Mathematical Statistics raport

Monte Carlo Method

Natalia Pyka 177887

Lecturer: R. Zdunek, PhD

Table of contents

1. Monte Carlo Method overview

Monte Carlo methods family, also known as Monte Carlo experiments, is a wide class of computational algorithms whose purpose is to obtain desired results by i.e. performing a simulation or computing the given equation many times for random input data. As a result, we obtain heuristic calculations similar to the results of „gambling with the experiment in the casino", hence the method was named after this well-known gambling city. The method of using such random experiments was firstly proposed by a polish matematician Stanisław Ulam who described the idea to John von Neumann. However, its first uses date back to 18th century when Buffon used stochastic methods to estimate the probability of a needle falling across a line in a uniform grid of parallel lanes. William Gosset also used similar methods in his experiments conducted in early 1900's.

The generic and simplified algorithm of Monte Carlo method is following:

- Define a domain or set of possible inputs
- Generate random input samples from a probability distribution over the domain (random number generator is required here)
- Perform predefined deterministic computations that make use of the input samples
- Gather the results.

To obtain reasonable results, number of calculated inputs should be large and a good random number generator should be applied to ensure that the input data are random enough.

Monte Carlo methods are widely used everywhere where it is impossible or not feasible to apply a deterministic algorithm. In example, they are used when calculating very complex integrals, when solving optimization problems or simulating systems with many coupled degrees of freedom. Other applications of Monte Carlo method include testing, calculating financial risk and generally every application with high uncertainty in inputs.

Cellular Potts model may be an example of computational model making use of the Monte Carlo method. Nowadays, it is used to simulate foam, fluid flows and biological tissues. In this model, every cell is represented as domain of pixels with the same cell id (also called spin). Typical Potts configuration model is shown in Fig.1.1.

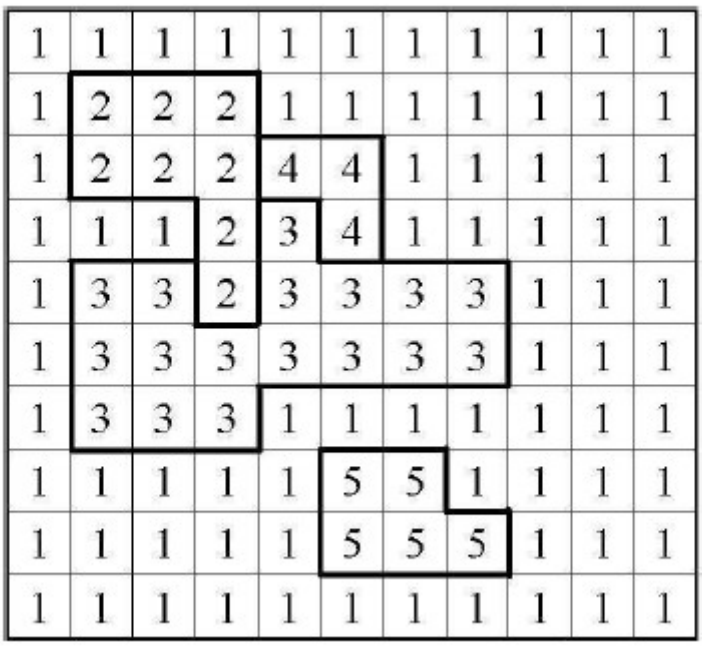

Fig.1.1 Generic cell configuration in CPM. Bold lines denote boundary of cells with given id. Source: [1]

Every such cell may be one entire part of simulated object, such as single soap bubble, biological cell or part of fluid. Potts model evolves by updating one such pixel, basing on a set of probabilistic rules. Hamiltonian of effective energy fnction is used to calculate the probability of accepting this random lattice update. Example algorithm proposed in [1] is following:

- pick a target lattice at random and one its random alien neighbor
- flip the target spin to its neighbor spin's value

- the probability that spin would produce a change in energy is calculated as follows:

$$P(\Delta H) = \begin{cases} \exp(-\Delta H/T) & \text{if } \Delta H > 0, \\ 1 & \text{otherwise,} \end{cases}$$

where T is a fluctuation temperature that controls the rate of acceptance. If T is large enough, almost all the moves are accepted (exp – deltaH/T is close to 1), otherwise for small values of T dynamics become deterministic and can trap within local minima.

2. Monte Carlo method applications

Monte Carlo methods have a wery broad range of applications. It was even used during performing simulations for the famous Manhattan project, although lack of good computational tools limited its possible application. Nowadays, main applications of the Monte Carlo method fall into one of the following main categories:

- Physics: physical chemistry, quantum chromodynamics – so-called Quantum Monte Carlo method may be applied here;
- Molecular modeling, including modeling of complex biological systems such as proteins;
- Weather forecasting;
- Engineering: process design, integrated circuit simulation, autonomous robotics, aerospace engineering, telecommunications;
- Applied statistics;
- Computer graphics, 3D rendering;
- Reliability engineering: calculating mean time between component failures;
- Artificial inteligence in video games – so-called Monte Carlo Tree Search Method is used;

2.1 Monte Carlo in scientific applications

Monte Carlo tree search

Monte Carlo tree search (MCTS) is a heuristic search algorithm of making optimal decisions. It is a combination of traditional Monte Carlo algorithm and tree searching. Basic MCTS algorithm is very simple:

- Build a search tree, according to the possible outcomes. Each node of such tree must include an estimated value based on simulation results and the number of times it has been visited;
- Start at its root node and select optimal child nodes until you reach a leaf node. Typically, node that maximises some quantity is chosen with each descending step;
- If the leaf node is not terminal (i.e it doesn't end the game), create one or more child nodes and select one of them. In the simplest implementation, MCTS adds one child node per iteration;
- Run a simulated playout from the child node until you achieve the result;
- Update the current sequence with the simulation result.

The algorithm is shown on Fig.2.1 in a form of state machine.

