Study of Monte Carlo Simulation: Principles, Methods, and Applications

Jiajie Li

Loomis Chaffee High School, Windsor, Connecticut, 06095, United States

Matthew_Li@Loomis.org

Abstract. Monte Carlo Simulation (MCS) is a powerful computational technique used to model complex stochastic systems, enabling the evaluation of probabilities and statistical outcomes through random sampling. Originating from the field of physics, MCS has since become a versatile tool in various disciplines, including finance, engineering, and risk management. This paper explores the historical development, core principles, and mathematical foundations of MCS, as well as its practical applications. Additionally, the paper highlights the importance of programming in implementing Monte Carlo methods, particularly through Python's random package. Three specific examples demonstrate the simulation process, allowing for the evaluation of the effectiveness and efficiency of random processes in generating desired results. The study also presents an overview of MCS's applicability in real-world scenarios, offering insights into its advantages and limitations. Through this exploration, the paper aims to provide a comprehensive understanding of the Monte Carlo simulation and its relevance in modern computational modeling.

Keywords: Monte Carlo Simulation, Probability, Statistics, Python, Modeling.

1. Introduction

Monte Carlo simulations (MCS) belong to a class of computer algorithms. These algorithms are employed to simulate and model the probability of different outcomes that involve making predictions about some stochastic processes in which some random component is present that cannot be determined easily using an analytical approach. Such cases often arise due to the complexities of random variables, which render conventional deterministic approaches insufficient and challenging to implement; other times, a population parameter, such as the expected value μ, is desired but cannot be found directly, so a sample statistic with an estimator $\bar{x}$ is obtained to use for inferences for μ. In general, MCS refers to a virtual simulation in which repeated random samples are generated by means of an algorithm from a certain probability distribution, and then analysis of the results is performed for the use of statistical inferences.

The word “Monte Carlo” refers to a city in Monaco situated on a prominent escarpment located on the Mediterranean Sea at the base of the Maritime Alps. Famously, the Casino de Monte-Carlo is well-known for gambling and entertainment, as well as the 1913 gambler’s fallacy, which closely ties to MCS [1]. The Monte Carlo Method is thus appropriately named after the casino by Nicholas Constantine Metropolis, reflecting the foundational idea of MCS on randomness, like that in a roulette game. In addition to Metropolis’ suggestion, in light of World War II, the nuclear physicists working on the Manhattan Project – namely, John von Neumann and Stanisław Ulam – invented and used Monte Carlo simulation as an essential tool for gaining necessary data for developing the weapon.

The intention behind any motivation for using MCS involves modeling a real-world system that cannot be analyzed otherwise using a traditional deterministic or analytical approach; the Monte Carlo simulation starts off by building a deterministic model that closely resembles the real scenario. Unlike solving equations in closed forms, MCS provides a statistical approach to estimating a value that typically resembles closely to the true value, due to the fact that the repeated samplings or large sample data obtained all follow the Law of Large Numbers and other methods of statistical inference [2].

MCS typically involves three main steps: sampling, estimation, and data analysis/inferences. Sampling involves taking repeated, independent random samples from a certain population with a

probability distribution $f(x)$. There, using the sample statistics, MCS provides an estimation of the numerical value for the random variable of interest, which can then be used to infer information about the system that is being modeled by analyzing the sample data.

Because of these various characteristics, Monte Carlo simulation is incorporated in a variety of fields because of its versatility, flexibility, and diversity. These include but are not limited to physical sciences, finance, engineering, risk analysis, statistics, and many more. As one can see later, Monte Carlo simulation can be used in problems as simple as estimating the value of π, or approximating a certain characteristic about a complex system – for instance, computing multi-dimensional integrals, computational physics, or tracing light rays in a closed box using MCMC [3]. MCS reduces the difficulty of modeling and simulating by using algorithms to speed up the process by millions of times. With the advancement in the computer industry and the increasing computing speed in the 21st century, MCS will become even faster and more versatile in the applicable areas, allowing researchers to model almost any feature of interest. In addition, the simulation's rooting in randomness determines the power of this tool when one desires to use sample statistics to infer about a larger population.

