Monte-Carlo Methods

Master of Financial Engineering - M2

Jung Jonathan

1 Introduction

1.1 History

This method has been developed by Stanislaw Ulam, John Von Neumann, and Nicholas Metropolis (these scientists were then part of the Manhattan project). It relies on random number generation and its name is a reference to the casinos in Monte-Carlo, where one can play gambling games.

1.2 Goal

Monte Carlo methods are used to:

1. Forecast the average variation of real systems subject to randomness (values of a stock, number of people in a queue, quantity of a given good in an inventory).
2. Estimate (numerically) certain quantities (surfaces, volumes and other such kind of integrals). Usually, these quantities are not random and the Monte-Carlo method uses random as an auxiliary computing tool.

This course will restrict to 1) for models of financial markets. In particular, the variations of the price of a given security on the market will be the main tool to evaluate the price of contracts on these securities (options). In fact, we will need to reproduce on a computer a possible evolution with time of the price of the security. Reproducing such a real process on a computer is often called *numerical simulation*.

1.3 Procedure

1. We consider a real system that is subject to random (for instance, the evolution with time of a stock, or a queing system, an inventory...),
2. We design a (mathematical) model of this system. This model is built using quantities such as random variables or parameters, and equalities that connect these quantities to one another. The distribution of the random variables have to be chosen according to the information available on the modeled system,
3. Using this model, one reproduces the evolution with time of the system in a fictitious futur determined by random numbers, and by the equations in the model. This step precisely consists in performing a numerical simulation,
4. Repeating numerical simulations, one can estimate the average evolution with time of the system, which will be used to do the forecast.

2 Mathematical modeling and numerical simulation on an example

A good model is:

- *Synthetical*; in other words it only contains the important aspects of the system, and does not take other aspects into account,
- *Simple*; the model should rely on few assumptions, be clearly described and ease the understanding of the system.
- *Realistic*; the model should be able to reproduce a possible reality.

In order to design a good model, one needs to follow these steps:

- Study the system in depth, understand the way it works and make it clear which of its aspects are the ones we are interested in,

- Specify the variables and parameters that define the way the system works,
- Use realistic assumptions (for instance on the distribution of random variables),
- Test the assumptions to verify their agreement with reality,
- Write the relations between the variables and the parameters as equations or inequalities according to these assumptions,
- Identify the parameters that can be used to tune the system (decision variables),
- Define a way to quantify the quality of the model.

Example (inventory management): A supermarket sells yogurts 1€ each. This supermarket buys these yogurts for 0.40€ each from a dairy products company. Statistics show that the daily demand is uniformly distributed between 800 yogurts/day and 1400 yogurts/day. Yogurts are delivered each morning, and the leftover yogurts are thrown away at the end of the day. How many yogurts must be ordered every morning in order to maximize profit?

Solution:

- We specify the variables and parameters that define the way that the system works:
  - Call D the daily demand (this is a random variable) and P the profit (also a random variable). Further call q the number of yogurts delivered in each morning (this is a decision variable, i.e. we can decide of its value). We search for the value q^* of q so that the average profit is maximal.
  - The parameters are the retail unit price (0.40€) and selling prices (1€) of a yogurt, as well as the average daily demand (1100 yaourts/day).
- We make assumptions (or choices) for the distribution of random variables:
  - Assume that D is uniformly distributed.
- We write the equations linking the variables and the parameters:
  - The profit is:

$$P = \begin{cases} 0.6 \times q, & \text{if} \quad D \geq q \\ 0.6 \times D - 0.4 \times (q - D), & \text{if} \quad D < q. \end{cases}$$

- We now need to use this model in order to find q^*... Usually, two directions are possible: analytical resolution or numerical simulation.
  - **Analytical resolution:** Since we know the distribution of D (uniform from 800 to 1400 yogurts), we can compute the average profit by integrating the profit as a function of the actual demand D between 800 and 1400. Since we need to compute this average for a uniform distribution, the differential is dD/600:

$$\mathbb{E}(P) = \int_{800}^{q} \big(0.6 \times D - 0.4 \times (q - D)\big) \times \frac{dD}{600} + \int_{q}^{1400} 0.6 \times q \frac{dD}{600}.$$

    We get: $\mathbb{E}(P) = ...$

    Issues: analytical resolution is not always possible. The calculations can be complicated, and an analytical expression of the solution may not even exist!

  - **Numerical simulation:** For a given value of q, generate random values of D uniformly between 800 and 1400: d_1, d_2, ..., d_n. Compute profit p_i made with demand d_i. The average profit for the chosen value of q is then obtained as the average of p_1, p_2, ..., p_n. Repeating this exercise for different values of q, one finds q^*.
  - **Main issues**:
    1. One needs to know the probability distributions of the modeled random variables.
       → How to find realistic distributions?
    2. One needs to generate random numbers according to these distributions
       → How can one do that?
    3. How many numerical simulations are needed to obtain a close estimate to the desired result?
    4. How can one improve the efficiency of numerical simulations?

