001714
1,000,000	3.14150640	0.00008625	0.002746	0.00193493	0.073972

As shown in Table 2, the estimated values generally became more stable as the sample size increased. When N=100, the mean estimated value was 3.10533333, with an absolute error of 0.03625932 and a relative error of approximately 1.154155%. This result indicates that a small number of random points cannot sufficiently represent the area ratio between the quarter circle and the unit square.

When the sample size increased to N=1,000, the estimated value became 3.14160000, which was extremely close to the true value of π. However, the absolute error increased slightly when N=10,000. A similar fluctuation occurred at N=1,000,000, where the absolute error was greater than that obtained at N=100,000. Therefore, the error did not decrease monotonically as the sample size increased.

Nevertheless, the standard deviation showed a clear decreasing trend. It declined from 0.18148324 at N=100 to 0.00193493 at N=1,000,000. This result demonstrates that the estimates obtained from repeated simulations became increasingly concentrated around the true value of π. Consequently, the decrease in standard deviation provides stronger evidence of convergence than the absolute error of an individual experimental group.

4.2. Influence of sample size on estimation error

The variation in estimation error can be explained by the stochastic nature of the Monte Carlo method. The estimator used in this study is expressed as

$$\hat{\pi} = \frac{4}{N}\sum_{i=1}^{N} I_i \tag{11}$$

where I_i equals 1 when the i -th random point lies inside the quarter circle and 0 otherwise. Since the probability that a point lies inside the quarter circle is $p = \pi/4$, the variance of the estimator is

$$\mathrm{Var}(\hat{\pi}) = \frac{4\pi-\pi^2}{N} \tag{12}$$

Therefore, the standard deviation of the estimator is proportional to:

DOI: 10.54254/2753-8818/2026.PU36482

$$N^{-\frac{1}{2}} \tag{13}$$

This theoretical relationship indicates that increasing the sample size reduces the overall statistical fluctuation of the simulation. To reduce the standard error by one half, the number of random samples must be increased by approximately four times [2, 4].

The experimental results are generally consistent with this theoretical convergence rate. Although the absolute error did not decrease continuously, the standard deviation decreased steadily with increasing sample size. The unusually small error obtained at $N = 1{,}000$ was caused by favourable random fluctuation and does not indicate that $N = 1{,}000$ is more reliable than larger sample sizes.

The law of large numbers guarantees that the estimated value approaches π as N tends to infinity, but it does not require every finite-sample estimate to be more accurate than the preceding estimate [3]. Therefore, the convergence of the Monte Carlo method should be evaluated through repeated experiments and statistical indicators rather than through a single estimation result.

4.3. Computational efficiency and method limitations

The computational cost of the Monte Carlo algorithm generally increased with the sample size. For each random point, the program generated two coordinates and determined whether the point satisfied

$$x_i^2 + y_i^2 \leq 1 \tag{14}$$

The computational complexity of the algorithm is therefore approximately

$$O(N) \tag{15}$$

When $N = 1{,}000{,}000$, the mean runtime reached approximately 0.073972 seconds. This result indicates that improving estimation stability requires additional computational resources.

The runtime values obtained for small sample sizes did not increase strictly in proportion to N. For example, the measured runtime for $N = 100$ was greater than that for $N = 1{,}000$. This difference may be related to timer precision, program initialisation, memory allocation, operating-system scheduling, and other fixed computational overheads. Therefore, the overall trend is more meaningful than individual runtime differences at very small sample sizes.

Although the classical Monte Carlo method is simple and flexible, its convergence rate of $O(N^{-1/2})$ is relatively slow. A substantial increase in sample size produces only a limited improvement in accuracy. In addition, the simulation result is affected by pseudo-random number generation and may vary across random seeds. Future improvements may include quasi-Monte Carlo methods, which replace purely random samples with low-discrepancy sequences to achieve a more uniform distribution of sampling points [10-12]. These approaches may improve convergence performance and reduce estimation error under the same sample size.

Overall, the results demonstrate that increasing the sample size improves the stability and reliability of the Monte Carlo estimator. However, the selection of sample size should consider both estimation precision and computational efficiency.

DOI: 10.54254/2753-8818/2026.PU36482

5. Conclusion

This study confirms that sample size plays a significant role in the stability and reliability of Monte Carlo estimation of π. The experimental results show that increasing the number of random samples generally reduces the fluctuation of the estimated values and improves their overall convergence toward the true value of π. In particular, the standard deviation decreased from 0.18148324 at N=100 to 0.00193493 at N=1,000,000, indicating a substantial improvement in estimation stability. Meanwhile, the absolute error did not decrease monotonically with increasing sample size, demonstrating that individual Monte Carlo estimates remain affected by random fluctuations. Therefore, the reliability of Monte Carlo simulation should be evaluated through repeated experiments and statistical measures rather than a single estimation result. The observed convergence behavior is consistent with the theoretical O(N-1/2) convergence rate, while the increase in computational time also demonstrates the trade-off between estimation precision and computational efficiency.

Several limitations remain in this study. The analysis is restricted to a low-dimensional geometric estimation problem, and only the classical Monte Carlo random sampling method is considered. Therefore, the conclusions regarding computational efficiency and convergence should not be directly generalized to more complex or high-dimensional applications.

Future research could compare classical Monte Carlo simulation with quasi-Monte Carlo, stratified sampling, and other variance-reduction techniques. Further investigation of different sampling strategies and larger-scale computational problems may provide a more comprehensive understanding of how sampling methods influence convergence speed, estimation accuracy, and computational efficiency.

References

[1] Metropolis, N., & Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association, 44*(247), 335–341.
[2] Hammersley, J. M., & Handscomb, D. C. (1964). *Monte Carlo methods*. Methuen.
[3] Fishman, G. S. (1996). *Monte Carlo: Concepts, algorithms, and applications*. Springer.
[4] Robert, C. P., & Casella, G. (2004). *Monte Carlo statistical methods* (2nd ed.). Springer.
[5] Owen, A. B. (2013). *Monte Carlo theory, methods and examples*. Stanford University.
[6] Kalos, M. H., & Whitlock, P. A. (2008). *Monte Carlo methods* (2nd ed.). Wiley.
[7] Kroese, D. P., Taimre, T., & Botev, Z. I. (2011). *Handbook of Monte Carlo methods*. Wiley.
[8] Rubinstein, R. Y., & Kroese, D. P. (2016). *Simulation and the Monte Carlo method* (3rd ed.). Wiley.
[9] Matsumoto, M., & Nishimura, T. (1998). Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation, 8*(1), 3–30.
[10] Niederreiter, H. (1992). *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics.
[11] Caflisch, R. E. (1998). Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica, 7*, 1–49.
[12] Lemieux, C. (2009). *Monte Carlo and quasi-Monte Carlo sampling*. Springer.