The Monte Carlo method and some applications

Ragaa Ahmed

Department of Mathematics
Instituto Superior Técnico

LisMath Seminar, IST Lisbon, 27th May 2016

Outline I

Outline II

Purpose of the seminar

- To explain the basic philosophy of Monte Carlo methods and suggest how they can be used to solve various problems.
- To show how random games (Monte Carlo methods) can be designed whose outcomes approximate solutions to differential equations.

Monte Carlo Methods (an Introduction)

Definition

- Is a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.
- Using in physical and mathematical problems and are most useful when it is difficult or impossible to use other mathematical methods.
- Is mainly used in three distinct problem classes: optimization, numerical integration, and generating draws from a probability distribution.

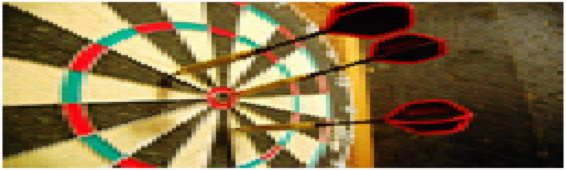
A breif history

- Courant, Friedrichs, and Lewy: Their pivotal 1928 paper has probabilistic interpretations and MC algorithms for linear elliptic and parabolic problems.
- Fermi/von Neumann: Use Monte Carlo in the calculation of neutron diffusion 1930.
- Ulam: He realised that computers could be used to solve such problems 1940.
- Many papers on Monte Carlo simulation appeared in physics literature 1950. The first major paper was published by Metropolis et al in 1953.

- Generalisation of the Metropolis algorithm by Hastings which led to development of MC 1970.
- Important papers appeared in the fields of computer vision and artificial intelligence but there were few significant publications in the field of statistics 1980.
- MC made the first significant impact in statistics in the work of Gelfand and Smith.
- Kac and Donsker: Used large deviation calculations to estimate eigenvalues of a linear Schrodinger equation.
- Forsythe and Leibler: Derived a MCM for solving special linear systems related to discrete elliptic PDE problems.

The basic philosophy

- The basic idea here is that games of chance (like throwing darts,...) can be played (generally on a computer) whose outcomes approximate solutions to real-word problems.
- First of all Monte Carlo methods are procedures for solving nonprobabilistic-type problems (problems whose outcome does not depend on chance) by probabilistic-type methods (methods whose outcome depends on chance).



The general philosophy of Monte-Carlo methods

Probabilistic game

The outcome of the game $\hat{p}$

(Like the fraction of heads in tossing a coin, throwing darts, and so forth)

Deterministic problem

The answer to the problem is p

(Like evaluation an integral, solving a PDE, and so forth)

$$\text{Outcome} = \hat{p} \xrightarrow[\text{Approximation}]{\qquad\qquad} \text{Answer} = p$$

Why Monte Carlo method?!!

- A standard Monte Carlo method provides an approximation at a given point without evaluating the values at other points.
- The PDE methods where some stability conditions may be required (like the Courant-Friedrichs-Lewy condition), the above Monte Carlo method does not require any extra condition to converge: it is unconditionally convergent.
- Since a Monte Carlo method provides random evaluations of E(X), different program runs will give different results (as a difference with a deterministic method which systematically has the same output).

- The memory required to run a PDE algorithm increases exponentially with the dimension, as a difference with a Monte Carlo approach.
- It is commonly admitted that a PDE approach is more suitable and efficient in dimension 1 and 2, whereas a Monte Carlo procedure is more adapted for higher dimensions.
- On the other hand, a PDE-based method computes a global approximation of u (at any point (t, x)), while a Monte Carlo scheme gives a pointwise approximation only.