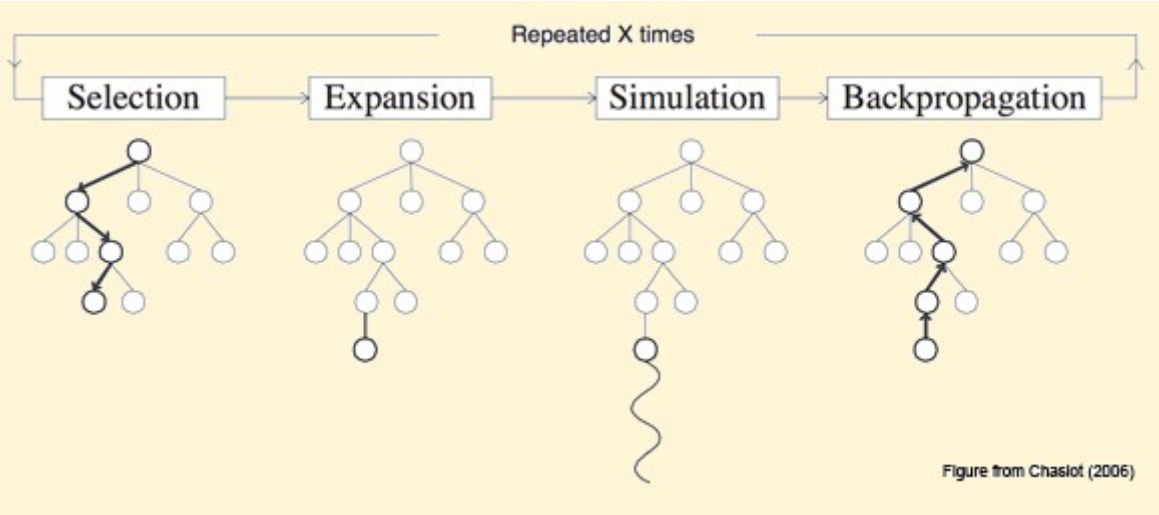

Fig. 2.1 Monte Carlo Tree Search method algorithm. Source: [2]

Quantum Monte Carlo

Quantum Monte Carlo method may be used to solve Schrodinger's equations for many interacting particles as well as to calculate accurately bulk properties of light elements such as hydrogen or helium or properties of isolated atoms and molecules made up from them. Here, Schodinger equation is represented by a random walk in multi-dimensional space in such a way that physical averages are exactly calculated [3]. Quantum Monte Carlo is a set of various Monte Carlo methods, of whom the most popular is Diffusion Monte Carlo.

Diffusion Monte Carlo

In this method, Schrodinger equation is written in imaginary time and it converges to the ground state exponentially fast. It can be demonstrated that all excited states decay exponentially fast with a decay constant given by the excitation energy from the ground state. Computer simulation of this many-bodies-problem is set up in the way shown on Fig. 2.2.

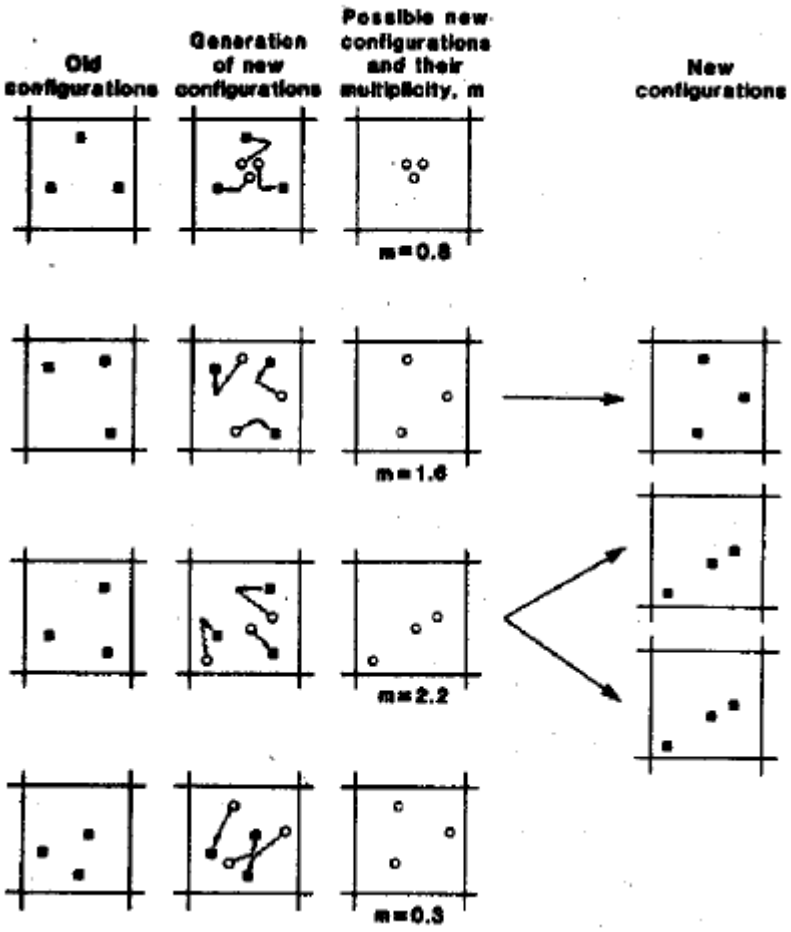

Fig.2.2 Diffusion Monte Carlo algorithm. Source: [3]

In this setup, an initial ensemble of systems is constructed by using a classical Monte Carlo calculation. Each ensemble consists of a number of so-called snapshots containing coordinates of all the particles. Sufficient number of such snapshots equals about 1000. Evolution is performed by considering each snapshot separately, displacing each of the snapshotted particles by some random value with a mean square displacemend proportional to the time step. Then a new ensemble is generated with a different number of snapshots, keeping in mind that for every evolving ensemble the trial energy must be adjusted with some kind of a feedback mechanism in order to keep the population stable. Snapshots are generated until steady state is reached – the samples generated in the last iteration are then samples of the ground-state wave function.

2.2 Monte Carlo Method in circuit analysis

It should be mentioned that MCM is commonly used in statistic analysis of integrated circuits. It is used to calculate response of a circuit with discrete components' parameters whose values are randomly varied within some specified ranges – i.e. typical 10kΩ resistor has actual value somewhere between 9900 and 10 100 Ω. This application is essential especially when filters are designed and every filter component may affect the desired filter response.
The following algorithm is used during Monte Carlo circuit analysis:

- Generate values of all the circuit components from a desired tolerance ranges (and a defined distribution if specified);
- Calculate the output signal values for such circuit, basing on i.e. Kirchoff's Voltage Law. Generally, every AC or DC method of analysis may be used if required;
- Repeat the analysis many times, each time for a different component values. The bigger the number of repetitions, the greater the probability that each component value within its all tolerance range will be considered during simulation.
- As a result, histograms for the statistical data are gathered.