3 Generation of uniform random variables

The numbers we will generate are called "pseudo-random". These numbers in fact deterministic, but they "look" random. Pseudo-random numbers have one important advantage over purely random numbers: as they are reproducible, they make it possible to test random algorithms in a deterministic way.
The random numbers we will generate in a first step are realizations of random variable uniformly distributed between 0 and 1.

Definition 3.1. *A random variable X is uniformly distributed in [0,1[if its density function is:*

$$f(x) = \begin{cases} 1, & \text{if} \quad 0 \leq x < 1 \\ 0 & \text{otherwise.} \end{cases}$$

3.1 Method : linear congruential generator

This method has been designed by Derrick Henry Lehmer in 1951. The parameters of the method are a (usually large) positive integer m called modulus, a (integer) multiplier a so that $0 < a < m$, and an increment c so that $0 \leq c < m$.

1. Choose a first number y_0 (called "seed") so that $0 \leq y_0 < m$,
2. Recursively compute the terms of the pseudo-random sequence using the following formula:

$$y_{i+1} = [a \times y_i + c] \mod m.$$

3. Since the y_i are between 0 (loose) and m (strict), one obtains a number in $[0, 1[$ dividing y_i by m.

Definition 3.2. *A congruential generator is called mixed if its increment c is not* 0*, and multiplicative if c is equal to* 0*.*

Remark 3.3. *As the number of integers between* 0 *(loose) and m (strict) is finite, the sequence y_i given by the congruential generator necessarily cycles.*

Definition 3.4. *The number of distinct values taken by the y_i in this cycle is called the period of the generator.*

Exercise: If one takes $a = c = 1$ and $y_0 = 0$, what is the sequence y_i? What is the period of this generator? Does the sequence look random?

For some combinations of the parameters m, a, and c, the obtained sequences look random. In order for y_i to look random, the two following conditions must be satisfied:

1. The sequence has a large period,
2. The values in the sequence are both irregular and uniform.

The parameters of the model must therefore be chosen with much care. There exist tests that allow to check whether the sequence is irregular (for instance the spectral test), and tests that allow to check whether the distribution of the values is uniform (adequacy tests). Some of these tests will be seen during the practicals.

Remark 3.5. *If m is the largest value that can be represented by a word of the microprocessor, the modulo operation can be performed very efficiently. For instance, $m = 2^{31}$ for of word of two bytes. Hence, such values of m are very popular. Another possibility that is almost as efficient is to choose $m = 2^{n-1} + 1$ or $m = 2^{n-1} - 1$.*

Theorem 3.6. ***(Hull-Dobell)**: The sequence provided by a mixed linear congruential generator has (maximal) period m if and only if:*

1. *c and m are relatively prime,*
2. *$a - 1$ is divisible by all prime factors of m,*
3. *$a - 1$ is a multiple of* 4 *if m is a multiple of* 4*.*

Definition 3.7. *Let a and m be two positive relatively prime integers. The smallest positive integer λ so that $a^\lambda = 1 \mod m$ is called the order of a modulo m. The number a is called a primitive root modulo m if its order modulo m is maximal. The order modulo m of any primitive root modulo m is called $\lambda(m)$.*

Example: m=9.
If $a = 2$, then:

$$2^2 \, mod \, 9 = 4 \, mod \, 9 = 4$$
$$2^3 \, mod \, 9 = 8 \, mod \, 9 = 8$$
$$2^4 \, mod \, 9 = 16 \, mod \, 9 = 7$$
$$2^5 \, mod \, 9 = 2 \times 7 \, mod \, 9 = 14 \, mod \, 9 = 5$$
$$2^6 \, mod \, 9 = 2 \times 5 \, mod \, 9 = 10 \, mod \, 9 = 1.$$

The order of 2 modulo 9 is 6. If $a = 4$:

$$4^2 \, mod \, 9 = 16 \, mod \, 9 = 7$$
$$4^3 \, mod \, 9 = 4 \times 7 \, mod \, 9 = 28 \, mod \, 9 = 1.$$

The order of 4 modulo 9 is 3. If $a = 5$:

$$5^2 \, mod \, 9 = 25 \, mod \, 9 = 7$$
$$5^3 \, mod \, 9 = 5 * 7 \, mod \, 9 = 35 \, mod \, 9 = 8$$
$$5^4 \, mod \, 9 = 5 * 8 \, mod \, 9 = 40 \, mod \, 9 = 4$$
$$5^5 \, mod \, 9 = 5 * 4 \, mod \, 9 = 20 \, mod \, 9 = 2$$
$$5^6 \, mod \, 9 = 5 * 2 \, mod \, 9 = 10 \, mod \, 9 = 1.$$

The order of 5 modulo 9 is 6. If $a = 7$:

$$7^2 \, mod \, 9 = 49 \, mod \, 9 = 4$$
$$7^3 \, mod \, 9 = 7 * 4 \, mod \, 9 = 28 \, mod \, 9 = 1.$$

The order of 7 modulo 9 is 3. If $a = 8$:

$$8^2 \, mod \, 9 = 64 \, mod \, 9 = 1.$$

The order of 8 modulo 9 is 2. Hence, $\lambda(9) = 6$, and there are two primitive roots modulo $9 : 2$ and 5.

