BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LVIII No. 2/2006	79 - 86	Seria Matematică - Informatică - Fizică

The Calculation of the Integrals using the Sample Mean Method

Mădălina Cărbureanu

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

Abstract

An interesting random number application is offered by the calculation of the definite integrals with the sample mean Monte Carlo method. What recommends the use of the sample mean method in the solving process of definite integrals are the advantages offered by this. One of the most important advantages is the quick way to obtain values which approximate the solution of the integral in question, from the best, with a minimum calculation effort. This method can be successfully applied even for difficult definite integrals.

Key words: *simulation, approximation, Monte Carlo integration, random numbers, sample mean method*

Introduction

There are four integration methods so called Monte Carlo, namely: the incidence method, also known as the hit or miss method, the sample mean method, the varied check method and the varied antithetic method.

In this article the stress will be laid on one of the integration Monte Carlo methods, namely the sample mean method. The basic idea and the method algorithm shall be presented first, after which, on the algorithm basis, we shall find the best approximations solutions for the next definite integrals:

$$\int_0^1 e^{arctgx} * \frac{1}{\left(x^2+1\right)^{3/2}} dx \tag{1}$$

and

$$\int_0^1 2x \sin\left(x^2+1\right) * e^{\cos\left(x^2+1\right)} dx \tag{2}$$

For each of these two definite integrals we shall build programmes in the C and Pascal languages and the data obtained, as a result of three consecutive trials for each of these programmes, will be noted in tables, on which bases we shall establish the value which offers the best approximation for the integral solution in question. Moreover, we shall calculate the CPU time for each programme and we shall draw the necessary conclusions.

The Fundamental Idea of the Sample Mean Monte Carlo Method

Let's have an unbounded function *f(x)*, defined on a domain which contains the (0, 1) interval. We want to calculate a definite integral, namely

$$\int_0^1 f(x)dx\,. \tag{3}$$

We have the next graphic representation for *f(x)* function [2]:

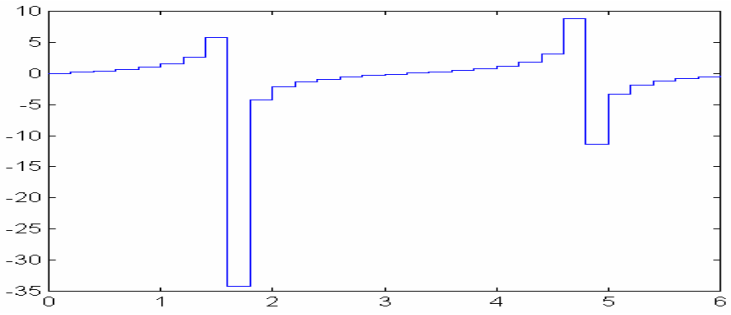

Fig. 1. The graphic representation for *f(x)* with the „stairs" function

Let's have *X* a continuous random variable with a *g(x)* density repartition. We know that the density repartition for a continuous random variable *X* is a function *g(x)* with the next properties:

$$g(x)>0 \text{ for } (\forall)x\in(\text{a, b}) \text{ and } \int_a^b g(x)dx=1, \tag{4}$$

while the average is defined as:

$$M(X)=\int_a^b xg(x)dx\,, \tag{5}$$

where $x\in(a, b)$ and *g(x)* is the density repartition. From the average definition and from the fact that our function *f(x)* is an unbounded function, it results that

$$M[f(x)]=\int_{-\infty}^{\infty} f(x)g(x)dx\,. \tag{6}$$

Further on we shall consider the private case in which the continuous random variable *X* is replaced by the continuous random variable *U*, uniformly distributed on the [0, 1] interval. We know that the continuous random variable *U* is uniform on (0, 1) if

$$g(x)=\begin{cases}1, x\in(0,1)\\ 0, otherwise\end{cases}. \tag{7}$$

and uniform on (*a*, *b*) if

$$g(x)=\begin{cases}\dfrac{1}{b-a}, x\in(a,b)\\ 0, otherwise\end{cases}. \tag{8}$$

Because the continuous random variable U is uniform on (0, 1) it results that

$$g(x)= 1/(1-0)=1, \tag{9}$$

therefore U has a constant density repartition $g(x)$. Using the average calculation formula, we have:

$$M[f(U)]= \int_{-\infty}^{\infty} f(x)g(x)dx \overset{g(x)=1}{=} \int_{-\infty}^{\infty} f(x)dx = \int_{0}^{1} f(x)dx , \tag{10}$$

therefore we obtained precisely the integral which we want to calculate. The hard law of the big numbers says that if we have a row of independent random variables $X_1, X_2, ..., X_n, ...$, with an identical repartition, of average

$$M[X_n]=\mu , \tag{11}$$

then

$$\mathrm{P}\left\{ \lim_{n\to\infty} \frac{X_1 + X_2 + ... + X_n}{n} = \mu \right\} = 1 . \tag{12}$$