Nowadays, many modern software tools such as OrCAD and Pspice use Monte Carlo Method to analyze circuit parameters, what the best possible proof of its popularity.

2.3 Monte Carlo Method in testing

Due to its simplicity and practical nature, stochastic Monte Carlo method is commonly used when testing or decision-making aid. It is a permutation test: all possible output values are calculated for observed data points. Its main advantage is that impact of each input factor's variation on the output data can be easily observed. This method doesn't require any extra Design For Test philosophy for simulation. By its usage it can be checked how much fluctuation of each input value can affect the final data. In [4], this method was considered for a braking system manufacturing process with

many complex input variable data, such as corner displacement, master cylinder flow rate and solenoid valve flow. Monte Carlo testing was applied to obtain the optimal combination of manufacturing parameters in order to optimize the performance of every produced unit. The general algorithm of all the tests making use of the method is naturally derived from aforementioned basic Monte Carlo method algorithm:

* Create a quantifical model of the phenomena that will be tested or supervised – be it physical, chemical, economical etc. Mathematical form of this process will be the transfer equation in further considerations;
* Define range and distribution of input parameters in your transfer equation. Each input must be considered here.
* Create a big amount of random input data for the transfer equation. Usually, about 100,000 input samples are required. **Sometimes, input data may not be independent from themselves and this should be also taken into account!**
* Simulate and gather results. In this step, it should be known which output values are acceptable.

A real-life example of using Monte Carlo method in simulation is shown on Fig.3.1 This Design of Experiment concerned process controls aid of field emission from carbon nanotubes.

Industry offers many applications faciliating Monte Carlo testing, such as Minitab [5], Apogee used mainly for simulating large production populations or @RISK, which is a tool using mainly Monte Carlo method in risk analysis and in Six Sigma Production Engineering [6]. Interesting application of Monte Carlo Simulation Test is shown in [7] where the simulation is performed on a robotic arm so that it can put 3D jigsaw puzzle together into a box in the shortest period possible. Example of mathematical model considering movement of the robotic arm is also presented there.

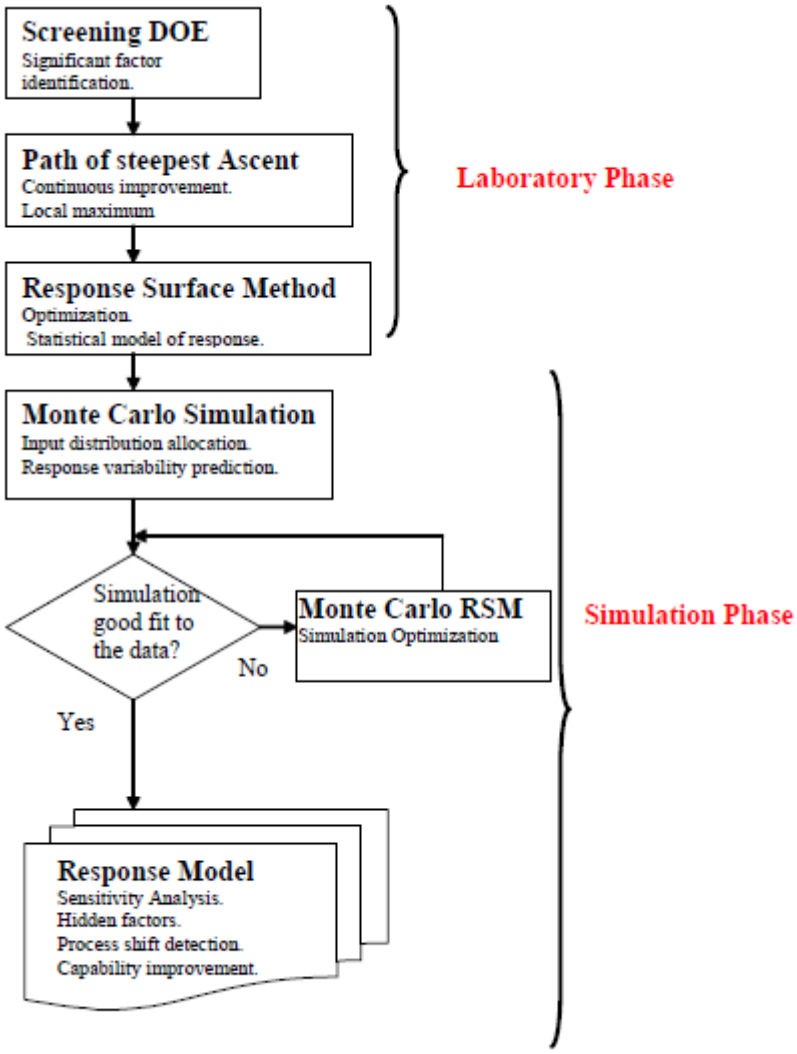

Fig. 3.1 Flowchart of the real experimental procedure making use of Monte Carlo RSM (Response Surface Methodology) method. Source: [8]

2.4 Monte Carlo filtering

A common example of filtering making use of Monte Carlo methods is an Ensemble Kalman filter, which is an approximation of the Kalman filter. By receiving probability density function of the modeled system and data probability (so-called Bayesian framework), probability density function after taking the data probability into account is calculated by performing a so-called Bayesian update by using a well-known Bayes theorem:

$$P(A|B) = P(B|A)\, P(A)/P(B)$$

This update is then merged with model advancing in time. In the original Kalman filter, covariance matrix is required to perform the update. However, maintaining all the covariance matrix in memory is not practical in use when working with complex many-dimensional systems.To get rid of the covariance matrix, a Monte Carlo approximation of an original Kalman filter was developed. In Ensemble Kalman filter, distribution of the system state is represented by ensemble, which is a collection of state vectors. Covariance matrix is replaced by sample covariance computed from the ensemble. Advancing the probability density function is then equal to advancing each member of such ensemble.

Sequential Monte Carlo method

Another way of using the Monte Carlo method in filtering is the Sequential Monte Carlo method, also known as particle filtering. This method can be used in very complex and versatile computational problems: real time analysis of dynamic systems [9] or in analysis of nonlinear and nonnormal stochastic processes [10]. It allows to approximate virtually any sequence of probability distributions, including eigenvalues of positive operators in physics or polymers simulation [11]. Still, Sequential Monte Carlo remains easy in its implementation and result analysis. Sequential Monte Carlo's state-space is shown in Fig. 3.2.