The structure of the paper is as follows. This paper aims to explore the core mathematical principles and computational strategies underlying Monte Carlo simulation. By illustrating three case studies, the paper demonstrates the practical efficiency and versatility of the method. It highlights how MCS can be incorporated into both theoretical stochastic modeling as well as real-world applications across diverse disciplines.

2. Definitions and Methods

2.1. I.I.D. Random Variable

It is essential to define an independent, identically distributed variable (i.i.d) because it is majorly used in probability theories, such as LLN and CLT. Assuming that a variable is i.i.d. simplifies situations, makes models easier to construct and calculations easier to perform, and provides a solid foundation for many other applications in Monte Carlo simulation, such as random sampling and Markov Chain Monte Carlo (MCMC).

It can be defined as such. Consider n random variable $X_1, X_2, \dots, X_n$, each with domain $x_n \in \mathbb{R}$, their respective cumulative distribution function is given by

$$F_{X_n}(x) = \mathbb{P}(X_n \leq x). \tag{1}$$

Their joint distribution function is defined by

$$F_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots, x_n) = \mathbb{P}(X_1 \leq x_1 \wedge X_2 \leq x_2 \wedge \dots \wedge X_n \leq x_n). \tag{2}$$

The random variables are said to be identically distributed if and only if

$$F_{X_1}(x) = F_{X_n}(x), \quad \forall x_n \in \mathbb{R}, \tag{3}$$

and independent if

$$F_{X_1,X_2,\dots,X_n}(x_1, x_2, \dots, x_n) = \prod_{k=1}^{n} F_{X_k}(x_k), \quad \forall x_1, x_2, \dots, x_n \in \mathbb{R}. \tag{4}$$

2.2. Law of Large Numbers

The law of large numbers (LLN) is as follows: Mathematically, let $X_1, X_2, \dots, X_n$ be a sequence of i.i.d. random variables with expected value $\mathbb{E}[X_i] = \mu$ and finite variance. The sample mean $\overline{X}_n$ is defined as:

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{5}$$

The Law of Large Numbers asserts that as n approaches infinity, the sample mean $\overline{X}_n$ converges to the expected value μ. This convergence can be understood in two forms:

Theorem 1. Weak Law of Large Numbers: $\lim_{n\to\infty} \mathbb{P}\left(\left|\overline{X}_n - \mu\right| \geq \epsilon\right) = 0 \quad \forall \epsilon > 0$(6)

Theorem 2. Strong Law of Large Numbers: $\mathbb{P}\left(\lim_{n\to\infty}\left|\overline{X}_n - \mu\right| = 0\right) = \mathbb{P}\left(\lim_{n\to\infty}\overline{X}_n = \mu\right) = 1$

Proof for the above theorems of LLN can be found here [4, p. 246-248]. For a random variable X with series of Bernoulli trials with a probability of "success" p, it follows that the expected value $\mathbb{E}[X] = p$. This is essentially the core behind the example of random walk in dimension 1: it is a binomial distribution with probability of going in either direction – right or left – equals to $\frac{1}{2}$.

2.3. Central Limit Theorem

The central limit theorem is a remarkable foundation gained from statistics and probability theory that is nicely applied universally in the real world, because it asserts that under appropriate conditions, regardless of the distribution of a data set, the averages and sums behave predictably and follow a Gaussian distribution using a probabilistic or statistical method. These methods used for a normal distribution can be further used to study many other types of distribution; it is also essential to Monte Carlo simulation because it guarantees that the averages of the sample statistics approximate normality [5].

2.3.1 Form 1.

The CLT states that for N independent, identically distributed random variable with mean μ and variance σ^2, the normalized sample mean $\overline{X}$ also follows a normal distribution with a large enough sample size, thus

Theorem 3. $\frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \sim N(0,1)$

2.3.2 Form 2.

Moreover, for any i.i.d. random variable X with an arbitrary probability distribution with mean μ and some finite variance σ^2, the characteristic function of the variable Y_N, the sum of n random variables X, with the form:

$$Y_N = X_1 + X_2 + X_3 + \cdots + X_N, \tag{6}$$

is given by:

$$\Phi_{Y_N}(k) = [\Phi_X(k)]^N \approx \exp\left(ikn\mu - \frac{k^2 n\sigma^2}{2}\right). \tag{7}$$

As $N \to \infty$, Y_N converges in distribution to a normal random variable: Theorem 4. $Y_N \sim N(n\mu, n\sigma^2)$. Thus remains Gaussian and follows a normal distribution [6].