The approach is useful in these areas of application whenever:

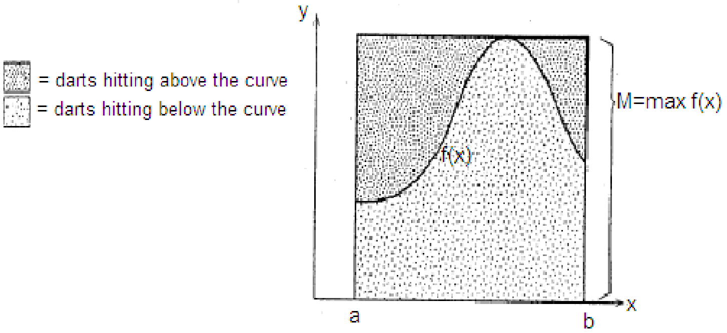
1. There is no other analytical or numerical solution to the problem,
2. Checking the validity of new stochastic, numerical or analytical methods,
3. Developing models for complex processes, and checking them against experimental values,
4. Monte Carlo can offer in some cases a faster approach than other methods such as finite differences, particularly in multidimensional problems.

Evaluating an integral

- To illustrate the method, suppose we want to evaluate the integral
$$I = \int_a^b f(x)dx$$
(a nonprobabilistic problem).
- To use the Monte Carlo method, we would devise a game of chance whose outcome is the value of the integral (or approximates the integral).
- There are, of course, many games that we could devise; the actual game we use would depend on the accuracy of the approximation, simplicity of the game, and so on.

- An obvious game to evaluate the integral would be throwing darts at the rectangle

$$R = \{(x, y) : a \leq X \leq b, 0 \leq y \leq maxf(x)\}$$


= darts hitting above the curve
= darts hitting below the curve
y
x
f(x)
M=max f(x)
a
b

- Its fairly obvious that if we randomly toss 100 or so darts at the rectangle R enclosing the graph,
- Hence, our outcome of the game

$$\hat{I}=[\text{fraction of tosses under } f(x)]\times(\text{area of R})$$

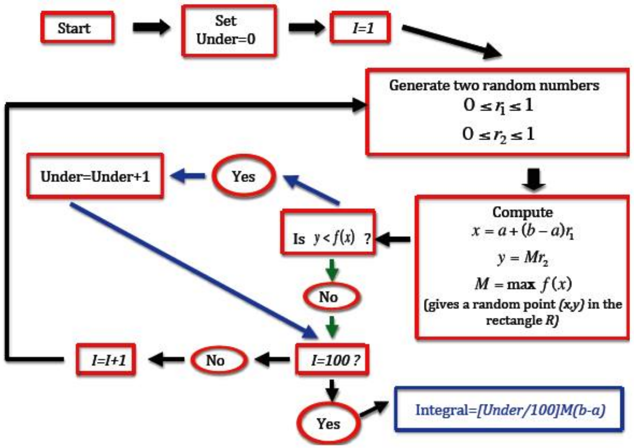
is used to estimate the true value of the integral I.

- Its fairly obvious that if we randomly toss 100 or so darts at the rectangle R enclosing the graph,
- Hence, our outcome of the game

$$\hat{I} = [\text{fraction of tosses under } f(x)] \times (\text{area of R})$$

is used to estimate the true value of the integral I.

- To carry out the actual computation on a computer, we would have to generate the **sequence of random points** in some way and have the computer play the dart tossing game.
- Assuming for the time that we have **a sequence of random points** then the flow diagram of the problem is as follows

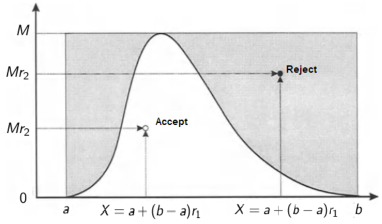


Flow diagram to evaluate $\int_a^b f(x)dx$ by the Monte Carlo method (100 tosses)

Random Numbers

Everything comes down to the question, how do we generate a sequence of random numbers $\{r_i; i = 1, 2, ...\}$ uniformly distributed in $[0; 1]$ to compute a random number x_i inside $[a; b]$

$$x_i = a + (b - a)r_i$$



Residue algorithm for generating random numbers

To generate a sequence of random integers (between 0 and P), we use the residue algorithm.