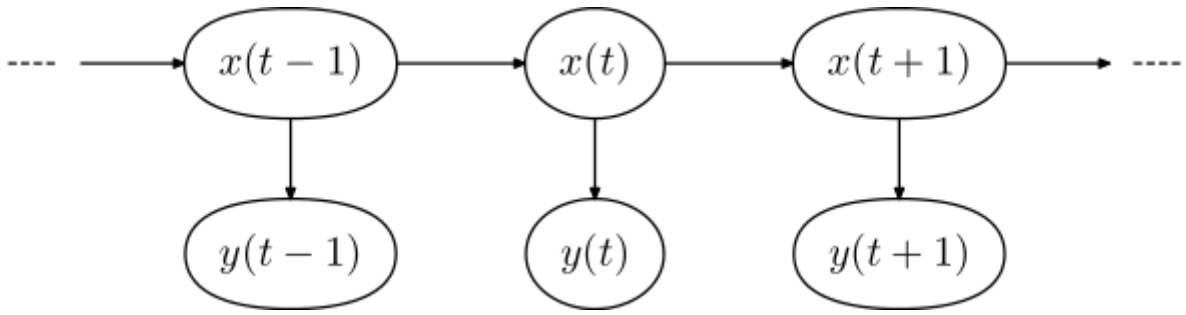

Fig.3.2 Sequential Monte Carlo method's state-space. X represents the hidden states, whereas y represents observation states.

Generally, the data model of Sequential Monte Carlo method is represented as

$$\begin{aligned} p(\mathbf{x}_0) & \\ p(\mathbf{x}_t|\mathbf{x}_{t-1}) & \quad \text{for } t \geq 1 \\ p(\mathbf{y}_t|\mathbf{x}_t) & \quad \text{for } t \geq 1. \end{aligned}$$

where

$$\mathbf{x}_{0:t} \triangleq \{\mathbf{x}_0, ..., \mathbf{x}_t\}$$
$$\mathbf{y}_{1:t} \triangleq \{\mathbf{y}_1, ..., \mathbf{y}_t\},$$

for x being the signal and y being the observation at the time t. The main goal is to estimate recursively in time the posteriori distribution:

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$$

as well as the expectation for some function of interest f(t):

$$I(f_t) = \mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}[f_t(\mathbf{x}_{0:t})] \triangleq \int f_t(\mathbf{x}_{0:t})\, p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})\, d\mathbf{x}_{0:t}$$

which can be a conditional mean or conditional covariance of xt. At any time, the posteriori distribution is given by the Bayes theorem:

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})\, p(\mathbf{x}_{0:t})}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})\, p(\mathbf{x}_{0:t})\, d\mathbf{x}_{0:t}}$$

When using a Sequential Monte Carlo, we obtain the following empirical estimate of (1):

$$P_N(d\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$$

where $dx_{0:t}$ stands for the Delta Dirac's mass located in $x_{0:t}$. Function of interest can be approximated in the following way:

$$I_N(f_t) = \int f_t(\mathbf{x}_{0:t})\, P_N(d\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} f_t\left(\mathbf{x}_{0:t}^{(i)}\right)$$

and from the strong law of numbers for high values of N we obtain

$$I_N(f_t) \underset{N\to+\infty}{\overset{a.s.}{\longrightarrow}} I(f_t)$$

As a result, one can estimate any quantity I (ft) from a set of N $x_{0:t}$ random samples. Rate convergence of this estimate is independent of the dimension of the integrand, whereas the

deterministic methods' rate of convergence decreases when integrand's dimension increases [9].

2.5 Numerical Monte Carlo methods

In numerical methods, Monte Carlo is commonly used to calculate expressions which are too complicated to solve them analytically. In example, random numbers from a complicated distribution can be calculated as presented in [12].

Calculating random numbers from a given distribution

When considering a complicated distriburion shown in Fig.3.3, the algorithm will be following:

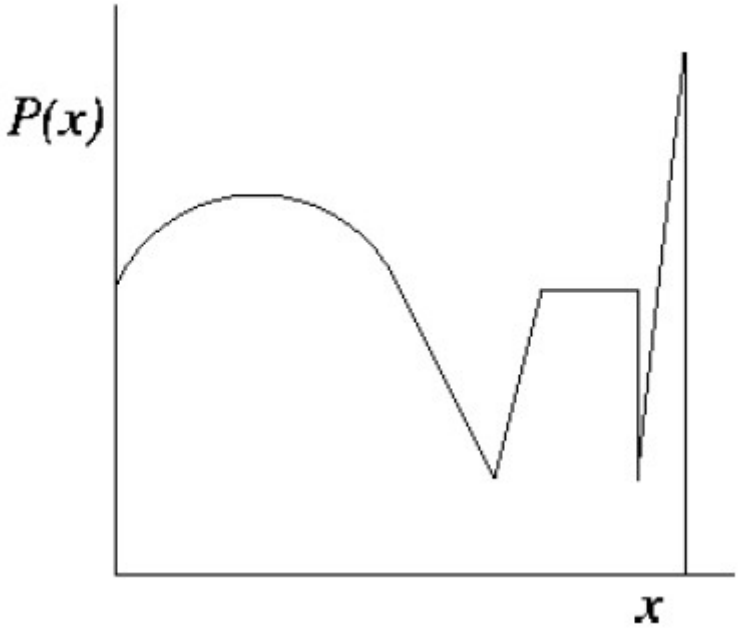

Fig. 3.3 Complex distribution used to calculate random numbers by using the numerical Monte Carlo method. Source: [13]

N. Pyka, *Monte Carlo Method*, Mathematical Statistics final raport

- Generate a set of x uniformly in its proper range;
- Generate a second random number – y;
- If y<P(x), keep and return the given value of x, else discard it.
- Repeat until number of kept x values will reach the required number.