2.4. Some Key Distributions

2.4.1 Bernoulli distribution

Repeated independent trials are called Bernoulli trials if there are only two possible outcomes in the sample space for each trial and that the probability of obtaining each outcome remains the same respectively for all trials. It is usually thought of as a "yes/no" question, where the outcome takes the form of "success" with a probability p and "fail" with $1 - p$. Thus, a probability distribution that records outcomes of such trials is called the Bernoulli distributions and they typically have the probability mass function as listed in the below example. Consider the example of a Bernoulli distribution being a coin toss. Let $x = 1$ represents the outcome of the head and $x = 1$ tail. The probability mass function is then given by:

$$p_X(x) = \begin{cases} p, & x = 0 \\ 1 - p, & x = 1 \end{cases}. \tag{8}$$

The Bernoulli trial is the case used for the random walk in dimension 1, as illustrated in section 3. It is also a special case of the binomial distribution, which is described below.

Sometimes one is interested in the number of "successes" in a series of Bernoulli trials without considering their order. Such cases are modeled with the Binomial Distribution in that it is the discrete probability distribution of this series of Bernoulli trials. It is sometimes also regarded as A combinatorial question asking "n choose k" "successes" with probability p. To answer this, the binomial coefficient is the solution to answering how many total cases there can be to have k successes. It is known as

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{9}$$

Thus, the probability mass function of a binomial distribution is given by the equation that models the number of combinations of k successes and $n-k$ fails.

$$b(k,n,p) = \binom{n}{k} p^k q^{n-k}. \tag{10}$$

2.4.2 Uniform distribution

In Monte Carlo simulations, the uniform distribution is an essential element behind random sampling - and thus, random number generation, as discussed below. In Monte Carlo methods, large sets of uniformly distributed random samples are used to apply the Monte Carlo integration and other simulations, since it provides an unbiased coverage of the sample space.

The (continuous) uniform distribution measures the outcome of a random variable X that lies between the interval $\alpha \leq X \leq \beta$ such that for all real numbers in this interval, the probability of every outcome is equally likely. Mathematically, the variable X is said to follow a uniform distribution (i.e. $X \sim U(\alpha, \beta)$) if its pdf and cdf is given by:

$$f(X) = \begin{cases} \frac{1}{\beta-\alpha}, & \alpha \leq x \leq \beta \\ 0, & \text{otherwise} \end{cases}, \tag{11}$$

$$F(X) = \begin{cases} 0, & x < \alpha \\ \frac{x-\alpha}{\beta-\alpha}, & \alpha \leq x \leq \beta \\ 1, & x > \beta \end{cases}. \tag{12}$$

2.4.3 Gaussian distribution

The Gaussian distribution ties closely with the central limit theorem as well as random sampling in Monte Carlo simulation, because the CLT ensures that the results from Monte Carlo trials will approximate a Gaussian distribution. This process can be seen in many different cases in the real world, and it also holds practical values across different studies, such as finance, physics, and machine learning. Many final outcomes of MCS are Gaussian-distributed random variables.

The Gaussian Function is generally used to represent the probability distribution of a normally distributed random variable X (a normal distribution) with an expected value $E(X) = \mu$ and variance σ^2. The Gaussian function, or the normal density function, is of the form:

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{13}$$

Thus, for $X \sim N(0,1)$, the probability density function f(x) of X is given by

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{14}$$

2.5. RNG and Random Sampling - the Critical Mechanisms behind MCS

2.5.1 Random number generator

Arguably, one of the most important processes in any Monte Carlo simulation is the process by which random samples are drawn. This typically means selecting a random number one at a time to perform the repeated trials. To understand how this is done, the concept of a random number generator is important to know. Before algorithms were created, random outputs or numbers were generated using physical methods such as flipping a coin or spinning a roulette wheel. As later computations became more complex, these physical methods were too slow and less effective to be implemented than computer algorithms, and it was found that the generated numbers exhibited bias and dependencies [3, p. 50].

