BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LVIII No. 1/2006	65 - 72	Seria Matematică - Informatică - Fizică

The calculation of the integrals using the Monte Carlo method

Mădălina Cărbureanu

Universitatea Petrol – Gaze din Ploieşti, Bd. Bucureşti 39, Ploieşti, Catedra de Informatică
e-mail: carbureanumada04@yahoo.com

Abstract

The appearance of the Monte Carlo simulation method is placed near the year 1944. This method was interpreted in many ways, it received various definitions, so we can say that this method knew a long and controversial forming and development process. In this article, we shall approach one of the Monte Carlo integration methods, namely the „hit or miss" method.What recommends this method for solving a varied problems range is the fact that, to obtain the best result, usually a much smaller calculation effort is necessary in relation with the problem's intricacy.

Keywords: *simulation, modelling, stochastic process, random numbers generator, hit or miss*

The beginnings of the „hit or miss" method

Knowing the importance and the complexity of the Monte Carlo simulation method, this article will present a short history of the method, various interpretations associated to this method over the time, the range of problems which can be solved using this method and its fields of application. The stress will be laid on one of the integration Monte Carlo methods, namely the „hit or miss" method and we shall focus on the advantages of using this method in the solving process of certains integrals. The basic idea and the method algorithm shall be presented, after which, on the basis of this algorithm, a programme will be achieved for finding an approximation of the $\int_0^1 \cos x dx$ integral. The second application will consist in achieving two programmes (one in the Pascal language and one in the C language) in order to find an approximation for $\int_1^2 \frac{1}{x} dx = \ln(2)$ and the data obtained as a result of three consecutive trials for each of these two programmes will be noted in two tables, to be compared and to allow us to reach the necessary conclusions.

The appearance of the Monte Carlo simulation method is placed around the year 1944, although it is very likely that the method should have appeared much earlier. This method has been invented by the American researchers of „Los Alamos National Laboratory", where it was used for the first time in the simulation process of the neutron behaviour in plutonium or uranium. In time various personalities have paid special attention to the study of this method, for instance: Stanislaw Marcin Ulam, Enrico Fermi, John von Neumann and Nicholas Metropolis.

The method was named so after the Monte Carlo city from the Monaco principality. Clearly, we may wonder what the basis of this association of names was.

The name „Monte Carlo" was given by Nicholas Metropolis during the second world war, when he was involved in the Manhattan project, dedicated to building the atomic bomb, because of the similarities between the simulation process and the process that developes during gambling and because of the fact that Monte Carlo city was the world wide center of roulette games, the roulette being a random number generator. Moreover, the random and iterative nature of this method is very similar to the activities which take place into a casino.

Another argument would be that, in those times there was not the technique of the electronical generation of the random numbers, numbers which were essential in the nuclear physics studies for building the atomic bomb; that is why various methods were used to obtain random numbers in large quantities. An example in this sense would be the fact that the American researchers from Los Alamos were sending various persons to the casinos from Monte Carlo „to collect" the numbers from the roulettes.

We asserted that the Monte Carlo method appeared much earlier, a first example in this sense being the simulation process for estimating the *pi* number's value; this experiment was made for the first time in the second half of the 19th century. This experiment consisted in throwing away, in a purely random manner, a needle upon the surface of a board, on which straight parallel lines were drawn. The estimation of the *pi* value was achieved taking into consideration the number of intersections between the needle and the lines drawn. Nowadays, this problem is known as the „Buffon's needle experiment".

Another example would be that in 1930, Enrico Fermi used the Monte Carlo method to calculate the diffusion of a neutron, and a little later he designed a device named Fermiac, used for error estimation in nuclear reactors.