Calculating pi by using Monte Carlo method

A common example of Monte Carlo numerical method's usage is calculation of pi value. A circle with R radius and center in (0,0) is considered. Its field is defined as inequity

$$x^2 + y^2 \leq R^2$$

To calculate pi value, we consider this inequity as some integral value and proceed with the following algorithm:

- Generate n random points from {x,y} subsets where x,y = [-1;1]. It can be seen that all these points belong to a square in which the circle is circumscribed.
- After generation of each n point, it is checked whether the point is inside or outside the circle;
- As a result, we obtain number of n random points m and number of n points that fell inside the circle k. Field of the circle is calculated as

$$P_k = \frac{P * k}{m}$$

where P stands for a field of the considered square. In this case, P is equal to 4.
For a large set of random points, obtained results will be close to the actual value of pi.

Example implementation of calculating pi in C++ written by the author is shown below. The

Code::Blocks IDE was used when running the program.

```
#include <iostream>
#include <cstdlib>
#define ITERATIONS 10000000
#define RADIUS 1
using namespace std;

int main()
{
    cout << "Calculating pi value for circle centered in (0,0) and
R=1:" << endl;
    int p=0;

    for (int i=0; i<ITERATIONS; i++)
    {
        float x=(rand()%2*RADIUS)-RADIUS; // random values from (-
1;1) range
        float y=(rand()%2*RADIUS)-RADIUS;

        if (x*x + y*y<=RADIUS) //here's the formula
            p++;
    }
          cout<<"Final   result   is:"   <<   (2*RADIUS*2*RADIUS)*p/
((float)ITERATIONS) <<endl;
return 0;
```

It is important to state that no matter how many iterations were performed, the pi value calculated in Code::Blocks was much closer to 3.1 instead of 3.14. Probably it is fault of a pseudo-random number generator used in this free-of-charge IDE.

Author states that it would be a very interesting experience to implement the Monte Carlo Method in Very High Speed Integrated Circuits Hardware Description Language (VHDL). Main advantage of such method would be a possibility to perform all the Monte Carlo iterations simultaneously what would drastically shorten its execution time. However, lack of available market solution with about 10 000 random number generators seems to be a problem when developing such solution.

Monte Carlo Integration

Monte Carlo Method may be used to solve complicated definite integrals which are either very hard or impossible to solve analytically. This method involves generating a random sequence of points from the given range and checking whether they are under or above the curve defined by a considered integral as presented in [14]. In such application, the following algorithm should be used to calculate the considered integrals:

Consider some known function $I = \int f(x)dx$ as shown on Fig. 3.4. This function is included within a rectangle whose size is defined by f_{max} (the y axis maximal value) and x_{max} (the x axis maximal value). For a simplification, we can now assume that f(x) and x minimal values equal 0.

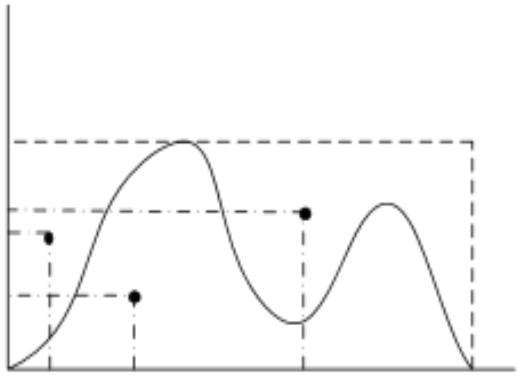

Fig.3.4 Generic function to be solved by using Monte Carlo Integration methods. Source: [14].

First of all, pair of random numbers $r_{1,} r_2$ from the proper range is generated. Then, $x_r = r_1 * x_{max}$ and $f_r = r_2 * f_{max}$ values are calculated and it is checked whether the point is under or above the curve. The point is located under the curve when $f_r \leq f(x_r)$. If the point is under the curve, the counter for points under the curve increments. This procedure is repeated many times (typically from 10 to 100 thousands).

The integral value equals

$$I = \frac{number\ of\ \ points\ under\ the\ curve}{number\ of\ trials} * f_{max} * x_{max}$$

This method of computing integrals was implemented in Octave to calculate the given integrals.

3. Monte Carlo methods implemented in Octave

The main task of this report was to implement the Monte Carlo Method in Octave in order to solve given numerical problems. For each solution, the function implementation was slightly different, therefore all the applied functions are presented.

3.1 Calculate value of π in Octave

The first task was to calculate exact value of the pi number. The following .m function was used to calculate pi value:

```
function [val,nk] = montecarlo (n)
nk=0;
```

```
for i=1:n
x = unifrnd(-1,1);
y = unifrnd(-1,1);

if (x*x + y*y <=1)
nk++;
endif

endfor
val=4*nk/n;
endfunction
```

Obtained pi value was close to 3,1415. When compared to the pi value obtained in C, value from Octave was much more precise, what shows that Octave's random number generator is more feasible in this application.

3.2 Solve the first definite integral

The following definite integral was to be solved:

$$\int_{-2}^{2} \exp(x + x^2)\, dx$$

Of course, it is possible to solve this problem analytically using substitution $w = e^x$. Analytical solution of this integral is equal to 93,1628.

The following function was implemented in Octave to calculate its value by using Monte Carlo Method:

```
function [val,nk] = montecarlo (n)

nk=0;

for i=1:n
x = unifrnd(-2,2);
```

```
#
# y - random variable generated within the possible range (minimal
# and maximal value of exp(x+x^2) within the given range.
#

y = unifrnd(0,exp(6));

if (y <=exp(x+x*x))
nk++;
endif

endfor
val=nk/n*4*exp(6);
endfunction
```

The following figure shows difference between the solution obtained in Octave and the analytical solution. This difference is shown in percents. Because of poor visibility, this and next figures are also given in Appendix.