1. Pick the first random integer any way you like between 0 and P (P was picked in advance).
2. Multiply this random integer by some fixed integer M (picked in advance).
3. Add to that product another fixed integer K (picked in advance).
4. Divide the resulting sum by P and pick the remainder as the new random integer. Now go back to step 2 and repeat steps 2-4 until you have enough random integers.

- This residue algorithm can be written as

$$r_{i+1} = (Mr_i + k) modP, i = 0, 1, 2, ...$$

which says, if we are given a random integer r_i , then to compute a new one r_{i+1}, we multiply by M, add K, divide by P, and pick the remainder.

Remarks

- If we choose, for example, $P = 100$ in our random number generator, the remainders will be one of the integers $0, 1, 2, \ldots, 99$, and, hence, our entire process will start repeating before long.
- In fact, our random numbers might be

$$15, 71, 43, 7, 43, 7, 43, 7, \text{(Cycle of two numbers)}$$

and, hence, our method is not good.

- It can be proven mathematically that if the numbers M, K, and P are chosen according to certain rules, then no matter how we pick the first random number r_0, the algorithm will generate the entire residue class.

- So, if we pick P very large (like 2^{40}), we are assured that (for practical purposes) the process will never repeat.

Monte Carlo solution of PDEs

The linear second order elliptic partial differential equation

$$L[u] = F(x, y) \tag{1}$$

with the operator $L[.]$ defined by

$$L[u] = Au_{xx} + 2Bu_{xy} + Cu_{yy} + Du_x + Eu_y,$$

where $\{A, B, C, D, E\}$ are all functions of (x, y). The operator L[.] may be discretized to yield the approximation:

$$\begin{aligned} L[u] \simeq & A_{i,j}\ [\frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{(\Delta x)^2}] \\ & + 2B_{i,j}\ [\frac{v_{i+1,j+1} - v_{i,j+1} - v_{i+1,j} + v_{i,j}}{(\Delta x)(\Delta y)}] \\ & + C_{i,j}\ [\frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{(\Delta y)^2}] + D_{i,j}\ [\frac{v_{i+1,j} - v_{i,j}}{\Delta x}] \\ & + E_{i,j}\ [\frac{v_{i,j+1} - v_{i,j}}{\Delta y}] \end{aligned} \tag{2}$$

where $x_i = x_0 + i(\Delta x)$, $y_j = y_0 + j(\Delta y)$, $v_{i,j} = u(x_i, y_j)$, and a subscript of i, j means an evaluation at the point (x_i, y_j).

If the $\{\Gamma_{.,.}\}$ and $Q_{i,j}$ are defined by

$$
\begin{aligned}
\Gamma_{i+1,j+1} &= [\frac{2B_{i,j}}{(\Delta x)(\Delta y)}],\\
\Gamma_{i+1,j} &= [\frac{A_{i,j}}{(\Delta x)^2} - \frac{2B_{i,j}}{(\Delta x)(\Delta y)} + \frac{D_{i,j}}{\Delta x}],\\
\Gamma_{i,j+1} &= [\frac{C_{i,j}}{(\Delta y)^2} - \frac{2B_{i,j}}{(\Delta x)(\Delta y)} + \frac{E_{i,j}}{\Delta y}],\\
\Gamma_{i-1,j} &= [\frac{A_{i,j}}{(\Delta x)^2}],\\
\Gamma_{i,j-1} &= [\frac{C_{i,j}}{(\Delta x)^2}],
\end{aligned}
$$

$$Q_{i,j} = [\frac{2A_{i,j}}{(\Delta x)^2} - \frac{2B_{i,j}}{(\Delta x)(\Delta y)} + \frac{2C_{i,j}}{(\Delta y)^2} + \frac{D_{i,j}}{\Delta x} + \frac{E_{i,j}}{\Delta y}],$$

then, equation (1) may approximated as

$$\begin{aligned}Q_{i,j}v_{i,j} =& \Gamma_{i+1,j}v_{i+1,j} + \Gamma_{i+1,j+1}v_{i+1,j+1} + \Gamma_{i,j+1}v_{i,j+1} \\ &+ \Gamma_{i-1,j}v_{i-1,j} + \Gamma_{i,j-1}v_{i,j-1} - F_{i,j}.\end{aligned}$$

