Unit 4: Monte Carlo Methods

Contents

1.0 Introduction

Monte Carlo methods (or Monte Carlo experiments) are a class of computational algorithms that rely on repeated random sampling to compute their results. Monte Carlo methods are often used in simulating physical and mathematical systems. The methods are especially useful in studying systems with a large number of coupled (interacting) degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures. More broadly, Monte Carlo methods are useful for modelling phenomena with significant uncertainty in inputs, such as the calculation of risk in business. These methods are also widely used in mathematics: a classic use is for the evaluation of definite integrals, particularly multidimensional integrals with complicated boundary conditions. It is a widely successful method in risk analysis when compared with alternative methods or human intuition. When Monte Carlo simulations have been applied in space exploration and oil exploration, actual observations of failures, cost overruns and schedule overruns are routinely better predicted by the simulations than by human intuition or alternative "soft" methods.

2.0 Intended Learning Outcomes (ILOs)

By the end this unit, you should be able to:

- Describe Monte Carlo method
- Trace the origin of Monte Carlo method
- Give examples of the application of Monte Carlo method

3.0 Main Content

3.1 Overview of Monte Carlo Method

There is no single Monte Carlo method; instead, the term describes a large and widely- used

class of approaches.
A **Monte Carlo algorithm** is an algorithm for computers. It is used to simulate the behaviour of other systems. It is not an exact method, but a heuristic one. Usually it uses randomness and statistics to get a result.

It is a computation process that uses random numbers to produce an outcome(s). Instead of having fixed inputs, probability distributions are assigned to some or all of the inputs. This will generate a probability distribution for the output after the simulation is run

However, these methods tend to follow the algorithm below:

1. Define a domain of possible inputs.
2. Generate inputs randomly from the domain using a certain specified probability distribution.
3. Perform a deterministic computation using the inputs.
4. Aggregate the results of the individual computations into the final result.

For example, to approximate the value of π using a Monte Carlo method:

1. Draw a square on the ground, then inscribe a circle within it. From plane geometry, the ratio of the area of an inscribed circle to that of the surrounding square is π / 4.
2. Uniformly scatter some objects of uniform size throughout the square. For example, grains of rice or sand.
3. Since the two areas are in the ratio π / 4, the objects should fall in the areas in approximately the same ratio. Thus, counting the number of objects in the circle and dividing by the total number of objects in the square will yield an approximation for π / 4.
4. Multiplying the result by 4 will then yield an approximation for π itself.

Notice how the π approximation follows the general pattern of Monte Carlo algorithms. First, we define an input domain: in this case, it's the square which circumscribes our circle. Next, we generate inputs randomly (scatter individual grains within the square), then perform a computation on each input (test whether it falls within the circle). At the end, we aggregate the results into our final result, the approximation of π.
Note, also, two other common properties of Monte Carlo methods: the computation's reliance on good random numbers, and its slow convergence to a better approximation as more data points are sampled. If grains are purposefully dropped into only, for example, the center of the circle, they will not be uniformly distributed, and so our approximation will be poor. An approximation will also be poor if only a few grains are randomly dropped into the whole square. Thus, the approximation of π will become more accurate both as the grains are dropped more uniformly and as more are dropped.

To understand the Monte Carlo method theoretically, it is useful to think of it as a general technique of numerical integration. It can be shown, at least in a trivial sense, that every application of the Monte Carlo method can be represented as a definite integral.

Suppose we need to evaluate a multi-dimensional definite integral of the form:

$$\psi = \int_0^1 \int_0^1 \cdots \int_0^1 f(u_1, u_2, \ldots, u_n) du_1 du_2 \cdots du_n = \int_{(0,1)^n} f(\boldsymbol{u}) d\boldsymbol{u} \quad \text{.....6}$$

Most integrals can be converted to this form with a suitable change of variables, so we can consider this to be a general application suitable for the Monte Carlo method.

The integral represents a non-random problem, but the Monte Carlo method approximates a solution by introducing a random vector $\boldsymbol{U}$ that is uniformly distributed on the region of integration. Applying the function f to $\boldsymbol{U}$, we obtain a random variable $f(\boldsymbol{U})$. This has expectation:

$$E[f(\boldsymbol{U})] = \int_{(0,1)^n} f(\boldsymbol{u})\, \phi(\boldsymbol{u})\, d\boldsymbol{u} \quad \text{.....(7)}$$

where ϕ is the probability density function of $\boldsymbol{U}$. Because ϕ equals 1 on the region of integration, [7] becomes:

$$E[f(\boldsymbol{U})] = \int_{(0,1)^n} f(\boldsymbol{u})\, d\boldsymbol{u} \quad \text{.....(8)}$$

Comparing [6] and [8], we obtain a probabilistic expression for the integral Ψ:

$$\psi = E[f(\boldsymbol{U})] \quad \text{.....(9)}$$

so random variable $f(\boldsymbol{U})$ has mean Ψ and some standard deviation σ. We define

$$H = f(\boldsymbol{U}^{[1]}) \quad \text{.....(10)}$$

as an unbiased estimator for Ψ with standard error σ. This is a little unconventional, since [10] is an estimator that depends upon a sample $\{\boldsymbol{U}^{[1]}\}$ of size one, but it is a valid estimator nonetheless.