Now with efficient and fast computer algorithms that can be easily incorporated into any computer, the idea of pseudorandomness is utilized to generate a sequence of numbers that is statistically random and deterministic. A pseudorandom number generator (PRNG) is the deterministic algorithm used in this case. Typically, a successful pseudorandom number generator has to satisfy the following properties to be considered reliable and efficient: The numbers generated must be uniformly distributed over their defined and applied range; Each number in the generated sequence must be independent of each other; The PRNG must have a long period because every RNG repeats; The PRNG should be computationally efficient, deterministic, and reproducible.

It must pass a combination of statistical tests for random numbers (some examples from the test suite can be found here [7]). The simplest method is known as the linear congruential generators (LCG), first introduced in 1958 by W. E. Thomson and A. Rotenberg [8]. It is defined by the recurrent formula:

$$X_{r+1} = aX_r + c \,(\mathrm{mod}\, m), \tag{15}$$

where X_r is the current state or seed, a is the multiplier, c is the increment, and m is the modulus.

This Linear Congruential generator can generate a decent sequence if fed with an appropriate choice of parameters. By the modular arithmetic, it is obvious that the length of the generated sequence of numbers, before getting into cycles, is strictly $m - 1$.

An example is with $a = 5, c = 1, m = 100,$ and $X_r = 1$. The recurrence relation is $X_{i+1} = (5X_i + 1)$ mod 100, and the sequence is: 1, 6, 31, 56, 81, 6, ... Although this appears a decent result, the result in fact consists of a sequence of only four numbers that are not random (each value after the seed happened to be obtained by adding 25). Thus, this example resulted in a short period and poor coverage of the values with a non-random pattern.

Therefore, the quality of randomness is highly sensitive to the choice of the parameters. A poor choice can lead to non-random numbers generated because of the simplicity of this formula. Thus, an appropriate choice of the values m and a can produce an ideal period that passes most statistical tests. One such example is with $a = 7^4$, $m = 2^{31} - 1$, and $c = 0$ by Park, Stephen K. and Miller, Keith W. [9].

Modern random number generators are needed for the Monte Carlo method and others generally implement more advanced pseudorandom number generators such as the multiple recursive generators and the Mersenne Twister [8]. The latter one is interesting to consider. It is a widely used PRNG designed to surpass the limitations of older RNGs like the LCG. It has several different implementations, but the most popular one, MT19937, has a period of $2^{19937} - 1$ and produces numbers that are uniformly distributed across many dimensions [10]. Unsurprisingly, the Mersenne Twister is the default PRNG for many software programs like Excel, and Matlab, as well as many programming languages including Python, where the package random uses the Mersenne Twister as its core generator which produces 53-bit precision floats and has the period from above [11].

2.5.2 Random sampling and methods

Random Sampling refers to the process in which independent, repeated random samples are drawn from any given pdf with a known cdf, which shall be called the target distribution, $F(x)$.

The inverse-transform method, or sometimes called the inversion technique, is a method used to generate random samples from a given probability distribution function of the random variable X when its cdf is known and invertible. The core idea is to start with a random number U drawn from the standard uniform distribution on $[0,1]$ and then using the inverse cumulative distribution function of f, denoted as $F^{-1}(u)$, to map U onto X, the domain of the target distribution.

$$\mathbb{P}(X \leq x) = \mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}\big(U \leq F(x)\big) = F(x) \tag{16}$$

This in turn proves that the random variable X generated by $F^{-1}(U)$ follows the desired CDF $F(x)$

Accept/Reject Method

The accept/reject method, or rejection sampling, is a powerful and generalized sampling method used in many Monte Carlo simulations. It is, in other words, a test imposed on generated outputs that if certain conditions are met then the sample is accepted; otherwise, reject and discard it and start the random sampling again.