Dividing through by $Q_{i,j}$ and defining $p_{i,j} = \dfrac{\Gamma_{i,j}}{Q_{i,j}}$ we have,

$$\begin{aligned}v_{i,j} =& p_{i+1,j}v_{i+1,j} + p_{i+1,j+1}v_{i+1,j+1} + p_{i,j+1}v_{i,j+1} \\ &+ p_{i-1,j}v_{i-1,j} + p_{i,j-1}v_{i,j-1} - \frac{F_{i,j}}{Q_{i,j}}.\end{aligned} \tag{3}$$

Since the operator $L[.]$ has been presumed to be elliptic, then Δx and Δy may be chosen small enough so that each of the p's are positive.

We interpret p's as the probabilities of taking a step in a specified direction. Specifically, for equation (3), if a particle is at position (i, j) at step N, then,

- With probability $p_{i,j+1}$, the particle goes to $(i, j+1)$ at step $N+1$.
- With probability $p_{i,j-1}$, the particle goes to $(i, j-1)$ at step $N+1$.
- With probability $p_{i+1,j}$, the particle goes to $(i+1, j)$ at step $N+1$.
- With probability $p_{i-1,j}$, the particle goes to $(i-1, j)$ at step $N+1$.
- With probability $p_{i+1,j+1}$, the particle goes to $(i+1, j+1)$ at step $N+1$.

- Suppose a particle starts at the point $P_0 = z$ and undergoes arandom walk according to the above prescription.
- After, say, m steps it will hit the boundary where the sequence of points that this particle visits is $\{P_0, P_1, P_2, ..., P_m\}$.
- Then, an unbiased estimator of the value of $u(z)$ for the following elliptic problem:

$$L[u] = F(x,y), \qquad \textit{for all points } x,y \textit{ in the domain } R,$$

$$u = \phi(x,y), \qquad \textit{for all points } x,y \textit{ on the boundary } \partial R,$$

is given by

$$u(z) \simeq \phi(P_m) - \sum_{j=0}^{m} \frac{F(P_j)}{Q(P_j)}$$

In practice, several random paths will be taken, and the average taken to estimate $u(z)$. That is,

$$u(z) \simeq \frac{1}{K} \sum_{k=1}^{K} \{\phi(P_{m_k}^k) - \sum_{j=0}^{m_k} \frac{F(P_j^k)}{Q(P_j^k)}\},$$

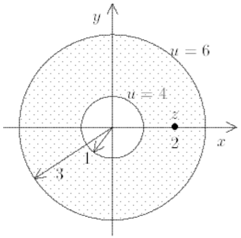
where $(P_0^k, P_1^k, ...P_{m_k}^k)$, represent the path taken by kth random particle.

Examples of Laplace equation

Example

$$PDE : \Delta u = 0 \qquad 0 < r < 3, \quad 0 < \theta < 2\pi$$

$BC : u(1,\theta) = 4, u(3,\theta) = 6$



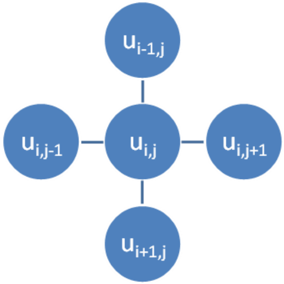
- We will approximate the value of $u(z)$, when $z = (r = 2; \theta = 0)$.
- The exact solution for this problem is $u(r) = 4 + 2 log r / log 3$, so that $u(z) = 4 + log 2 / log 3 \simeq 5.261$.
- To approximate the solution to this problem numerically, We will use the rectangular variables x and y, rather than the polar coordinate variables r and θ.
- Using a standard second order approximation to the Laplacian, we find