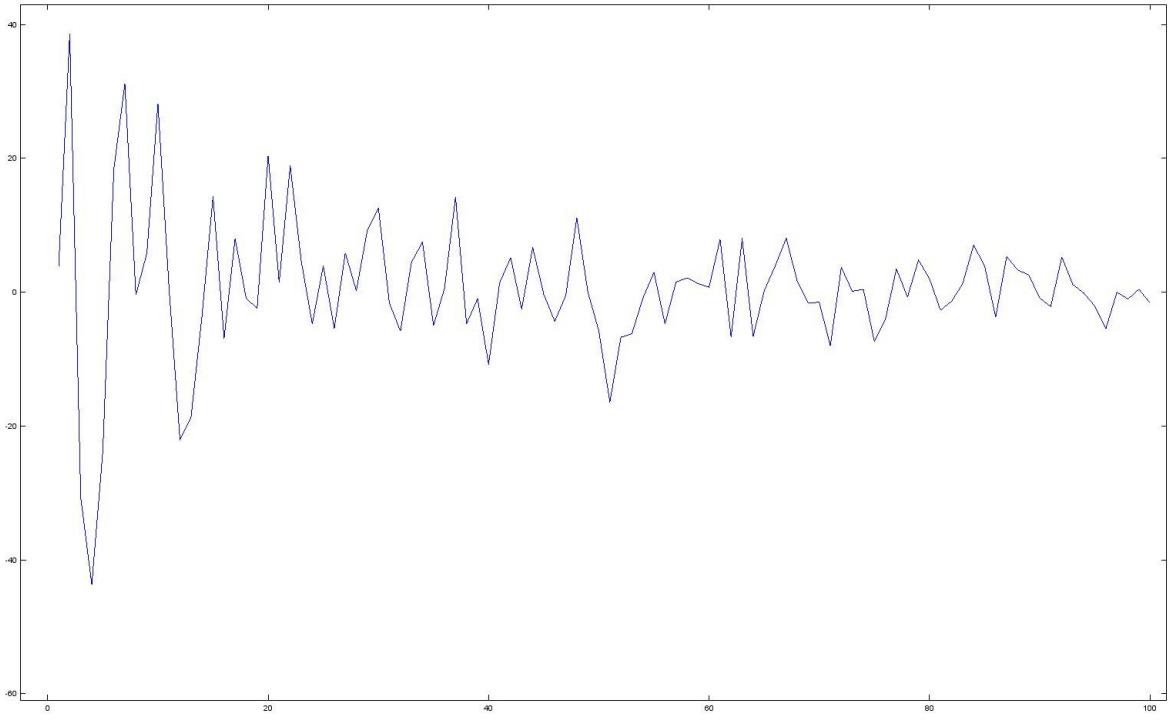

Fig. 3.5 Error function for first finite integral given in % versus number of iterations of

Monte Carlo Algorithm, ranging from 100 to 10000

3.3 Solve the second finite integral

The following definite integral was to be solved:

$$\int_0^2 \int_0^1 x\,y\,\exp\left(-\left(x^2\,y\right)\right) dx\,dy$$

The following function was implemented in Octave to calculate its value:

```
function [val,nk] = montecarlo (n)

nk=0;

for i=1:n

#
#z - matrix built of a plausible value of x (0;2), possible value
#of y (0;1) and range of possible values of x*y*exp(-x^2*y), which
#equals (0;0.42887).
#
z = [unifrnd(0,1), unifrnd(0,2), unifrnd(0,0.42887)];

if (z(3) <= z(1)*z(2)*exp(-z(1)*z(1)*z(2)))
nk++;
endif

endfor
val=nk/n*1*2*0.42877;
endfunction
```

Analytical solution of this integral is equal to 0.56778. The following figure shows difference between the solution obtained in Octave and the analytical solution. This difference is shown in percents:

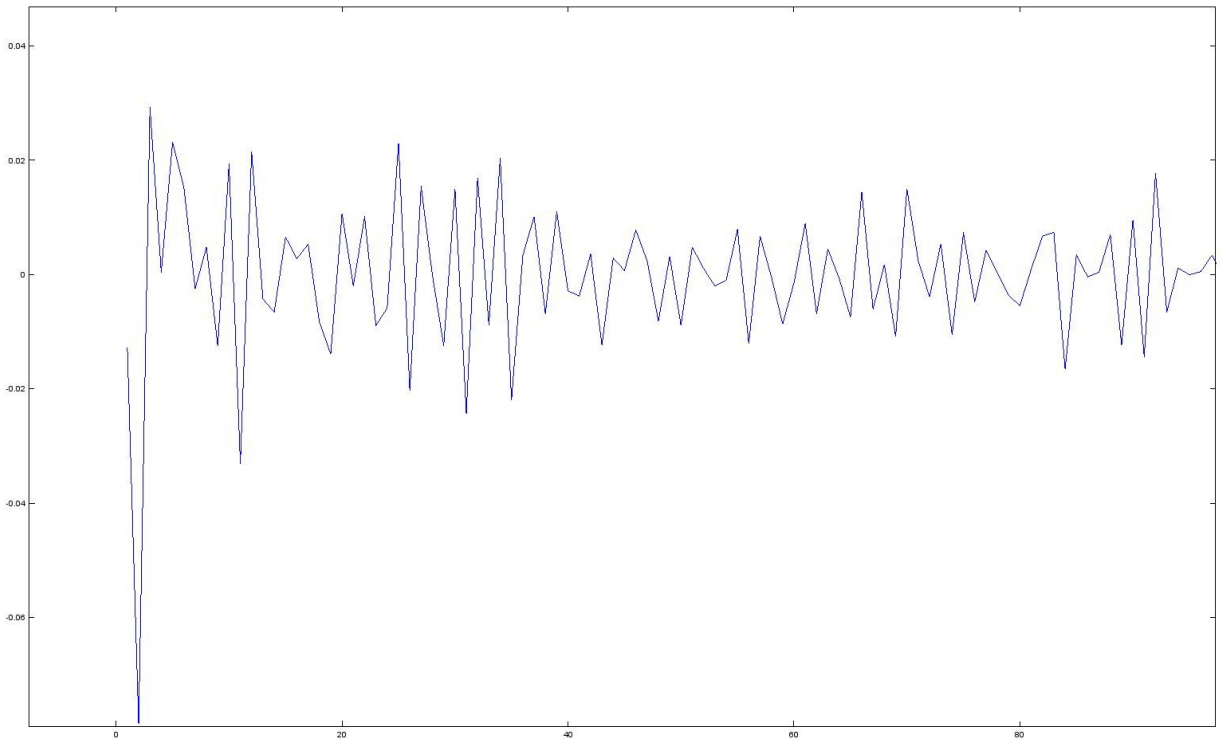

Fig. 3.6 Error function for second finite integral given in % versus number of iterations of Monte Carlo Algorithm, ranging from 100 to 10000

3.4 Solve the third finite integral

The following definite integral was to be solved:

$$\int\int \sin\sqrt{\ln(x+y+1)}, (x,y): (x-0.5)^2+(y-0.5)^2 \le 0.25$$

Here, author remarked that x and y variables form a 2-dimensional circle centered in (0.5, 0.5) and

radius equal to 0.5. Therefore, the following algorithm was used to cope with the problem:

- Choose x within its possible range. For a circle centered in (0.5,0.5) with radius equal to 0.5, this range equals (0;1).
- Choose y value within its possible range for a given x value, so that it is located within the circle;

The following function was implemented to calculate integral's value:

```
function [val,nk] = montecarlo (n)

nk=0;

for i=1:n

x=unifrnd(0,1);
y=unifrnd((-sqrt(0.25-(x-0.5)^2)+0.5),(sqrt(0.25-(x-0.5)^2)+0.5));
z=unifrnd(0,1);

if (z <= sin(sqrt(log(x+y+1))))
nk++;
endif

endfor
val=nk/n*3.1415*0.5*0.5*1;
endfunction
```

Value of such function was too complex to calculate it analytically. However, author was pretty clever and made the approximation based on calculating the integral with x and y values ranging from 0 to 1. Code in Octave was slightly altered – value of y and val variables was now chosen in the following way:

```
y=unifrnd(0,1);
% …
%...
```

```
val=nk/n;% to be strict - *1*1, because of the range of x and y;
```

in such case, analytical solution was equal to 0.71472. The following figure shows difference between the solution obtained in Octave and the analytical solution (in percents):

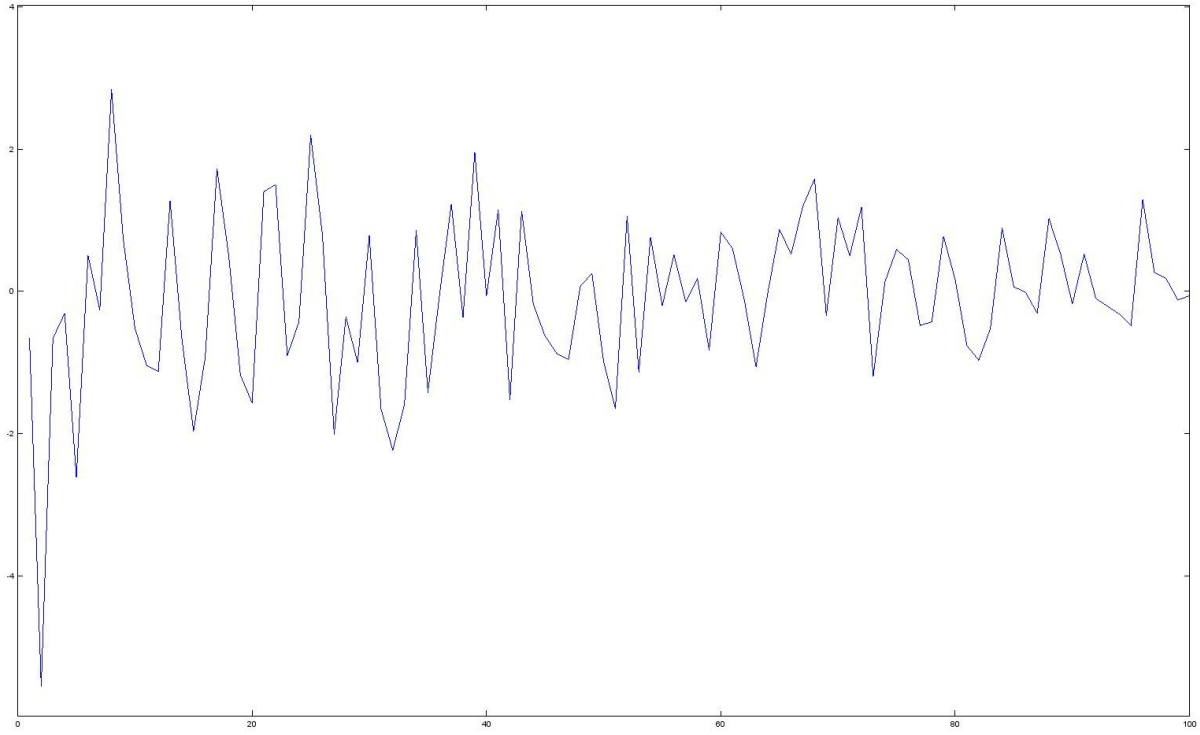

Fig. 3.7 Error function for third finite integral given in % versus number of iterations of Monte Carlo Algorithm, ranging from 100 to 10000

3.5 Solve the fourth finite integral

The following definite integral was to be solved:

$$\int_0^1 f(x)dx,\ f(x)=\exp\left(-0.5\mathrm{x}\,\Sigma^{-1}x^T\right),\ \Sigma=\begin{bmatrix}3 & 1 & -1\\ 1 & 2 & -0.5\\ -1 & -0.5 & 1\end{bmatrix}$$

Here, x should be a matrix with values ranging from 0 to 1.
The following function was used to calculate its value:

```
function [val,nk] = montecarlo (n)

nk=0;
E= [3,1,-1;1,2,-0.5;-1,-0.5,1];

for i=1:n

x=unifrnd(0,1,[3,1]);
z=unifrnd(0,e);

if (z <= exp(-0.5*x'*inv(E)*x))
nk++;
endif

endfor
val=nk/n*1*e;
endfunction
```

Because the E matrix was positive-definite, product of 3 Gaussian mixtures with various σ values (respectively for x1, x2 and x3 fields) was obtained as a result. Product of these mixtures was equal to about 0.64. The following figure shows difference between the solution obtained in Octave and the analytical solution (in percents):