The procedure is as follows:

First define the target distribution $f(x)$ over the interval (α, β). Select a proposal distribution $p(x)$ such that for some scaling constant/factor C and $\forall x \in (\alpha, \beta)$, $C \times p(x) \geq f(x)$. Then, sample uniformly a point X_i from the x-axis of the proposal distribution. Draw a vertical line at $x = X_i$ until it intersects $C \times p(x)$. Next, sample uniformly along this vertical line. If the sampled value is smaller than or equal to $f(X_i)$, accept it; otherwise, discard it. Go to step (a) for n repeated iterations

Mathematically, the method is done by sample X_i from $g(x)$; drawing $u \sim U(0,1)$; accept the sample X_i if $u < \frac{f(X_i)}{Cp(X_i)}$; otherwise, reject it and start the sampling again. The samples x-values will follow the desired distribution [3, p. 60].

3. Applications

3.1. Estimating the Value of pi

Among many of the Monte Carlo simulations, a classical example involves determining an experimental value of π. Such simulation can be carried out by randomly placing many points within a square of length 1 in which a quarter of a circle is inscribed, as shown in Fig. 1.

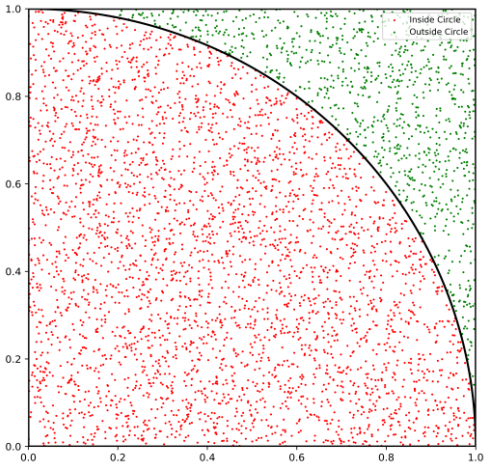

Fig. 1 Illustrated Image of the Unit Square (Photo/Picture credit: Original).

One can use the method of rejection sampling here to obtain an experimental value of π. The area ratio between the square and the quarter of the circle is

$$\frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi}{4} \tag{17}$$

Essentially, in the context of rejection samppling, the "arc" represents the density function of the proposal distribution $p(x)$ and the "square" represents the function $C \times p(x)$. Thus the estimated value of π can be obtained.

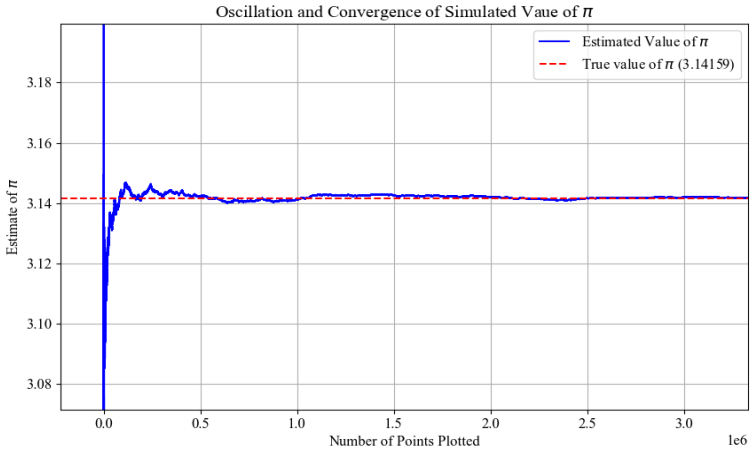

Fig. 2 Convergence of Simulated Value of π using MCS (Photo/Picture credit: Original).

The X-axis represents the total number of points plotted within the square, and the Y-axis is the value of π after finding the ratios; by accepting the points that follow below the proposal distribution, as indicated in Fig. 2, the simulated value of π oscillates in the first few hundreds of points but eventually stabilizes itself and converges.

3.2. Probability of Forming a Triangle

Imagine that there exists a stick of length 1 unit. An interesting simulation can be done to see the probability of forming a triangle after cutting at two randomly selected point on the stick. Here, MCS is used to calculate such probability illustrates that the probability converges to a fixed value as the number of trials (cuttings) goes to a very large number (Fig. 3).

