EQF13/26: Monte Carlo Simulation

Peter Jäckel* and Eckhard Platen†

Abstract

In this introduction to the chapter on Monte Carlo simulation, we give a brief review of the history of the method, its wide application in today's sciences and engineering, a top-level categorization of its uses, and provide some general background as to the individual articles' subjects in quantitative finance.

1 The history of the Monte Carlo method

The history of Monte Carlo methods goes back a long time. The generic idea of random, or "stochastic", sampling is straightforward and appealing in its elegance and has been used for centuries. Possibly the first systematic application of statistical sampling techniques in science and engineering was by Enrico Fermi in the early 1930's to predict the results of experiments related to the properties of the neutron [Met87] which had recently been discovered by James Chadwick in 1932. In 1947, Stanislaw Ulam suggested to John von Neumann that the newly developed ENIAC computer would give them the means to carry out calculations based on statistical sampling with hitherto unattained efficiency and comparative ease [URvN47]. Their coworker Nicholas Metropolis dubbed the numerical technique "the Monte Carlo method" partly inspired by Ulam's anecdotes of his gambling uncle who "just had to go to Monte Carlo".

Since the deployment of the ENIAC which could do about 5000 additions or 400 multiplications per second and occupied the size of a large room, computing power has grown dramatically. In the early 1970's, a computer design was introduced that had at its heart an electronic component first introduced in 1958, a so-called "integrated circuit". All of a sudden, a computer's *Central Processing Unit* shrank from the size of a domestic refrigerator to that of a fingernail. The number of transistors in a single integrated circuit kept growing at an almost constant exponential rate since then[1], and with it grew the computing power of the computer. In addition to that, miniaturization and the introduction of new materials allowed for equally dramatic increases in computer's clock speeds. At the time of this writing, on a

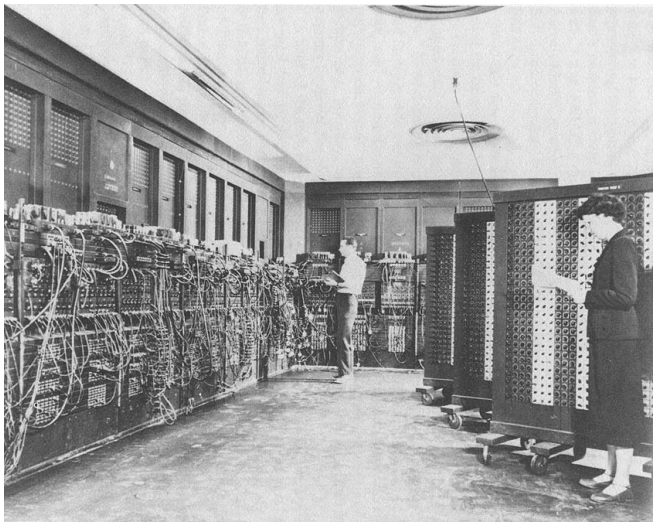

Figure 1: The ENIAC. Smithsonian Institution Photo No. 53192 reproduced from [Cer83] with kind permission by Paul Ceruzzi.

CPU that trades for £25 to retail customers, over 2 billion double precision floating point multiplications can be carried out per second[2] which means that the kind of hardware used these days as a wordprocessor can do in one second what used to take the ENIAC over two months.

It is no surprise, then, that by now the use of Monte Carlo methods has become ubiquitous in science, technology, and business. Simulation techniques are used in: oil well exploration; stellar evolution; electronic chip design; reactor design; quantum chromo dynamics; material sciences; physical chemistry; nanostructure, protein, and polymer research; operations research, e.g., when designing the relationships and control mechanisms between raw materials input, manufacturing, and delivery; ground and air traffic control systems design; communication and computer system design and testing, e.g., network theory; biomolecular research, e.g., cancer drug design; all areas of finance and insurance; weather forecasting (where it is referred to as "ensemble forecasting"[3]); and local authorities planning and commision-

*OTC Analytics
†University of Technology, Sydney, NSW, Australia
Key words and phrases. Monte Carlo, simulation.

[1]This phenomenon was surprisingly accurately predicted by Gordon Moore in 1965, whence it is often referred to as "Moore's law".

[2]This figure is per se, of course, misleading since any realistic calculation involves more than the multiplication of the same two numbers over and over again. It does help, though, to highlight the scale of speed improvements in hardware since the days of the ENIAC.