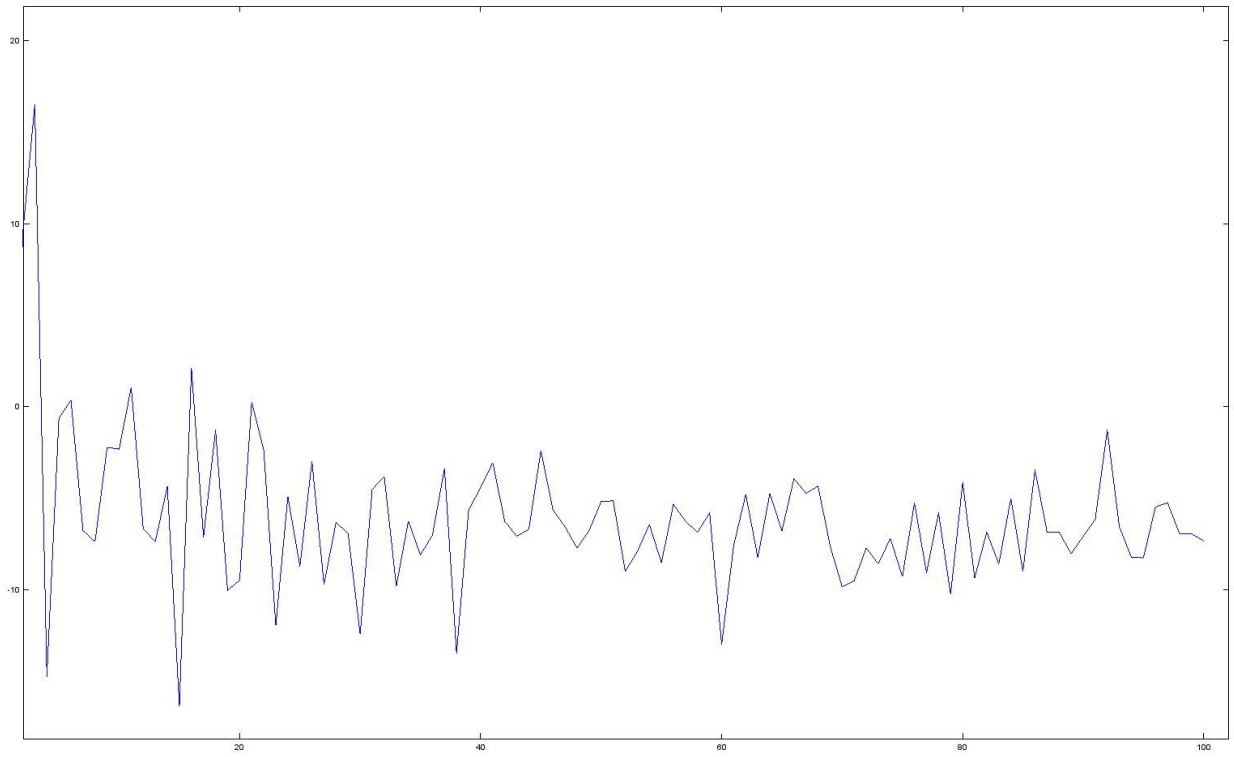

Fig. 3.8 Error function for fourth finite integral given in % versus number of iterations of Monte Carlo Algorithm, ranging from 100 to 10000

4. Conclusions

As a conclusion for this report, it should be pointed out that the Monte Carlo Method provides an useful and versatile tool usable in many engineering applications: both in integrated circuit testing and in quantum physics. Personally, author didn't encounter any other algorithm which would be so popular and widespread. Probably it is a simplicity of the main Monte Carlo algorithm that warranties its wide usage.
The only severe limitation is the need to use a good random numbers generator. This may be a blocker when trying to implement the method in some environments – sometimes, such generator should be generated anew what would cause additional costs.

In numerical applications, results obtained by using the Monte Carlo Method were close to the results obtained by deterministic methods in Wolphram Alpha. Unfortunately, lack of free software tools able to compute values of last two integrals was a problem when comparing the results. To make matters worse, there was no tool able to compare execution time of calculating the integers by using some deterministic methods and the Monte Carlo Method. Author thinks that no matter what the integral is, Monte Carlo Method provides a known computational time depending on number of its iterations and in some applications it could be its great advantage when comparing to the deterministic solutions whose computational time may vary a lot, depending on the input integral.

N. Pyka, *Monte Carlo Method*, Mathematical Statistics final raport

5. References

[1] Debasis Dan et al., *Solving the Advection-Diffusion Equations in Biological Contexts using the Cellular Potts Model*, Biocomplexity Institute and Department of Physics;

[2] Monte Carlo Tree Search tutorial by EPSRC EP/I001964/1 grant, Imperial College London, http://mcts.ai/about/index.html ;

[3] David Ceperley, Berni Alder, *Quantum Monte Carlo*, Science, vol. 231, February 1986, pp. 555-560;

[4] Statease leaflet, *Optimizing Product Design While Taking Manufacturing Variability into Account*, available at http://www.statease.com/pubs/trw_automotive.pdf ;

[5] Paul Sheehy, Eston Martz, *Doing Monte Carlo Simulation in Minitab Statistical Software*, ASQ Lean Six Sigma Conference, February 2012;

[6] @Risk product main page, http://www.palisade.com/risk/ ;

[7] Bastian Solutions, *The Sensitive Engineer: Using Monte Carlo Simulation to Understand the Sensivity of a Complex System*, http://www.bastiansolutions.com/blog/index.php/2011/11/23/the-sensitive-engineer-using-monte-carlo-simulation-to-understand-the-sensitivity-of-a-complex-system/#.UqQjHsSLKSo ;

[8] Dirk Jordan, *Monte Carlo Simulation as Process Control Aid*, Proceedings of the 2007 Crystal Ball User Conference;

[9] Jun S. Liu, Rong Chen, *Sequential Monte Carlo Methods for Dynamic Systems*, Journal of the American Statistical Association, Vol. 93, No. 443 (Sep., 1998), pp. 1032- 1044;

N. Pyka, *Monte Carlo Method*, Mathematical Statistics final raport

[10] Jesús Fernández-Villaverde, *Sequential Monte Carlo Filtering:an Example*, University of Pennsylvania, Juan F. Rubio-Ramírez, Federal Reserve Bank of Atlanta, January 5, 2004;

[11] A. Doucet, *Introduction to Sequential Monte Carlo Methods*, materials from lecture, http://carbon.videolectures.net/2007/pascal/mlss07_tuebingen/doucet_arnaud/mlss07_doucet_smcm_01.pdf ;

[12] A. Doucet et al., *An Introduction to Sequential Monte Carlo methods*, ch.1, p.6;

[13] Scott Oser, Monte Carlo and Numerical Methods, materials from lecture, available at http://www.phas.ubc.ca/~oser/p509/Lec_04.pdf ;

[14] Osman/EECS/WSU EE351, *Numerical Integration Using Monte Carlo Method,* materials from lecture, available at http://www.eecs.wsu.edu/~osman/EE351_S06/montecarlo.pdf.