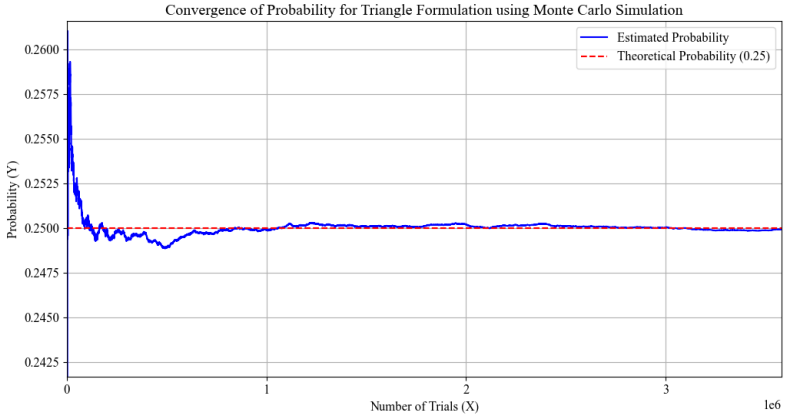

Fig. 3 Convergence of Probability for Triangle Formation Using MCS (Photo/Picture credit: Original).

The essence of this problem can be framed under the triangle inequality theorem, which states that the sum of the length of any two sides of the triangle must be greater than the length of the third side. For this problem, it can be simplified into only comparing the sum of the two shortest segments with the remaining segment.

As can be seen, the probability of a triangle being formed converges to 0.25 as the number of trials increases to infinity. Using MCS is efficient in this case because the probability involved is straightforward and requires a large enough sample size to show this convergence, around 10^4, which in turn obeys the law of large numbers.

This simple case of triangle formation problem can be extended to many real-world systems and applications where random processes involve geometric properties. For example, in structural

engineering, this simulation can be modified and used to study the probabilistic formation of stable structures or frameworks; in wireless communication, the spatial arrangement of nodes can be modeled under this triangle formation simulation to determine connectivity.

3.3. Simple Random Walk Simulated

Another interesting application of Monte Carlo simulation can be dealing with the famous random walk. Here, we simulate the random walks in all three dimensions.

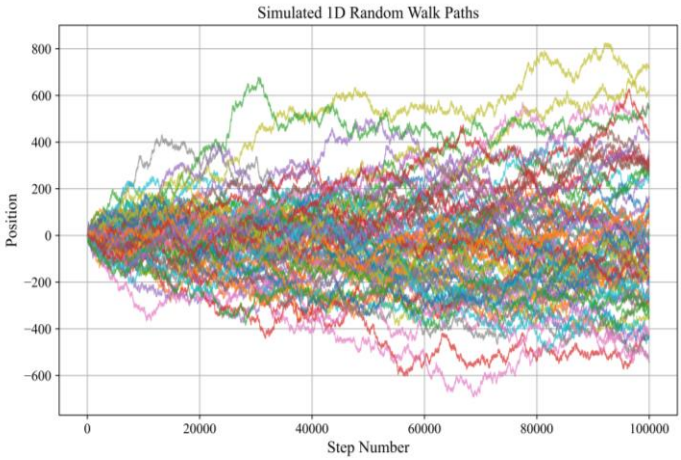

Fig. 4 Simulated Random Walk Paths in Dimension 1 (Photo/Picture credit: Original).

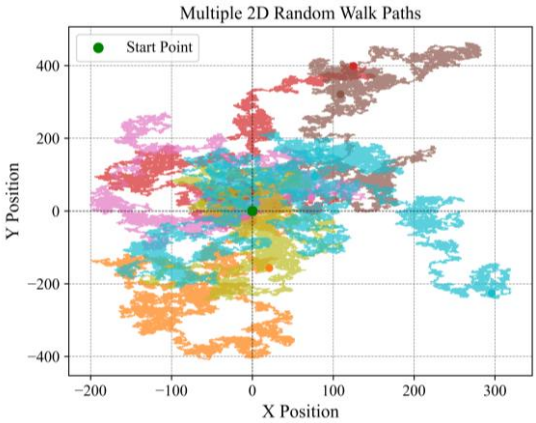

Fig. 5 Simulated Random Walk Paths in Dimension 2 (Photo/Picture credit: Original).