$$\Delta u \simeq \frac{v_{i+1,j} + v_{i-1,j} + v_{i,j-1} - 4v_{i,j}}{h^2} = 0$$

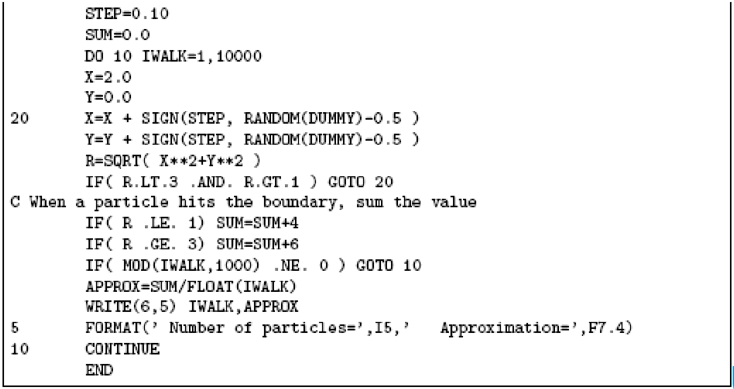
where $v_{i,j} = u(x_i, y_j), \quad h << 1.$

Then we have

$$v_{i,j} = \frac{v_{i+1,j}}{4} + \frac{v_{i-1,j}}{4} + \frac{v_{i,j+1}}{4} + \frac{v_{i,j-1}}{4}$$

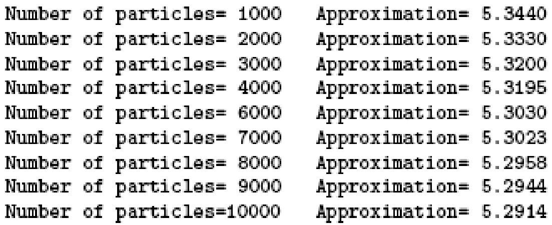


Fortran programm for Monte Carlo method applied to elliptic equations: (Zwillinger, 1997)



```
        STEP=0.10
        SUM=0.0
        DO 10 IWALK=1,10000
        X=2.0
        Y=0.0
20      X=X + SIGN(STEP, RANDOM(DUMMY)-0.5 )
        Y=Y + SIGN(STEP, RANDOM(DUMMY)-0.5 )
        R=SQRT( X**2+Y**2 )
        IF( R.LT.3 .AND. R.GT.1 ) GOTO 20
C When a particle hits the boundary, sum the value
        IF( R .LE. 1) SUM=SUM+4
        IF( R .GE. 3) SUM=SUM+6
        IF( MOD(IWALK,1000) .NE. 0 ) GOTO 10
        APPROX=SUM/FLOAT(IWALK)
        WRITE(6,5) IWALK,APPROX
5       FORMAT(' Number of particles=',I5,'   Approximation=',F7.4)
10      CONTINUE
        END
```

This code used to simulate the motion of the particles according to the above probability law. That output of that program is given below for $u(r = 2, \theta = 0)$. As more more points are taken, the approximation becomes better.



```
Number of particles= 1000   Approximation= 5.3440
Number of particles= 2000   Approximation= 5.3330
Number of particles= 3000   Approximation= 5.3200
Number of particles= 4000   Approximation= 5.3195
Number of particles= 6000   Approximation= 5.3030
Number of particles= 7000   Approximation= 5.3023
Number of particles= 8000   Approximation= 5.2958
Number of particles= 9000   Approximation= 5.2944
Number of particles=10000   Approximation= 5.2914
```