[3]According to the UK Meteorological Office (www.metoffice.gov.uk/science/creating/daysahead/ensembles), "an ensemble forecasting system samples the uncertainty inherent in weather prediction to provide more information about possible future weather conditions." In other words, it is a mini Monte Carlo simulation consisting of a comparatively small number (~ 24) of individually deterministic weather forecast calculations, each primed with slightly differing input scenarios based on the currently known

ing[4].

Today, the comparative ease of implementation, in combination with the readily available required computer power, makes the use of Monte Carlo techniques more often than not the method of choice. This has gone to the extent where it is deployed (almost) as a black-box tool. As an example for this, we give a quote from the local authority planning and commissioning site mentioned above:

> This spreadsheet based tool gives local authorities access to Monte Carlo statistical modelling techniques. The technique allows local authorities to take account of the different factors which may affect spend levels. This will give authorities the ability to more accurately estimate expenditure to take account of uncertainty.
>
> Monte Carlo is a recognised statistical technique, which is recommended by the Treasury.

2 Basic ideas

The key defining feature of a Monte Carlo simulation may be stated as follows.

Definition 1 *A Monte Carlo method is any technique whose purpose it is to approximate a specific measure defined on a given domain by the aid of sampling according to a pre-determined distributional law.*

Note that this definition does *not* involve any of:-

1. randomness,
2. expectations or moments,
3. stochastic processes.

It may come as a surprise that randomness is not included. It is true that randomness of some of the input numbers is often associated with Monte Carlo methods. In truth, however, and this was already known to the pioneers of Monte Carlo methods, all that is really needed for a Monte Carlo method to succeed is *asymptotic adherence* to the desired pre-determined distributional law on the given domain that is sampled upon. For the sake of explanation, but without loss of generality, we shall assume in the following that the domain is a hypercube $H_m = (0,1)^m$ for some $m \in \mathbb{N}$ and that the desired distribution is uniform in H_m. Any "draw" in the simulation sequence is thus naturally best to be seen as an m-tuple, or a vector $\boldsymbol{u} \in H_m$.

Randomness is predominantly used in conceptual considerations for its convenience of guaranteeing uniform coverage as a consequence of *serial independence of draws*. Obviously, if one vector-valued sample drawn from H_m is in a statistical sense completely independent from the next draw, and so on, then, asymptotically, the set of all draws will cover H_m uniformly. However, it is both intuitive, as well as readily demonstrable, that it

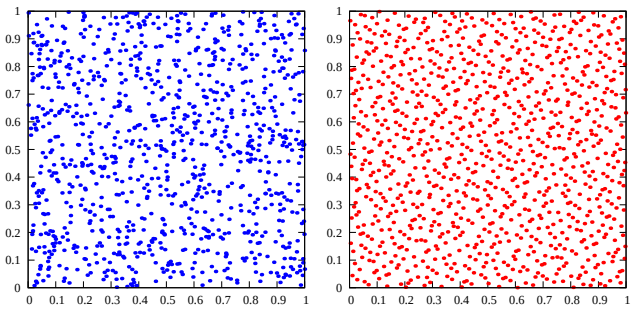

Figure 2: One thousand random (left) and strategic (right) draws from $(0,1)^2$. Note the significantly more uniform coverage of the unit square on the right.

is fairly easy to achieve a more uniform coverage with a deliberate strategy that takes into account all the points that are already part of the simulation. It is in fact possible to prove that at least asymptotically (i.e., in the limit of large numbers of draws), by any self-consistent measure, any strategy (algorithm) to generate numbers that attempts to "avoid" points already drawn, gives more uniform coverage of the sampling domain than random numbers. In this asymptotic sense, amongst all number sequences one can construct, random numbers are the worst possible category of numbers one could use for Monte Carlo simulation purposes. In practical applications, though, a Monte Carlo simulation is hardly ever carried out with the intent to determine some kind of asymptotic behaviour. What matters for real calculations is to have a result as accurate as possible with the fewest necessary iterations, and it is this latter part that makes the use of non-random numbers (also known as *low-discrepancy* or *quasi-random* numbers) slightly more delicate, and this is discussed in article **eqf13-19** by one of the most accomplished researchers in this field, H. Niederreiter. Unfortunately, in the 1940's, no algorithms for the generation of usable non-random numbers were available when Monte Carlo simulations on the ENIAC commenced, despite the fact that the theoritcal foundations for such number sequences had already been laid in 1916 by Hermann Weyl [Wey16]. As a consequence, practically used Monte Carlo simulations relied on the generation of pseudo-random numbers for the first few decades. The term "pseudo-random" is commonly used to indicate that these are computer-generated numbers, as opposed to random numbers as we assume them to occur in natural phenomena such as radioactive decay. The distinction is somewhat philosophical, but noteworthy nevertheless since it is, after all, not actually possible to generate randomness on a machine that is designed to give *deterministic* results. This is summarized in John von Neumann's famous words [Knu81]:

> *Anyone who considers arithmetical methods of producing random numbers is, of course, in a state of sin.*

As it turns out, algorithms for the generation of number sequences that can be considered *random enough* for practicaly use, in some sense, can be devised, though, by the aid of modern number theory, and this is the subject of article **eqf13-03** by P. L'Ecuyer.

weather system status.

[4] see, e.g., www.everychildmatters.gov.uk/resources-and-practice/IG00215

At a mathematical level, Monte Carlo methods have a wide range of applicability. Of course, they can be used simply to compute an approximation for the expecation of a given function, say, $\mathsf{E}[f(\boldsymbol{u})]$, and in finance, this is probably the most common application: draw N vectors $\boldsymbol{u}_1$ to $\boldsymbol{u}_N$, and evaluate the equally weighted N-iteration estimator

$$\hat{\mathsf{E}}_N[f(\boldsymbol{u})] := \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{u}_i)\ . \tag{1}$$

They can, however, also be used for calculations that do not fit well into the format $\mathsf{E}[f(\boldsymbol{u})]$. A common use of Monte Carlo methods is to find a local or global extremum of a given function. Examples for this application are the so-called *Metropolis* algorithm, and the related *simulated annealing* procedure which, in finance, is sometimes used for model calibration. Another frequent application of Monte Carlo methods in finance is the calculation of a *quantile level*, i.e., to find f_q such that the probability that $f(\boldsymbol{u}) < f_q$ is q, or, in other words, f_q is implicitly defined by

$$\mathsf{E}\left[\mathbf{1}_{\{f(\boldsymbol{u})<f_q\}}\right] = q. \tag{2}$$

The simplest algorithm to estimate f_q given q is to draw a number of, say N, iterations for $\boldsymbol{u}$, evaluate $f(\cdot)$ for all the drawn vectors, order the so generated f values by decreasing size, and pick the value at position $\lfloor q \cdot N \rfloor$. The quantile level is only one of various measures for *extreme events* which are discussed in article **eqf13-20**.

The Monte Carlo method is widely accepted as the designated method of choice for the following two types of calculations:-

• Mathematical problems that are not posed as a set of concise equations, but instead are only described as a procedure, or algorithm, or *process*. An example is the estimation of statistics for traffic flow problems where the reaction of individual traffic constituents (e.g., cars) on ambient conditions is known, but, due to their nonlinearity, the net effect of many such constituents reacting to the (re-)actions all other constituents is hard to predict. Some of the problems Fermi, von Neumann, Metropolis, and Ulam originally worked on also belong to this category: they investigated the statistics of neutrons passing through matter by simulating repeated impacts with and deflections by atoms. Some problems are given in the form of stochastic differential equations (SDEs) for which we have no analytical solutions but which can be integrated numerically. Numerical integration of an SDE is mathematically the design of a numerical algorithm that has starting values and an iterative procedure involving numbers to be sampled from a certain domain. Standard Euler integration, for instance, of a simple SDE, over k time steps, typically becomes a sampling problem on $\mathbb{R}^k$ which is usually transformed to $(0,1)^k$. Note that the numerical integration is comprised only by the step that is the conversion of the SDE into a sampling problem. Numerical integration of SDEs is technically not a Monte Carlo technique. It is in principle possible, though rarely done, to evaluate the resulting sampling problem with other means such as multi-dimensional quadratures, etc. In practice, however, numerical integration of stochastic differential equations is usually combined with a Monte Carlo simulation of the integration procedure. Articles **eqf13-01** and **eqf13-02** by E. Platen are dedicated to the subject of numerical integration of stochastic differential equations, and how it interacts with the embedding Monte Carlo simulation. The mathematical background behind the design of many of these numerical integration schemes is the equivalent to Taylor's expansion for stochastic differential equations, and these *stochastic Taylor expansions* are explained in article **eqf13-23**.