Various definitions and the application area of the Monte Carlo method

We obviously wonder, what is the Monte Carlo method in fact, what does it consis of? We have several possible answers available for this question, namely:

- It is a method for solving a certain problem by appealing to the random variables, using, in order to find the searched solution, multiple random experiments. At this point the following remark is necessary: in practice, the usage of multiple random experiments is reduced to making certain calculations with random numbers;
- It is a powerful method which can be aplied to those problems which are hard to solve;
- It is a game of luck in which the result obtained further to a big number of actions is precisely the searched solution;
- It is a method which uses powerful random numbers generators;
- It is a method through which the same problem can be solved by different procedures;
- A method which uses random numbers in a deliberate way in order to make calculations which have the structure of a stochastic process;
- A method whose purpose is to indentify the parameters of a distribution by means of observations made on an random variable;
- A method through which the values of a random variable are generated at random, by using a random number generator with a uniform distribution on the [0, 1] interval and of a probability distribution associated with the said random variable;
- A modelling method of the random variables in order to establish their repartition specific features, when these specific features cannot be established by analytical expressions on the basis of the theoretical probability density functions;

- A method by which the real process is replaced with an artificial process, in which, in order to obtain the right results, it is necessary that the random variables generated during the simulation experiments should accurately reproduce the real random variable.[1]

The range of problems for which the Monte Carlo method was developed is very wide, including, for example, solving the linear equations systems, the inversion of matrices, finding the proper vectors and values of a matrix, the calculation of the simple and multiple integrals, solving certain problems at the limit from the field of differential equations, etc.

At present, the applications of the Monte Carlo method find their utility in: cancer therapy, forecasts of all type, solving some traditional physics problems, such as the planets evolution and designing the nuclear reactors. Likewise, the Monte Carlo method is used, in an excessive way, in the processes of modelling the chemical materials and products, modelling the metallic alloys and the analysis of polimer structure.

An interesting application of random numbers consists in calculating the integrals with the so-called Monte Carlo method. There are four integration methods so called Monte Carlo, namely: the incidence method, also known as the „hit or miss" method, the average poll method, the varied check method and the varied antithetic method.

The fundamental idea of the „hit or miss" method

Let us have a continuous function $f(x)$, defined on [0, 1], so that $0 \le f(x) \le 1$. We want to calculate the next integral: $\int_0^1 f(x)dx$. We have the next graphic representation for $f(x)$ function:

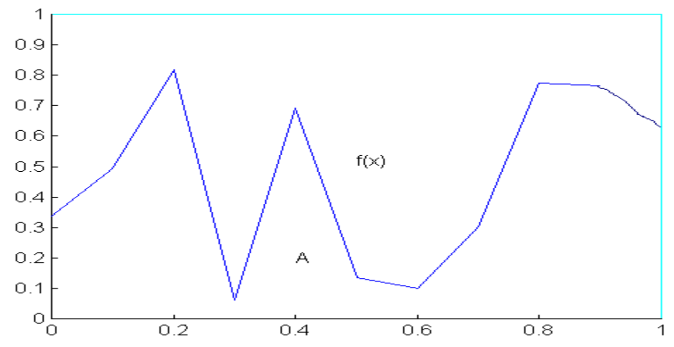

Fig. 1. The surface limited by the f(x) function graphic and the coordinate axes

In accordance with the drawing above, it was marked with A the surface limited by the $f(x)$ function graphic and the coordinate axes, a surface defined as follows: $A=\{(x, y) | \, y \le f(x)\}$. We can easily notice that, from the geometric point of view, $\int_0^1 f(x)dx$ represents the area of the surface A. To calculate this integral, through the Monte Carlo method, we shall use the uniform random variable. We shall consider U and V two independent random variables, with a uniform repartition on the [0, 1] interval. We suppose that we have a sample available, formed

of n independent observations of the (U, V) pair, in other words, we have the $\{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ sequence available, formed of n pairs of random numbers with a uniform distribution in the rectangle from the figure above. We shall note with nr_A the number of those pairs of the sample above which belong under the $f(x)$ graphic. The ratio nr_A/n is known under the name of „hit or miss estimator" of the integral $\int_0^1 f(x)dx$.

Starting from the above, the algorithm which underlies the „hit or miss" method, is the following:

- We have n pairs of random numbers (u_i, v_i) available, with i= 1, 2, …, n, pairs with a uniform repartition in the rectangle from the figure above;
- We shall consider, the (u_i, v_i) pairs, for which $v_i \le f(u_i)$ and we shall note their number with nr_A;
- The value of the ratio nr_A/n represents the „hit or miss estimator" of the integral $\int_0^1 f(x)dx$.