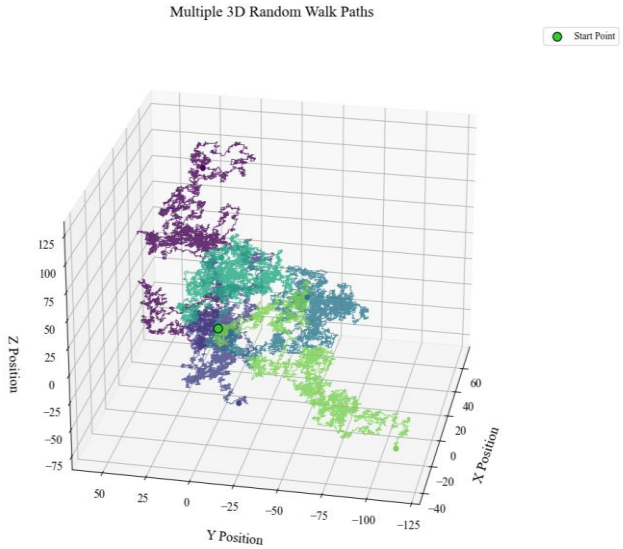

Fig. 6 Simulated Random Walk Paths in Dimension 3 (Photo/Picture credit: Original).

Fig. 4 demonstrates the results of many simulated random walks in one dimension. Each distinct color path represents an independent random walk, where the movement occurs along a single axis, and the movement is recorded by monitoring the position of the particle. Fig. 5 is the outcome of many random walk simulation in two dimensions. There is a corresponding x and y position of the particle as its path diverges. Fig. 6 similarly illustrates the random walk paths simulated in three-dimensional space. The addition of a third axis introduces greater complexity and randomness. Although the simulation results do not suggest that the claim that the random walks will be guaranteed to end up returning to the origin, it is known from theoretical proofs that the probability of eventually returning to the origin is certain (i.e. 100%) in both dimension 1 and 2 but not in 3 [12].

Simulating probability and walk paths of simple random walks is, in large part, helpful for many cases that are not limited to the mathematical derivation. The simulation in dimension two can be used to model particle diffusion and in dimension three the molecular motion or financial modeling. For example, one can use this idea to develop a fair price for options pricing and options trading in finance, to explore classical gambler's ruin as well as electrical networks. More uses and examples can be found here [13].

With this simulated random walk, one can draw some interesting observations and analysis from the data obtained. In the following discussion, the paper will use the data from dimension 1 for simplicity. In dimension 1, the distribution of the end points can be analyzed to see where each end point on a path is located, and one can do this by using MCS again to draw a probability distribution curve to model this case (Fig.7).

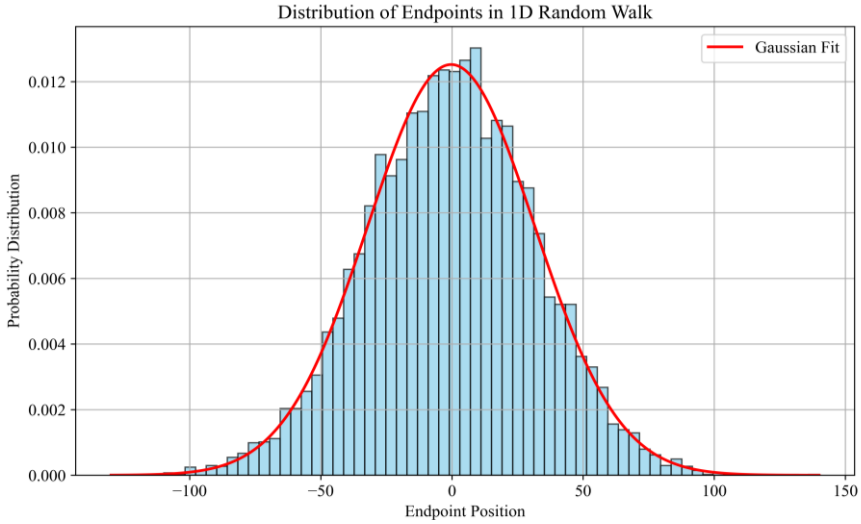

Fig. 7 Probability Distribution of End Points (Photo/Picture credit: Original).