To estimate Ψ with a standard error lower than σ, let's generalize our estimator to accommodate a larger sample $\{\boldsymbol{U}^{[1]}, \boldsymbol{U}^{[2]}, \ldots, \boldsymbol{U}^{[m]}\}$. Applying the function f to each of these yields m independent and identically distributed (IID) random variables $f(\boldsymbol{U}^{[1]})$, $f(\boldsymbol{U}^{[2]})$, … , $f(\boldsymbol{U}^{[m]})$, each with expectation Ψ and standard deviation σ. The generalization of [10]

$$H = \frac{1}{m} \sum_{k=1}^{m} f(\boldsymbol{U}^{[k]}) \quad \text{.....(11)}$$

is an unbiased estimator for Ψ with standard error

$$\frac{\sigma}{\sqrt{m}} \quad \text{.....(12)}$$

If we have a realization $\{\boldsymbol{u}^{[1]}, \boldsymbol{u}^{[2]}, \ldots, \boldsymbol{u}^{[m]}\}$ for our sample, we may estimate Ψ as:

$$h = \frac{1}{m} \sum_{k=1}^{m} f(\boldsymbol{u}^{[k]}) \quad \text{.....(13)}$$

We call [11] the **crude Monte Carlo estimator**. Formula [12] for its standard error is important for two reasons. First, it tells us that the standard error of a Monte Carlo analysis decreases with the square root of the sample size. If we quadruple the number of

realizations used, we will half the standard error. Second, standard error does not depend upon the dimensionality of the integral [6]. Most techniques of numerical integration such as the trapezoidal rule or Simpson's method suffer from the **curse of dimensionality**. When generalized to multiple dimensions, the number of computations required to apply them, increases exponentially with the dimensionality of the integral. For this reason, such methods cannot be applied to integrals of more than a few dimensions. The Monte Carlo method does not suffer from the curse of dimensionality. It is as applicable to a 1000-dimensional integral as it is to a one-dimensional integral.
While increasing the sample size is one technique for reducing the standard error of a Monte Carlo analysis, doing so can be computationally expensive. A better solution is to employ some technique of variance reduction. These techniques incorporate additional information about the analysis directly into the estimator. This allows them to make the Monte Carlo estimator more deterministic, and hence have a lower standard error.

Due to high mathematics required and burden of understanding at this level, we have to stop this discussion here.

3.2 History of Monte Carlo Method

Physicists at Los Alamos Scientific Laboratory were investigating radiation shielding and the distance that neutrons would likely travel through various materials. Despite having most of the necessary data, such as the average distance a neutron would travel in a substance before it collided with an atomic nucleus or how much energy the neutron was likely to give off following a collision, the problem could not be solved with analytical calculations. John von Neumann and Stanislaw Ulam suggested that the problem be solved by modelling the experiment on a computer using chance. Being secret, their work required a code name. Von Neumann chose the name "Monte Carlo". The name is a reference to the Monte Carlo Casino in Monaco where Ulam's uncle would borrow money to gamble.

Random methods of computation and experimentation (generally considered forms of stochastic simulation) can be arguably traced back to the earliest pioneers of probability theory but are more specifically traced to the pre-electronic computing era. The general difference usually described about a Monte Carlo form of simulation is that it systematically "inverts" the typical mode of simulation, treating deterministic problems by *first* finding a probabilistic analogy. Previous methods of simulation and statistical sampling generally did the opposite: using simulation to test a previously understood deterministic problem. Though examples of an "inverted" approach do exist historically, they were not considered a general method until the popularity of the Monte Carlo method spread.

It was only after electronic computers were first built (from 1945 on) that Monte Carlo methods began to be studied in depth. In the 1950s they were used at Los Alamos for early work relating to the development of the hydrogen bomb, and became popularized in the fields of physics, physical chemistry, and operations research. The Rand Corporation and

the U.S. Air Force were two of the major organizations responsible for funding and disseminating information on Monte Carlo methods during this time, and they began to find a wide application in many different fields.
Uses of Monte Carlo methods require large amounts of random numbers, and it was their use that spurred the development of pseudorandom number generators, which were far quicker to use than the tables of random numbers which had been previously used for statistical sampling.