Theorem 3.8. *The period of a multiplicative generator of prime modulus m divides $m-1$. This period is equal to $\lambda(m) = m - 1$ if a is a primitive root modulo m.*

Proof. Exercise!! Indication: Fermat's small theorem states that if m is a prime that does not divide a, then

$$a^{m-1} = 1 \, mod \, m.$$

□

Theorem 3.9. *A multiplicative generator of modulus $m = 2^\beta$ has (maximal) period $\lambda\left(2^\beta\right) = \frac{m}{4}$ if and only if $a \, mod \, 8 = 3$ or 5.*

3.2 Mixture method

- *Data*: two pseudo-random sequences x_i and y_i distributed within $[0, 1[$ and a positive integer N (often taken equal to 100).
- *Result*: a pseudo-random sequence z_i obtained mixing the terms of sequence x_i.
- *Method:*

  1. Set $v(i) = x_i$ for all $i = 1, ..., N$
  2. For $i = 1, 2, ...$, compute $j = 1 + int(N \times y_i)$ and set $z_i = v(j)$ and $v(j) = x_{N+i}$.

  where $int(x)$ stands for the integer part of x.

Remark 3.10. ***(mixing method)**: this method is a heuristics supposed to produce a sequence z_i "more" random than sequence x_i (even if this "greater" randomness cannot be justified theoretically). In particular, the period of sequence z_i is the LCM (Least Common Multiple) of the periods of x_i and y_i (for instance, if the period of x_i is 6 and the period of y_i is 7, then the period of z_i is $6 \times 7 = 42$).*

4 Generating random variables

We aim here at generating random variables with given laws, especially non-uniform ones.

4.1 Rejection sampling

Method (rejection sampling):

- *Data*:
  - a density function $f(x)$ whose one searches for realizations;
  - a density function $g(x)$ whose one can generate realizations and a number M so that $f(x) \leq M \times g(x)$ for all real number x.
- *Result*: A realization y following density $f(x)$.
- *Method of rejection sampling:*
  1. Generate y following density $g(x)$ and z uniformly in $[0, M \times g(y)]$
  2. If $z > f(y)$, start again from 1), otherwise return y.

Remark 4.1. *This method requires a random number generator for density $g(x)$. One may use a uniform random number generator as described above, provided $f(x)$ has a compact support.*

4.2 Inverse transform sampling

Definition 4.2. ***(cumulative function)**: If X is a random variable of density $f(x)$, the cumulative function of X is the function that sends x to the integral of f from minus infinity to x*

$$F(x) = \int_{-\infty}^{x} f(u)du.$$

Proposition 4.3. *A cumulative function is always increasing.*

Method (inverse transform sampling):

- *Data*: A continuous and strictly increasing cumulative function $F(x)$ whose we search for realizations.
- *Result*: A realization y following cumulative $F(x)$.
- *Method:*
  1. Generate a random number z uniformly inside $[0, 1[$,
  2. Return $y = F^{-1}(z)$.

Example (exponential random variable): Let X be an exponential random variable with density:

$$f(t) = \begin{cases} \lambda\, e^{-\lambda\, t} & \text{if} \quad t \geq 0 \\ 0 & \text{if} \quad t < 0 \end{cases}$$

The cumulative function of X is

$$F(t) = \begin{cases} 1 - e^{-\lambda t} & \text{if} \quad t \geq 0 \\ 0 & \text{if} \quad t < 0. \end{cases}$$

The inverse of F is therefore:

$$F^{-1}(t) = -\frac{\ln(1-t)}{\lambda}.$$

Hence, if T uniformly distributed random variable in $[0, 1[$, $X = -\frac{\ln(1-T)}{\lambda}$ is exponentially distributed.

Remark 4.4. *The average of the above exponential random variable is $\frac{1}{\lambda}$.*

4.3 Box and Miller

The normal distribution plays a fundamental role in the modeling of real-world processes. In particular, one finds it in the usual models of financial markets in the Black-Scholes equations for example. The following method (Box and Miller) allows to generate easily normal random variables.