Further on we shall use the result obtained above and the hard law of the big numbers and we shall make the next observation: if F is a random variable of finite average $M[F]$ and through $\overline{f}$ we shall understand the selection mean, given by the formula

$$\overline{f} = \frac{1}{n}\sum_{i=1}^{n} f_i , \tag{13}$$

where f_i is a sample of the F values, then, in accordance with the hard law of the big numbers, we obtain that [3],[4]:

$$P\left\{ \lim_{n\to\infty} \overline{f} - M[F] = 0 \right\} = 1 . \tag{14}$$

In other words, if $U_1, U_2, ..., U_n$ are independent random variables with a uniform repartition on the (0, 1) interval, then $f(U_1), f(U_2), ..., f(U_n)$ are also independent random variables with an identical repartition and of μ average. The hard law of the big numbers allows us to say that

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(U_i) = M[f(U)] = \int_{0}^{1} f(x)dx , \tag{15}$$

therefore the integral value $\int_{0}^{1} f(x)dx$ can be approximated by generating a big number of U_i random numbers and taking the average value of the $f(U_i)$ range like approximation, finally we have that

$$\int_{0}^{1} f(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} f(U_i) . \tag{16}$$

This approximation method of an integral value using random numbers is named the sample mean method. Using those above, we obtain the next integration algorithm, named the sample mean Monte Carlo algorithm[1]:

1. We shall consider n random numbers $\{u_i\}$, i=1, 2, …, n, uniformly distributed on [0, 1];
2. We calculate $f(u_i)$ values;
3. We calculate the sample mean

$$\overline{f} = \frac{1}{n}\sum_{i=1}^{n} f(u_i) \ ; \tag{17}$$

4. $\overline{f}$ is the sample mean estimator for the integral $\int_0^1 f(x)dx$.

The Use of the Sample Mean Monte Carlo Method in the Solving Process of the Definite Integrals

The first application consists in achieving two programmes, one in the C language, one in the Pascal language, programmes in which we shall use the sample mean algorithm in order to calculate the definite integral:

$$\int_0^1 e^{arctgx} * \frac{1}{\left(x^2+1\right)^{3/2}} dx \, .$$

The execution result of these two programmes consists in displaying the approximate value of the integral in question. For each of these two programmes, we shall build a table, in which we shall note the data obtained as result of three consecutive trials and for each step (for N=10, 100, 1000, 10000 and 100000) we shall calculate the average of the data obtained. Then we shall analyse the data obtained as a result of the execution of these two programmes and the CPU time.

In the solving process of these two integrals, we shall use the following pseudocode, where $f(x)$ is the function for which we shall calculate the integral:

```
long step, N, I;
real U,S;
real ValInteg;
begin
  for step←0, 5 do begin
      N←10;
      for I←1,step do N←N*10;
      endfor;
      S ←0;
  Initialize the random number
  generator;
      for I←0, N-1 do begin
        Generate U∈[0, 1];
        S←S+f(x);
      endfor;
      ValInteg←S/N;
      write ”The approximate value
      of the integral is”,
      ValInteg;
 endfor;

end.
```

The programme made in the C language to obtain the approximate solution of the integral in question is the following:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
double f(double X)
{ return(exp(atan(X))*1/
  (pow(pow(X,2)+1,1.5)));
}
void main(void)
{long step;
double U,S;
long N,I;
float ValInteg;
clrscr();
for(step=0;step<=5;step++)
{
 N=10;
 for(I=1;I<=step;I++) N=N*10;
 S=0;
 randomize();
 for(I=0;I<N;I++)
  {U=(double)rand()/RAND_MAX;
   S=S+f(U);
  }
 ValInteg=(float)S/N;
 printf("\nFor N=%ld the
approximative value of the
integral is %f",N,ValInteg);
 }
getch();
}
```

Table 1. The results obtained in C after three consecutive trials

Trials for *N*=10,100, 1000,...	Trial1		Trial2		Trial3			
	Value	CPU 1(s)	Value	CPU 2(s)	Value	CPU 3(s)	Av_Value	Av_CPU
N=10	1.035104	0.000000	1.039890	0.000000	1.066327	0.000000	1.047107	0.000000
N=100	1.057052	0.000000	1.037571	0.000000	1.043039	0.054945	1.045887	0.018315
N=1000	1.044955	0.109890	1.055796	0.164835	1.049376	0.164835	1.050042	0.146520
N=10000	1.052236	1.538462	1.048766	1.593407	1.049095	1.593407	1.050032	1.575092
N=100000	1.051035	16.043956	1.050687	16.208791	1.050554	16.263736	1.050758	16.172161

As we see in table 1, the ratio for *N* is 10 and the corresponding ratio for average CPU is, in average, 9.66, not taking into account the first run, for *N*=10, instantly executed. Hence, we estimate that for calculating the integral value with a minimal error the CPU time is good enough. Additionally, these CPU times were obtained on a PentiumII processor, 266MHz, 64MB RAM.

As result of testing the programme made in the Pascal language, using the function below, we obtained the following data:

```
function f(X:real):real;
var expresie:real;
begin
  expresie:=(Exp(ArcTan(X))*1/ ((Sqr(X)+1)*Sqrt(Sqr(X)+1)));
  f:=expresie;
end;
```

Table 2. The results obtained in Pascal after three consecutive trials

Trials for *N*=10,100, 1000,…	Trial1		Trial2		Trial3			
	Value	CPU 1(s)	Value	CPU 2(s)	Value	CPU 3(s)	Av_Value	Av_CPU
N=10	1.044356	0.000000	1.059994	0.000000	1.043195	0.000000	1.049181	0.000000
N=100	1.041314	0.000000	1.047694	0.060000	1.045527	0.050000	1.044845	0.036666
N=1000	1.051813	0.060000	1.052816	0.050000	1.047059	0.050000	1.050562	0.053333
N=10000	1.048587	0.610000	1.051744	0.660000	1.050298	0.600000	1.050209	0.623333
N=100000	1.050760	6.100000	1.051225	5.930000	1.050424	5.930000	1.050803	5.986666

Data in table 2 show that the ratio for average CPU is, in average, 7.57, not taking into account the first run, for *N*=10, also instantly executed. Hence, we estimate that for calculating the

integral value with a minimal error the CPU time is even better than the CPU time obtained above.Additionally, these CPU times were also obtained on a PentiumII processor, 266MHz, 64MB RAM.

Solving

$$\int_0^1 e^{arctgx} * \frac{1}{\left(x^2+1\right)^{3/2}} dx ,$$

we get that the solution of this integral is 1.050881. What we are interested in is to obtain values which best approximate the solution of our integral. Next, we shall refer to the results and the CPU time obtained through the execution of these two programmes. In the table obtained through the execution of the programme written in the C language, the value which offers a rather good approximation for the solution of our integral is 1.050758, value obtained at the step N=100000. In the second table, obtained through the execution of the programme written in the Pascal language, the value which approximates the solution of our integral is 1.050803, value which is better than the value obtained through the execution of the programme written in the C language, therefore the value 1.050803 is the best approximation for the solution of our integral. We notice that the best approximations for the solution of our integral have been obtained at the step N=100000, therefore we can say that to obtain very good approximations for the solution of the integral in question, the step N must be bigger and bigger.

In order to determine the CPU time using the Pascal programme, we chose the *GetTime* function. We also observe that the CPU time obtained through the execution of the Pascal programme is better (is shorter) than the CPU time obtained through the execution of the C programme and that is why the solution of the integral in question was obtained in a quick way by executing the Pascal programme.

The second application consists in the use of the sample mean method for solving an integral which isn't so easy to solve, namely

$$\int_0^1 2x \sin\left(x^2+1\right) * e^{\cos\left(x^2+1\right)} dx .$$

For solving this application in the C language, we shall apply the same idea used in the first application, but in this case the function is the following:

```
double f(double X){
  return (2*X*sin(pow(X,2)+1)*exp(cos(pow(X,2)+1)));}
```

Table 3. The results obtained in C after three consecutive trials

Trials for N=10,100, 1000,…	Trial1		Trial2		Trial3			
	Value	CPU 1(s)	Value	CPU 2(s)	Value	CPU 3(s)	Av_Value	Av_CPU
N=10	1.207893	0.000000	0.989208	0.000000	0.745904	0.000000	0.981001	0.000000
N=100	1.046251	0.000000	1.085825	0.000000	1.019167	0.000000	1.050414	0.000000
N=1000	1.074144	0.054945	1.038392	0.054945	1.037802	0.109890	1.050112	0.073260
N=10000	1.056226	0.659341	1.058849	0.659341	1.054348	0.659341	1.056474	0.659341
N=100000	1.055802	6.813187	1.059495	7.032967	1.055233	6.868132	1.056843	6.904762

As we see in table 3, the ratio for N is 10 and the corresponding ratio for average CPU is, in average, 9.73, not taking into account the first runs, for N=10 and N=100, instantly executed. Hence, we estimate that for calculating the integral value with a minimal error the CPU time is good enough. Additionally, these CPU times were also obtained on a PentiumII processor, 266MHz, 64MB RAM.

The function and the results obtained through the Pascal programme execution are the following:

```
function f(X:real):real;
var expresie:real;
begin
  expresie:=2*X*Sin(Sqr(X)+1)*
  Exp(Cos(Sqr(X)+1));
  f:=expresie;
end;
```

Table 4. The results obtained in Pascal after three consecutive trials

Trials for *N*=10,100, 1000,…	Trial1		Trial2		Trial3			
	Value	CPU 1(s)	Value	CPU 2(s)	Value	CPU 3(s)	Av_Value	Av_CPU
N=10	1.083401	0.000000	0.854219	0.000000	1.181345	0.000000	1.039655	0.000000
N=100	1.034007	0.000000	1.099127	0.000000	1.032828	0.000000	1.055320	0.000000
N=1000	1.054937	0.000000	1.039869	0.000000	1.060209	0.060000	1.051671	0.020000
N=10000	1.064352	0.390000	1.055158	0.440000	1.060414	0.380000	1.059974	0.403333
N=100000	1.061273	4.230000	1.058183	4.230000	1.054173	4.230000	1.057876	4.230000

Data in table 4 show that the ratio for average CPU is, in average, 9.57, not taking into account the first two runs, for *N*=10 and *N*=100, also instantly executed.

Because the best approximations of an integral solution are obtained at the *N*=100000 step, fact confirmed by the first application, we can assert with certainty that the approximate solution of our integral is 1.057876, value obtained through the execution of the Pascal programme.

Also, the CPU time obtained through the execution of the Pascal programme is better (is shorter) than the CPU time obtained by executing the C programme, from where we can detach the same conclusion like in the first application. The sample mean method utility is entirely justified, because this method can be also successfully applied for integrals which aren't so easy to solve, supplying very good approximate solutions for these, in the shortest time, with a minimum effort.

Conclusions

What recommends the sample mean Monte Carlo method for solving a wide range of definite integrals is the fact that in order to obtain the best results, usually, a much smaller calculation effort is necessary in relation to the integral difficulty, unlike other solving methods of this kind of integrals. Moreover, if the first integration Monte Carlo method, namely the „hit or miss" method uses at the base of the method algorithm a bounded function *f(x)*, in the sample mean method case it gives up the *f(x)* function boundary, which makes this method much more efficient, with the possibility to apply this method to a much larger problems gamut, obviuosly constituting an advantage.

Furthermore, the sample mean method is superior to the „hit or miss" method, because the average of the error square obtained through the sample mean method is smaller or equal to the error obtained through the „hit or miss" method.

As we have already noticed through the applications above, the values obtained through the execution of the programme written in the Pascal language were offering a very good approximation for the integral solution in question, unlike the values obtained through the execution of the programme written in the C language, which also offered a rather good approximation, but not so good like that obtained through the Pascal programme, this because

the Pascal language is benefiting from random numbers generators much more powerful than those of the C language. Determining the CPU time for each programme, we observed that the CPU time for the Pascal programme is better (is shorter) than the CPU time obtained through the execution of the C programme and, that is why, we can assert that we obtained the best approximation for the integral in question in a shorter time than using the C programme.

In the integrals solving process, the sample mean Monte Carlo method is based on the intensive use of random numbers, which results in obtaining those very good values which approximate the integrals solutions very easily, obviously constituting an advantage more.

Clearly, the sample mean algorithm can be implemented in any programming language, as C++, Visual Basic, Java, etc., obvious with even better results, due to the facilitations and the random numbers generators which are more powerful than those of the Pascal and C languages, the choice of one of these languages pertaining to the programmer who is interested in this method.

References

1. Gorunescu, F., Prodan, A. - *Modelare stochastică şi simulare,* Editura Albastră, Cluj-Napoca, 2001
2. Ghinea, M., Fireţeanu, V. - *Matlab. Calcul numeric. Grafică. Aplicaţii*, Editura Teora, Bucureşti, 2004
3. Hammersley, J.M., Handscombe, D.C. - *Monte Carlo methods*, John Wiley, New York, 1964
4. Knuth, D.E.- *Arta programării calculatoarelor*, Vol., Editura Teora, Bucureşti, 2000
5. Rubinstein, R.Y. - *Simulation and the Monte Carlo method*, John Wiley, New York, 1981

Calculul integralelor folosind metoda mediei de sondaj

Rezumat

Acest articol prezintă una dintre metodele de integrare Monte Carlo, şi anume metoda mediei de sondaj. Prin cele două aplicaţii prezentate se urmăreşte a se evidenţia uşurinţa de aplicare a acestei metode în procesul de rezolvare a integralelor definite, precum şi rapiditatea în obţinerea unora dintre cele mai bune soluţii aproximative pentru integralele în cauză., ceea ce recomandă aplicarea cu succes a acestei metode chiar şi pentru integrale definite destul de dificil de rezolvat.