• Mathematical problems that require the evaluation of an expectation on a high-dimensional domain. The alternative to a Monte Carlo simulation for such problems are lattice methods. The commonly used error measure for lattice techniques is a term proportional to the square of the lattice discretisation, say, h^2. The number of nodes (i.e., sampling points) in a lattice scales like $N = h^{-d}$ with the dimensionality d. Putting this together shows that the error of a lattice algorithm relates to the number of evaluated points like $\sim 1/N^{(2/d)}$. As d increases, the relationship between a lattice method error and N becomes hopelessly inefficient, and this fact is sometimes referred to as the *curse of dimensionality*. When using random numbers, the commonly used error measure of a Monte Carlo estimator for N iterations scales like $1/\sqrt{N}$, irrespective of the dimensionality of the domain. A straightforward scale comparison indicates that the Monte Carlo error estimator is asymptotically superior in its behaviour as a function of N as soon as $d > 4$. For low-discrepancy numbers, as explained in detail in article **eqf13-19**, the error measure scales closer to $1/N$ which indicates that lattice methods are (asymptotically) superior to those (used within a Monte Carlo simulation) only when the dimensionality d is 1.

Beyond these two categories, there are many other applications of Monte Carlo techniques, often associated with bespoke Monte Carlo algorithms. Many of these algorithms belong to the family of *Monte Carlo Markov Chain* methods [Liu01, GRS96, Bre99]. These methods are, to date, less commonly used in finance. An exception is perhaps the Robbins-Monro algorithm [Aro03] whose basic purpose is to find the root of a function that is defined as an expectation (and can practically be evaluated only by the aid of a Monte Carlo simulation itself). The Robbins-Monro algorithm is designed to avoid having to nest a Monte Carlo simulation as an objective function inside a numerical root finding procedure. It has been used in the context of variance reduction methods which are discussed in article **eqf13-10**.

3 Monte Carlo methods in quantitative finance

The starting point of the use of the Monte Carlo method in quantitative finance, or specifically, in financial engineering, was probably the suggestion by P. Boyle to use it for the valuation of options [Boy77]. Despite the

fact that Monte Carlo methods, initially, had the reputation of being *slow*, they started to become popular rather quickly, probably primarily due to the enormous ease with which they could be implemented. For the valuation of exotic financial contracts, especially in the world of equity and foreign exchange derivatives, and particularly when multiple underlyings were involved, a Monte Carlo method could be implemented directly from the contract's term sheet, without the need for any further mathematics other than a model for the dynamics of the financial underlyings. This made it possible to combine multi-purpose financial models, which only describe the (stochastic) dynamics of the financial underlyings, with *product description language* engines, that allowed any trader, or even structurer, to price, and risk-manage, entirely new financial products within minutes, without any new model development or any code recompilation. The use of Monte Carlo simulations is made particularly easy in this context by the fact that most financial contracts' term sheets are written in a language close to an investor's perception of the evolution of the financial contract *forward in time*, making term sheet definitions intrinsically well compatible with the Monte Carlo method's forward looking nature: *if you can describe it on a term sheet, you can probably implement it as a payoff*. Over time, computers became faster, reducing concerns some people may have had with respect to Monte Carlo methods. Eventually, multi-core computers became commonplace in the financial industry, and since Monte Carlo methods are so extremely easily amenable to multi-threading, possibly more so than any other numerical technique, they could readily be adapted to take advantage of the extra computing power.

The adoption of Monte Carlo methods as a standard technique was helped not only by pure technological progress of computers, though. A whole range of mathematical developments and discoveries provided improvements both in terms of robustness and relative speed that made it possible to carry out Monte Carlo simulations much faster than before. Many of these methods are known as *variance reduction methods*, which is the subject of article **eqf13-10**. A very important generic relative speedup technique is the use of low-discrepancy, as opposed to pseudo-random, numbers, such as the Niederreiter'88 [Nie88], Niederreiter-Xing [NX95], and Sobol' [Sob67, Sob76, PTVF92, Jäc02] numbers, and article **eqf13-19** is dedicated to the theory behind these number sequences. We say *relative* speedup to indicate that the use of these numbers doesn't necessarily make the execution of N iterations actually faster (albeit that these numbers are usually faster to generate than pseudo-random numbers). They provide a relative speedup due to the fact that, for approximately the same residual numerical error, one may need only of the order $\sqrt{N}$ iterations in comparison to the use of pseudo-random numbers. Another type of speedup is associated with bespoke simulation algorithms for specific mathematical problems such as the numerical integration and simulation of square root processes described by SDEs such as

$$\mathrm{d}X = \sigma\sqrt{X} \cdot \mathrm{d}W$$

which is discussed in article **eqf13-09**. Simulations of numerical integrations of SDEs with jumps are described in article **eqf13-15**. The simulations required for the important LIBOR market model for interest rate derivative valuation are particularly challenging due to their intrinsic high-dimensionality, and are elaborated in article **eqf13-16**.