The advantages of using the „hit or miss" method in the solving process of certain integrals

Next, a C programme is presented, made after the algorithm above, in order to calculate the integral $\int_0^1 \cos x dx$.

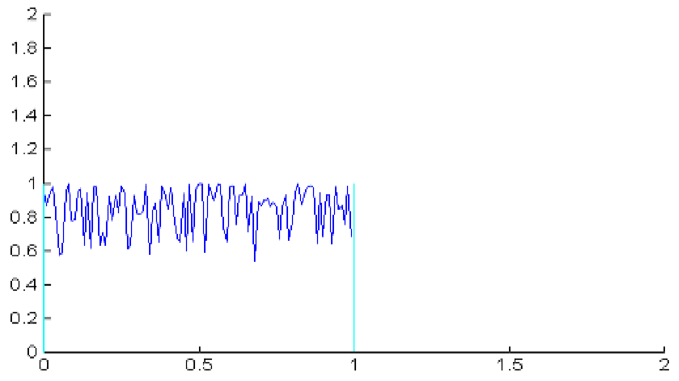

Fig. 2. The region between the x-axis and the graph of cos(x) on the interval [0,1]

```
//The „hit or miss" method
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
double Hit_or_Miss( int N )
  {int NoundergraphA=0;
   double x,y;
   for(int i=0;i<N;i++)
    {x=(double)rand( )/RAND_MAX;
     y=(double)rand( )/RAND_MAX;
     if(y<=cos(x))
                             Nounde
    }
  return(double)NoundergraphA/N;
  }
```

```
void main(void)
{clrscr();
int N;
printf("The  Monte  Carlo1  method-
The Hit_or_Miss method");
printf("\nYou  have  to  introduce
the  number  of  pairs  from  the
rectangle=");
scanf("%d",&N);
printf("The  approximate  value  of
the          integral            is
%lf",Hit_or_Miss(N));
getch();
}
```

In the *N* variable, we retain the number of random numbers pairs with a uniform distribution in our rectangle. From the total of *N* random numbers pairs, we count in the variable *NoundergraphA*, those pairs which verify the relation $y \le \cos(x)$. For a sufficiently big *N*, the integral value is approximated by the *NoundergraphA*/*N* ratio. The *x* and *y* variables are used to obtain those two random numbers rows, with a uniform distribution on the interval [0, 1].

According to the following results, the programme is performed three times, using different values for variable *N*. [2, 3]

```
The Monte Carlo1 method-The Hit_or_Miss method
Introduce the number of pairs from the rectangle= 10000
The approximate value of the integral is 0.843500
The Monte Carlo1 method-The Hit_or_Miss method
Introduce the number of pairs from the rectangle= 15000
The approximate value of the integral is 0.844867
The Monte Carlo1 method-The Hit_or_Miss method
Introduce the number of pairs from the rectangle=1000000
The approximate value of the integral is 0.845401
```

We can notice that the approximation obtained for the integral value in question is so much the better since the number of pairs with a uniform distribution in the rectangle, namely *N*, is greater.

The second application consists in the approximation of the area of a region located between the x-axis and the graph of the curve $y = 1/x$ on the interval [1, 2]. This signifies that we shall find an approximation for the integral $\int_1^2 \frac{1}{x}dx = \ln(2)$. On the interval [1, 2], the curve $y = 1/x$ fulfil the relation $0 \le \frac{1}{x} \le 1$. For this reason, the region which interests us is inside a rectangle with the area equal to 1, bounded by *x*=1, *x*=2, *y*=0 şi *y*=1. The value ln(2) is approximated by random generating (*x*, *y*) pairs which satisfy the relations $1 \le x \le 2$ and $0 \le y \le 1$. Then, we retain those generated pairs which satisfy the condition $y \le 1/x$.