3.4 Applications of Monte Carlo Methods

As stated above, Monte Carlo simulation methods are especially useful for modelling phenomena with significant uncertainty in inputs and in studying systems with a large number of coupled degrees of freedom. Areas of application include:

a. Physical sciences:

Monte Carlo methods are very important in computational physics, physical chemistry, and related applied fields, and have diverse applications from complicated quantum calculations to designing heat shields and aerodynamic forms. The Monte Carlo method is widely used in statistical physics, particularly Monte Carlo molecular modelling as an alternative for computational molecular dynamics as well as to compute statistical field theories of simple particle and polymer models. In experimental particle physics, these methods are used for designing detectors, understanding their behaviour and comparing experimental data to theory, or on vastly large scale of the galaxy modelling.

Monte Carlo methods are also used in the models that form the basis of modern weather forecasting operations.

b. Engineering

Monte Carlo methods are widely used in engineering for sensitivity analysis and quantitative probabilistic analysis in process design. The need arises from the interactive, co-linear and non-linear behaviour of typical process simulations. For example,

- in **microelectronics** engineering, Monte Carlo methods are applied to analyze correlated and uncorrelated variations in analog and digital integrated circuits. This enables designers to estimate realistic 3 sigma corners and effectively optimise circuit yields.
- in **geostatistics and geometallurgy**, Monte Carlo methods strengthen the design of mineral processing flow sheets and contribute to quantitative risk analysis.

c. Visual Designs

Monte Carlo methods have also proven efficient in solving coupled integral differential equations of radiation fields and energy transport, and thus these methods have been used in global illumination computations which produce photorealistic images of virtual 3D models, with applications in video games, architecture, design and computer generated films.

d. Finance and business

Monte Carlo methods in finance are often used to calculate the value of companies, to evaluate investments in projects at a business unit or corporate level, or to evaluate financial derivatives. Monte Carlo methods used in these cases allow the construction of stochastic or probabilistic financial models as opposed to the traditional static and deterministic models, thereby enhancing the treatment of uncertainty in the calculation.

e. Telecommunications

When planning a wireless network, design must be proved to work for a wide variety of scenarios that depend mainly on the number of users, their locations and the services they want to use. Monte Carlo methods are typically used to generate these users and their states. The network performance is then evaluated and, if results are not satisfactory, the network design goes through an optimization process.

f. Games

Monte Carlo methods have recently been applied in game playing related artificial intelligence theory. Most notably the game of Battleship have seen remarkably successful Monte Carlo algorithm based computer players. One of the main problems that this approach has in game playing is that it sometimes misses an isolated, very good move. These approaches are often strong strategically but weak tactically, as tactical decisions tend to rely on a small number of crucial moves which are easily missed by the randomly searching Monte Carlo algorithm.

Monte Carlo simulation versus "what if" scenarios

The opposite of Monte Carlo simulation might be considered deterministic modelling using single-point estimates. Each uncertain variable within a model is assigned a "best guess" estimate. Various combinations of each input variable are manually chosen (such as best case, worst case, and most likely case), and the results recorded for each so-called "what if" scenario.

By contrast, Monte Carlo simulation considers random sampling of probability distribution functions as model inputs to produce hundreds or thousands of possible outcomes instead of a few discrete scenarios. The results provide probabilities of different outcomes occurring.

For example, a comparison of a spreadsheet cost construction model run using traditional "what if" scenarios, and then run again with Monte Carlo simulation and Triangular probability distributions shows that the Monte Carlo analysis has a narrower range than the "what if" analysis. This is because the "what if" analysis gives equal weight to all scenarios.

A **randomized algorithm** or **probabilistic algorithm** is an algorithm which employs a degree of randomness as part of its logic. The algorithm typically uses uniformly distributed random bits as an auxiliary input to guide its behaviour, in the hope of achieving good performance in the "average case" over all possible choices of random bits. Formally, the algorithm's performance will be a random variable determined by the

random bits; thus either the running time, or the output (or both) are random variables
In common practice, randomized algorithms are approximated using a pseudorandom number generator in place of a true source of random bits; such an implementation may deviate from the expected theoretical behaviour.

g. Uses in mathematics

In general, Monte Carlo methods are used in mathematics to solve various problems by generating suitable random numbers and observing that fraction of the numbers which obeys some property or properties. The method is useful for obtaining numerical solutions to problems which are too complicated to solve analytically. The most common application of the Monte Carlo method in mathematics are:

i. Integration

Deterministic methods of numerical integration usually operate by taking a number of evenly spaced samples from a function. In general, this works very well for functions of one variable. However, for functions of vectors, deterministic quadrature methods can be very inefficient. To numerically integrate a function of a two-dimensional vector, equally spaced grid points over a two-dimensional surface are required. For instance a 10x10 grid requires 100 points. If the vector has 100 dimensions, the same spacing on the grid would require 10^{100} points which is far too many to be computed. But 100 dimensions is by no means unusual, since in many physical problems, a "dimension" is equivalent to a degree of freedom.
Monte Carlo methods provide a way out of this *exponential time-increase*. As long as the function in question is reasonably well-behaved, it can be estimated by randomly selecting points in 100-dimensional space, and taking some kind of average of the function values at these points. By the law of large numbers, this method will display convergence (i.e. quadrupling the number of sampled points will halve the error, regardless of the number of dimensions).

ii. Optimization

Most Monte Carlo optimization methods are based on **random walks**. The program will move around a marker in multi-dimensional space, tending to move in directions which lead to a lower function, but sometimes moving against the gradient.