Here, the normal curve fit is generated using the Gaussian function. As expected from the Central Limit Theorem (CLT) the histogram follows a normal distribution, with each data point in the list "endpoints" being a random variable Y defined as the sum of n i.i.d. random variables X_i:

$$Y = X_1 + X_2 + X_3 + \cdots + X_n = \sum_{i=1}^{n} X_i \tag{18}$$

4. Conclusion

Monte Carlo Simulation (MCS) remains one of the most powerful, versatile, and used computation tools to this day that serves to address statistical processes and model stochastic systems in a range of disciplines. Its inherent reliance on randomness determines this method's strength and efficient ability to model complex systems that are often otherwise impossible to analyze, thus providing crucial insights into uncertainty and allowing inferences to be made. Monte Carlo simulation extends beyond the scope of this paper. Advanced branches such as Markov Chain Monte Carlo (MCMC) and the Metropolis-Hastings algorithms, for example, are some of the most important tools that build on simple Monte Carlo methods, and it provides enhanced accuracy when enough data is available.

Despite these characteristics, MCS is not without limitations. For one, the computational cost can be significant in cases such as MCMC that require longer runs to achieve convergence. Also, MCS depends on high-quality random number generators and is susceptible to a lack of variance reduction techniques like importance sampling. In practical challenges, selecting the appropriate Monte Carlo method (cumulative MC or MCMC) can significantly influence performance outcomes. MCS algorithms often struggle to fully exploit parallel computing architectures, and their non-asymptotic error analysis remains underdeveloped.

Looking forward, the intersection of Monte Carlo methods with emerging modern technologies such as machine learning, artificial intelligence, and high-performance computing becomes more apparent and holds more potential. These applications can be further benefited by improving sampling techniques, algorithm developments, and optimized efficiency and accuracy. An example can be to explore advanced PRNGs through the lens of the industrial application. Monte Carlo simulation is not merely a tool but also a method of thinking that can easily adapt to more computation-heavy and analysis-necessary tasks and research. As data becomes more transparent, complex, and easy-to-obtain, MCS will continue to serve as a valuable framework for statistical analysis and inferences.

References

[1] Wikipedia Contributors. Gambler's fallacy. 2024. Available from: https://en.wikipedia.org/wiki/Gambler%27s_fallacy

[2] Kroese D. P., Brereton T., Taimre T., Botev Z. I. Why the Monte Carlo method is so important today. WIREs Computational Statistics, 2014, 6(6): 386–392.

[3] Rubinstein R. Y., Kroese D. P. Simulation and the Monte Carlo method. 3rd ed. Hoboken, New Jersey: John Wiley & Sons; 2017. (Wiley series in probability and statistics).

[4] Feller W. Introduction to probability theory and its applications. 1st ed. Vol. 1. New York: John Wiley & Sons; 1950.

[5] Filmus Y. Two proofs of the central limit theorem. Department of Computer Science, University of Toronto, 2010.

[6] Murthy K. P. N. Monte Carlo: Basics. Indian Society for Radiation Physics; 2000.

[7] Owen A. B. Monte Carlo theory, methods and examples. 2013. /

[8] Wikipedia. List of random number generators – Pseudorandom number generators (PRNGs). 2024. Available from: https://en.wikipedia.org/wiki/List_of_random_number_generators#Pseudorandom_number_generators_(PRNGs)

[9] Park S. K., Miller K. W. Random number generators: Good ones are hard to find. Communications of the ACM, 1988, 31(10): 1192–1201.

[10] Matsumoto M., Nishimura T. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Transactions on Modeling and Computer Simulation (TOMACS), 1998, 8(1): 3–30.

[11] Python Software Foundation. random — Generate pseudo-random numbers. 2024. Available from: https://docs.python.org/3/library/random.html

[12] Pólya G. On the recurrence of a random walk. Mathematische Annalen, 1921, 84(1-2): 149–160.

[13] Shonkwiler R. W., Mendivil F. Explorations in Monte Carlo Methods. 2nd ed. Springer; 2024.