Method (Box and Miller):

- *Data*: An average μ and a standard deviation σ.
- *Result*: Two realizations y_1 and y_2 of independent and normally distributed random variables of average μ and standard deviation σ.
- *Method:*
  1. Generate two realizations z_1 and z_2 of uniformly distributed random variables in $[0, 1[$
  2. Return:

$$y_1 = \sigma \sqrt{-2 \ln(z_1)}\, cos(2\,\pi\, z_2) + \mu,$$
$$y_2 = \sigma \sqrt{-2 \ln(z_1)}\, sin(2\,\pi\, z_2) + \mu.$$

Remark 4.5. *If only one realization is needed, it is only necessary to return y_1 or y_2.*

5 Application: Pricing an asian option

Definition 5.1. ***(asian option)**: An asian option is an option whose payoff is not computed using the value of the underlying security at exercise date T, but using the average value taken by the underlying security between $t = 0$ and $t = T$.*

Remark 5.2. *Asian options are a simple example of exotic options. As with european options, exercise only occurs at expiration date.*

Example: Consider an asian option with strike K and expiration date T over an underlying security of value S_t at time t, the payoff is:

$$\max\left(0, \frac{1}{1+T}\sum_{t=0}^{T} S_t - K\right) \quad \text{for a call}$$
$$\max\left(0, K - \frac{1}{1+T}\sum_{t=0}^{T} S_t\right) \quad \text{for a put}$$

Method (pricing an asian call option using Monte-Carlo mothod): It is assumed that the underlying security follows a binomial model with parameters u and r so that $1/u < 1 + r < u$.

1. Compute the value of the underlying security at every node of the graph (from $t = 0$ to $t = T$, that is, from left to right in the graph),
2. Compute the values of q and $1-q$ for the model over one period, under the assumption that the underlying security prices follows a martingale,
3. Randomly generate M paths from left to right in the graph, and for each of them, compute the payoff of the option at exercise date,
4. Deflate these M payoffs (dividing them by $(1 + r)^T$) and compute the average of the resulting numbers,
5. This average is the price of the option.

Example: If at time $t = 0$, the price of a stock is 100 and that $u = 1.06$, then:

100	106	112.36	119.106
	94.3396	100	106
		88.9996	94.3396
			83.9619
$t = 0$	$t = 1$	$t = 2$	$t = 3$

Further assume that $r = 0.02$. Compute the payoff at exercise of an asian call option with strike $K = 100$ over this stock for the following left to right paths:

1. u - d - u
2. u - u - u
3. d - u - u

Solution:

- Under the assumption that the underlying security prices follows a martingale, we have $q = \frac{1+r-d}{u-d}$.
- There are 8 possible paths: u-u-u, u-u-d, u-d-u, u-d-d, d-u-u, d-u-d, d-d-u, d-d-d.
- We generate M paths $\left(x_i^1, x_i^2, x_i^3\right)_{1\leq i\leq M}$ where x_i^1, x_i^2 and x_i^3 are random number in $[0, 1[$. The prices of the securities $S_{1,i}$, $S_{2,i}$ and $S_{3,i}$ at time $t = 1$, $t = 2$ and $t = 3$ associated to the path $\left(x_i^1, x_i^2, x_i^3\right)_{1\leq i\leq M}$ are

$$S_{1,i} = \begin{cases} u \times 100, & \text{if } x_i^1 < q \\ d \times 100, & \text{elsewhere,} \end{cases}$$
$$S_{2,i} = \begin{cases} u \times S_{1,i}, & \text{if } x_i^2 < q \\ d \times S_{1,i}, & \text{elsewhere,} \end{cases}$$
$$S_{3,i} = \begin{cases} u \times S_{2,i}, & \text{if } x_i^3 < q \\ d \times S_{2,i}, & \text{elsewhere.} \end{cases}$$

  The payoff P_i at exercice date $T = 3$ of the path $\left(x_i^1, x_i^2, x_i^3\right)$ is

$$P_i = \max\left(0, \frac{100 + S_{1,i} + S_{2,i} + S_{3,i}}{1+3} - K\right).$$

- We deflate this M payoffs,

$$\forall i \in [\![1, M]\!], \quad P_i = \frac{P_i}{(1+r)^3}.$$

- the price C of the asian call option is

$$C = \frac{1}{M}\sum_{i=1}^{m} P_i.$$

In this case, we remark that we can compute the exact value. For that, we need to compute the payoffs of each path. There are 8 possible paths: u-u-u, u-u-d, u-d-u, u-d-d, d-u-u, d-u-d, d-d-u, d-d-d. The paths have payoffs

$$u-u-u : \max\left(0, \frac{1}{3+1}\left(100 + 106 + 112.36 + 119.106\right) - 100\right),$$
$$u-u-d :,$$
$$u-d-u :,$$
$$u-d-d :,$$
$$d-u-u :,$$
$$d-u-d :,$$
$$d-d-u :,$$
$$d-d-d : .$$