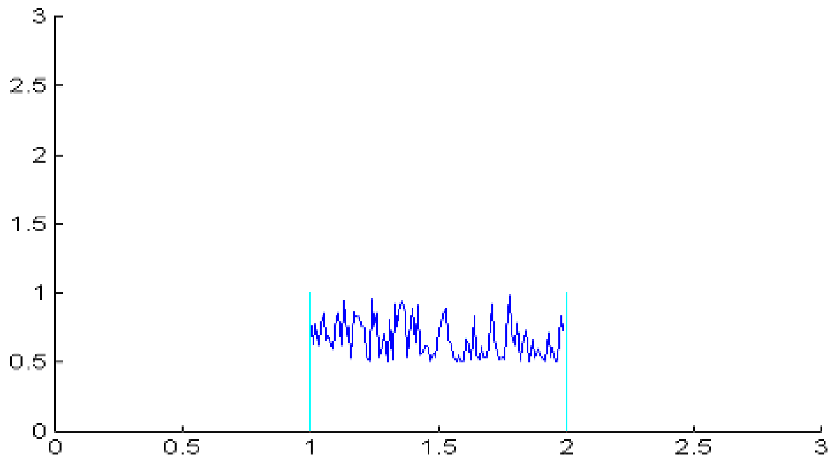

Fig. 3. The region between the x-axis and the graph of the curve y=1/x on the interval [1, 2]

To solve this application, we shall build two programmes, one in the Pascal language, the other in the C language, after which we shall analyze the results obtained.

For each of these two programmes, we are making two tables, in which we note the results obtained after three consecutive trials, and for each step (for *n*=1, 2, 4, 8, …) we do the average of the values obtained after these three trials.

These two programmes below provide a sequence of approximations of the said integral value, by reduplication of the *n* value.

```
The Hit_or_Miss1_Pascal programme;
uses crt;
var step:integer;
    x,y,ValInteg:real;
    NoundergraphA:longint;
    n,i:longint;
begin
clrscr;
for step:=0 to 20 do
begin
n: =1;
for i:=1 to step do
n: =n*2;
NoundergraphA: =0;
randomize;
for i:=0 to n-1 do
Begin
x: =random+1;
y: =random;
if y<=1/x then
   NoundergraphA:=   NoundergraphA
+1;
end;
writeln(' The number of pairs
under the function graph is ',
NoundergraphA);
ValInteg: = NoundergraphA/n;
writeln(n,' The integral value is
',ValInteg:0:6);
readln;
end;
end.
```

Table 1. The results obtained in Pascal after three consecutive trials for ln(2) using the programme

Trials for n=1,2,4,8,…	Trial 1	Trial 2	Trial 3	Average
n=1	1.000000	1.000000	1.000000	1.000000
n=2	1.000000	0.500000	1.000000	0.833333
n=4	0.500000	0.750000	0.500000	0.583333
n=8	0.875000	0.625000	0.500000	0.666667
n=16	0.562500	0.562500	0.625000	0.583333
n=32	0.625000	0.562500	0.718750	0.635417
n=64	0.718750	0.640625	0.625000	0.661458
n=128	0.632813	0.664063	0.757813	0.684896
n=256	0.648438	0.687500	0.683594	0.673177
n=512	0.691406	0.712891	0.642578	0.682292
n=1024	0.700195	0.693359	0.689453	0.694336
n=2048	0.677246	0.704590	0.706055	0.695964
n=4096	0.689453	0.681885	0.695557	0.688965
n=8192	0.690552	0.682251	0.699097	0.690633
n=16384	0.687622	0.692444	0.696838	0.692301
n=32768	0.690582	0.689880	0.694672	0.691711
n=65536	0.691818	0.696793	0.693069	0.693893
n=131072	0.692406	0.692169	0.692688	0.692421
n=262144	0.694290	0.692619	0.693130	0.693346
n=524288	0.692245	0.693340	0.692520	0.692702
n=1048576	0.693000	0.693407	0.692794	0.693067

```
//The Hit_or_Miss1_C programme
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int step;
float x,y,ValInteg;
long int NoundergraphA;
long int n,i;
void main(void)
{clrscr();
for(step=0;step<=20;step++)
 {n=1;
  for(i=1;i<=step;i++)  n=n*2;
  NoundergraphA=0;
  randomize();
  for(i=0;i<n;i++)
  {x=(float)rand()/RAND_MAX+1;
   y=(float)rand()/RAND_MAX;
   if(y<=1/x) NoundergraphA++;
   }
printf("\nThe   number   of   pairs
under the function graph is %ld",
NoundergraphA);
  ValInteg=(float)NoundergraphA/n;
printf("\nFor      n=%ld      the
approximative   value   of   the
integral is %f",n,ValInteg);
  }
  getch();
 }
```