Example

To find a function $u(x, y)$ that satisfies

$$PDE : \Delta u = 0, \qquad 0 < x < 1, \;\; 0 < y < 1$$

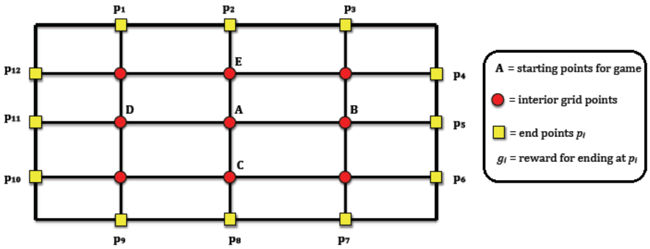
$$BC : u(x,y) = g(x,y) = \begin{cases} 1; & \text{On the top of the square} \\ 0; & \text{On the sides and bottom} \\ & \quad\text{of the square} \end{cases}$$

To illustrate the Monte Carlo method in this problem, we introduce a game called tour du wino.

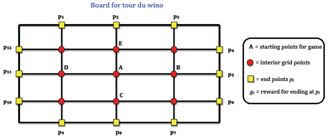
How Tour du Wino is played?

1. The wino starts from an arbitrary point (point A in our case).
2. At each stage of the game, the wino staggers off randomly to one of the four neighboring points.The probability of going to each of these neighbors is 1/4.

Board for tour du wino



3. After arriving at a neighboring point, the wino continues this process wandering from point to point until eventually hitting a boundary point p_i . He then stops, and we record that point p_i. This completes one random walk.
4. We repeat steps 1-3 until many random walks are completed. We now compute the fraction of times the wino had ended up at each of the boundary points p_i.



Suppose the wino receives a reward g_i and if he ends his walk at the boundary point p_i, then the average reward for all this walks is

Probability of random walk ending at p_i [Stanley(1982)]

Boundary point p_i	$P_A(p_i)$ = fraction of times the wino ends at p_i	g_i = reward for ending at p_i
1	0.04	1
2	0.15	1
3	0.03	1
4	0.06	0
5	0.17	0
6	0.05	0
7	0.06	0
8	0.15	0
9	0.03	0
10	0.06	0
11	0.16	0
12	0.04	0

$$\begin{aligned} R(A) &= g_1 P_A(p_1) + g_2 P_A(p_2) + \dots + g_{12} P_A(p_{12}) \\ &= 1(.04) + 1(.15) + 1(.03) + 0(.06) + \dots + 0(.04) \\ &= 0.22. \end{aligned}$$

The game is completed with the determination of $R(A)$.

Remarks

It turns out that the average reward is the approximate solution to our Dirichlet problem at Point A. This interesting observation is based on two facts:

1. Suppose the wino started at a point A that was on the boundary of the square. Each resulting random walk ends immediately at that point, and the wino collects the amount g_i. Thus, his average reward for starting from a boundary point is also g_i.

2. Now suppose the wino starts from an interior point. Then, the average reward R(A) is clearly the average of the four average rewards of the four neighbors

$$R(A) = 1/4[R(B) + R(C) + R(D) + R(E)]$$

- We have seen that R(A) satisfies two equations

$$\begin{cases} R(A) = 1/4(R(B) + R(c) + R(D) + R(E)) & \text{(A an interior point)} \\ R(A) = g_i & \text{(A a boundary point)} \end{cases}$$

- If we let g_i be the value of the boundary function $g(x, y)$ at the boundary point p_i, then our two equations are exactly the two equations we arrived at when we solved the Dirichlet problem by the finite-difference method.

TÉCNICO LISBOA

- That is, $R(A)$ corresponds to $u_{i,j}$ in the finite-difference equations

$$\begin{cases} u_{i,j} = \dfrac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}}{4} & ;(i,j) \text{ an interior point} \\ u_{i,j} = g_{i,j} & ; g_{i,j} \text{ the solution at a boundary point} \end{cases}$$

- Hence, $R(A)$ will approximate the true solution of the PDE at A.