Another popular application for random numbers in numerical simulation is in numerical optimization (choosing the best element from some set of available alternatives). These problems use functions of some often large-dimensional vector that are to be minimized (or maximized). Many problems can be phrased in this way: for example a computer chess program could be seen as trying to find the optimal set of, say, 10 moves which produces the best evaluation function at the end. The travelling salesman problem is another optimization problem. There are also applications to engineering design, such as design optimization.

iii. Inverse problems

Probabilistic formulation of inverse problems leads to the definition of a probability distribution in the space models. This probability distribution combines ***a priori*** (prior knowledge about a population, rather than that estimated by recent observation) information with new information obtained by measuring some observable parameters (data). As, in the general case, the theory linking data with model parameters is nonlinear, the **aposteriori** probability in the model space may not be easy to describe (it may be multimodal, some moments may not be defined, etc.).

When analyzing an inverse problem, obtaining a maximum likelihood model is usually not sufficient, as we normally also wish to have information on the resolution power of the data. In the general case we may have a large number of model parameters, and an inspection of the marginal probability densities of interest may be impractical, or even useless. But it is possible to pseudorandomly generate a large collection of models according to the posterior probability distribution and to analyze and display the models in such a way that information on the relative likelihoods of model properties is conveyed to the spectator. This can be accomplished by means of an efficient Monte Carlo method, even in cases where no explicit formula for the a priori distribution is available.

h. Computational mathematics

Monte Carlo methods are useful in many areas of computational mathematics, where a *lucky choice* can find the correct result. A classic example is Rabin's algorithm for primality testing (algorithm which determines whether a given number is prime). It states that for any *n* which is not prime, a random *x* has at least a 75% chance of proving that *n* is not prime. Hence, if *n* is not prime, but *x* says that it might be, we have observed at most a 1-in-4 event. If 10 different random *x* say that "*n* is probably prime" when it is not, we have observed a one-in-a-million event. In general a Monte Carlo algorithm of this kind produces one correct answer with a guarantee that ***n* is composite, and *x* proves it so**, but another one without, but with a guarantee of not getting this answer when it is wrong too often; in this case at most 25% of the time.

Remark:

In physics, two systems are **coupled** if they are interacting with each other. Of special interest is the **coupling** of two (or more) vibratory systems (e.g. pendula or resonant circuits) by means of springs or magnetic fields, etc. Characteristic for a coupled oscillation is the effect of beat.

4.0 Self-Assessment Exercise(s)

Answer the following questions:

1. How is Monte Carlo method different in approach from the typical mode of simulation, in deterministic problems?
2. How is Monte Carlo method used in Engineering and Mathematics?

5.0 Conclusion

Monte Carlo methods, relies on repeated computation of random or pseudo-random numbers. These methods are most suited to computations by a computer and tend to be used when it is unfeasible or impossible to compute an exact result with a deterministic algorithm (i.e. an algorithm whose behaviour can be completely predicted from the input)

6.0 Summary

In this unit we discussed the following:

- The algorithm of Monte Carlo method
- The history of Monte Carlo method which spurred the development of pseudorandom number generator
- The application of Monte Carlo methods in areas such as physical sciences, Engineering, Finance and Business, telecommunications, Games, Mathematics, etc.

7.0 Further Readings

- Devore, J. L. (2018). *Probability and statistics for engineering and the sciences*. Toronto, Ontario: Nelson.
- Georgii, H. (2013). *Stochastics: Introduction to probability and statistics*. Berlin: De Gruyter.
- Giri, N. C. (2019). *Introduction to probability and statistics*. London: Routledge.
- Johnson, R. A., Miller, I., & Freund, J. E. (2019). *Miller & Freunds probability and statistics for engineers*. Boston: Pearson Education.
- Laha, R. G., & Rohatgi, V. K. (2020). *Probability theory*. Mineola, NY: Dover Publications.
- Mathai, A. M., & Haubold, H. J. (2018). *Probability and statistics: A course for physicists and engineers*. Boston: De Gruyter.
- Pishro-Nik, H. (2014). *Introduction to probability, statistics, and random processes*. Blue Bell, PA: Kappa Research, LLC.
- Spiegel, M. R., Schiller, J. J., & Srinivasan, R. A. (2013). *Schaums outline of probability and statistics*. New York: McGraw-Hill.