Table 2. The results obtained in C after three consecutive trials for ln(2) using the programme

Trials for n=256,512,1024, …	Trial 1	Trial 2	Trial 3	Average
n=256	0.726562	0.718750	0.765625	0.736979
n=512	0.683594	0.695312	0.734375	0.704427
n=1024	0.685547	0.668945	0.705078	0.686523
n=2048	0.679199	0.682617	0.709961	0.690592
n=4096	0.687256	0.680664	0.698486	0.688802
n=8192	0.693848	0.687378	0.697266	0.692831
n=16384	0.697388	0.689697	0.697083	0.694723
n=32768	0.695435	0.691956	0.693573	0.693655
n=65536	0.696091	0.692078	0.694168	0.694112
n=131072	0.694458	0.695168	0.693001	0.694209
n=262144	0.693958	0.695122	0.693153	0.694078
n=524288	0.693396	0.693481	0.693541	0.693473
n=1048576	0.692236	0.693190	0.693511	0.692979

We know that $\ln(2) \approx 0.6931$, which is exactly the value that we must obtain. From table1, obtained through the compiling of the programme written in the Pascal language, we see that the aproximative value of ln(2) is 0.693067, and by rounding off this value, we obtain that the value of $\ln(2) \approx 0.6931$.

From table2, obtained through the compiling of the programme written in the C language, we see that the approximative value of ln(2) is 0.692979, value which is not so great as the value obtained by compiling the programme written in the Pascal language.[4, 5, 6]

Conclusions

The conclusion we can draw from the remark above would be that, when we have to solve a certain problem, from the multitude of programming languages available, we must choose that language which can provide us the best solutions in the shortest time.

As seen in the beginning of this article, in time, this method was interpreted in many ways, it received various definitions, specific features, concepts, so we can say that this method has brought to light a long and controversial forming and development process.

We can say that this method, in its rudimentary form, was very much used along the centuries, but only in the past decades has it seen a fast „growing-up”, enough to solve the most sophisticated applications with this method.

What recommends this method for solving a wide range of problems is the fact that in order to obtain the best result, usually a much smaller calculation effort is necessary in relation with the problem's intricacy, as compared to other determinist methods.

In the end, we must say that the development of this method has been practically impossible without the use of modern computers. The processing of an enormous volume of information, which makes this method very efficient, suposses the existence of last generation machines, which can process a very big number of particular cases and then make the statistical processing of the numerical data obtained.

References

1. Hammersley, J.M., Handscombe, D.C.- *Monte Carlo methods*, John Wiley, New York, 1964
2. Gorunescu, F., Prodan, A.- *Modelare stochastică şi simulare*, Editura Albastră, Cluj- Napoca, 2001
3. Knuth, D.E.- *Arta programării calculatoarelor*, Volumul 2, Editura Teora, Bucureşti, 2000
4. Fishman, G.S.- *Monte Carlo: Concepts, Algorithms and Applications*, Springer Verlag, 1996
5. Gorunescu, F., Gorunescu, M., Sterpu, M.- Hit or Miss Monte Carlo Simulation for Improper Integrals, *Analele Universităţii din Craiova, Seria Matematică-Informatică*, *vol. XX*, 22-26, 1993
6. Rubinstein, R.Y.- *Simulation and the Monte Carlo method*, John Wiley, New York, 1981

Calculul integralelor folosind metoda Monte Carlo

Rezumat

Apariţia metodei de simulare Monte Carlo este plasată în jurul anului 1944. Această metodă a cunoscut multe interpretări, a primit definiţii variate, prin urmare putem afirma că această metodă a parcurs un lung şi controversat proces de formare şi dezvoltare. În acest articol, vom prezenta una dintre metodele de integrare Monte Carlo, cunoscută sub numele de metoda „hit or miss".Ceea ce recomandă utilizarea acestei metode în rezolvarea unei game variate de probleme este faptul că, pentru a obţine cel mai bun rezultat, este necesar un efort de calcul mic în comparaţie cu dificultatea problemei.