These rules give the solution at one point inside the square:

1. Generate several random walks starting at some specific point A and ending once you hit a boundary point. Keep track of how many times you hit each boundary point.
2. After completing the walks, compute the fraction of times you have ended at each point p_i. Call these fractions $P_A(p_i)$.
3. Compute the approximate solution u(A) from the formula

$$u(A) = g_1 P_A(p_1) + g_2 P_A(p_2) + \dots + g_N P_A(p_N)$$

where g_i is the value of the function at p_i and N is the number of boundary points.

Example

To find a function $u(x, y)$ that satisfies

$$PDE : u_{xx} + (sinx)u_{yy} = 0, \qquad 0 < x < \pi, \quad 0 < y < \pi$$

BC: $u(x, y) = g(x, y)$ On the boundary of the square

- To solve this example, we replace u_{xx} , u_{yy} and sin x by

$$u_{xx} = [u_{i,j+1} - 2u_{i,j} + u_{i,j-1}]/h^2, \;\; u_{yy} = [u_{i+1,j} - 2u_{i,j} + u_{i-1,j}]/k^2$$

$$sinx = sinx_j.$$

Then plug them into the PDE we have

$$u_{i,j} = \frac{u_{i,j+1} + u_{i,j-1} + sinx_j(u_{i+1,j} + u_{i-1,j})}{2(1 + sinx_j)}$$

- In other words, if the wino is at the point (i, j), he then goes to the point:

$$(i, j+1) \text{ with probability } \frac{1}{2(1+sinx_j)}$$
$$(i, j-1) \text{ with probability } \frac{1}{2(1+sinx_j)}$$
$$(i+1, j) \text{ with probability } \frac{sinx_j}{2(1+sinx_j)}$$
$$(i-1, j) \text{ with probability } \frac{sinx_j}{2(1+sinx_j)}$$

- Other than this slight modification, the game is exactly the same as before.

Some Physical Application

- In fact, Monte Carlo methods were originally developed to study difficult **neutron-diffusion problems** that were impossible to solve analytically.
- The Monte Carlo method has been applied to conductive and radiative heat transport. Its application to convective heat transport has been minimal despite the fact that energy transport in turbulent flows depends primarily on random processes.

Monte Carlo simulations can be used a wide range of conduction problems:

1. Steady state conduction,
2. Transient conduction,
3. Various geometrical configurations,
4. Different boundary conditions including radiative and convective heat transport with volumetric sources.
5. Anisotropic and nonhomogeneous media.

References I

Kroese, D. P.; Brereton, T.; Taimre, T.; Botev, Z. I."Why the Monte Carlo method is so important today". WIREs Comput Stat 6: 386–392 (2014). doi:10.1002/wics.1314.

Emmanuel, G., Introduction to stochastic calculus and to the resolution of PDEs using Monte Carlo simulations - Lectures notes of XV Spanish-French School on Numerical Simulation in Physics and Engineering (2012), https://cel.archives-ouvertes.fr/cel-00736268.

Jerzy, T., Object-Oriented Computer Simulation of Discrete-Event Systems, published by Kluwer Academic Publishers in (1999).

Stanley J. F., Partial Differential Equations for Scientists and Engineers, Dover publication, inc. New york (1982).

TÉCNICO LISBOA

References II

- http://mathfaculty.fullerton.edu/mathews/n2003/montecarlopimod.html
- Zwillinger, D., Handbook of Differential Equations, 3rd edition, Academic Press (1997).
- Talay, D., Monte Carlo Methods for PDE's. In Encyclopaedia of Mathematics, M. Hazewinkel (Ed.). Kluwer Academic Press (1997).
- Maire, S. and Talay, D., On a Monte Carlo method for neutron transport criticality computations, IMA Journal Numerical Analysis, 657-685 (2006).
- Vajargah, B. F. and Vajargah, K. F., Monte Carlo Method for Finding the Solution of Dirichlet Partial Differential Equations, Applied Mathematical Sciences, 453 -462 (2007).