As Monte Carlo methods started being used with financial models, practitioners invented further techniques for specific purposes that are nowadays considered part of the standard toolbox. The handling of sensitivity calculations, in the context of Monte Carlo simulations, can be tricky, due to the inherent nature of a numerical sensitivity calculation to amplify noise. In article **eqf13-04**, the most common techniques to deal with this issue are covered. A mathematically elegant, though in practice perhaps less frequently used framework, that can help to address the same issues is the method known as *integration by parts for stochastic functionals* and is presented in article **eqf13-12**. A particularly interesting development is the family of *weighted Monte Carlo* methods which provide means for noise-reduced sensitivity calculation and noise-reduced model calibration. This framework gives a generic mathematical analysis that encompasses a proof that the popular *control variates* technique for variance reduction is in fact just a special case of a weighted Monte Carlo method, as discussed in article **eqf13-11**.

Finally, we should mention that a lot has been done with respect to the handling of early exercise opportunities that may be given within financial contracts. Apart from simple one-dimensional American equity or FX options, these are very common among exotic fixed income derivatives. Since most of these exotic contracts require rather complex, and high-dimensional, models for realistic risk-management and hedging, it became desirable to find techniques that can be used to evaluate early exercise rights within a simulation setting. In this chapter, we present four articles on the most important techniques for Bermudan options: **eqf13-06** deals with basis function regression methods for Bermudan options; in **eqf13-13**, the Broadie-Glasserman stochastic mesh technique is discussed, article **eqf13-24** explains Bermudan Monte Carlo methods based on exercise boundary optimization, and article **eqf13-25** is on upper bound methods for callable products.

There is much more one could say about Monte Carlo methods in finance, some of it highly applicable in many situations, some of it bespoke for specific applications, and some of it merely for the sake of its mathematical elegance. We believe, though, that all of the most important aspects of Monte Carlo methods in Quantitative Finance are covered in this chapter. For further details beyond this, we recommend the books [KP99], [Jäc02], and [Gla03].

References

[Aro03] B. Arouna. Robbins-Monro algorithms and variance reduction in finance. *Journal of Computational Finance*, 7(2):35–61, 2003.

[Boy77] P. Boyle. Options: a Monte Carlo approach. *Journal of Financial Economics*, 4:323–338, May 1977.

[Bre99] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues*. Springer, 1999.

[Cer83] P.E. Ceruzzi. *Reckoners, the prehistory of the digital computer, from relays to the stored program concept, 1935-1945*. Greenwood Press, 1983. ISBN 0-313-23382-9.

[Gla03] P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2003.

[GRS96] W.R. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1996.

[Jäc02] P. Jäckel. *Monte Carlo methods in finance*. John Wiley and Sons, February 2002.

[Knu81] D. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, 1969,1981.

[KP99] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, 1992, 1995, 1999.

[Liu01] J.S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[Met87] N. Metropolis. The Beginning of the Monte Carlo Method. Special Issue 15, Los Alamos Science, 1987.

[Nie88] H. Niederreiter. Low-Discrepancy and Low-Dispersion Sequences. *Journal of Number Theory*, 30:51–70, 1988.

[NX95] H. Niederreiter and C. P. Xing. Low-discrepancy sequences obtained from algebraic function fields over finite fields. *Acta Arithmetica*, 72:281–298, 1995.

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992. www.library.cornell.edu/nr/cbookcpdf.html.

[Sob67] I. M. Sobol'. On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals. *USSR Computational Mathematics and Mathematical Physics*, 7:86–112, 1967.

[Sob76] I. M. Sobol'. Uniformly Distributed Sequences with an Additional Uniform Property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, 1976.

[URvN47] S. M. Ulam, R. D. Richtmyer, and J. von Neumann. Statistical methods in neutron diffusion. Technical report, Los Alamoms Scientfic Laboratory report LAMS-551, 1947.

[Wey16] H. Weyl. Über die Gleichverteilung von Zahlen mod. eins. *Mathematische Annalen*, 77:313–352, 1916.