Monte Carlo Methods for Numerically Solving Partial Differential Equations

Nick Handelman

5/01/2018

1 Introduction

Nicholas Metropolis, a scientist at Los Alamos National Laboratory in the 1940's who coined the term "Monte Carlo", was involved with the early development and application of Monte Carlo (MC) methods. In [2], he discusses MC's history and development and the people involved. The people involved included well known scientists such as John Von Neumann, Enrico Fermi, Edward Teller, Stanislaw Ulam and others.

He begins by discussing the creation of ENIAC, the first electronic computer, during World War 2. This is an important development because MC requires a large amount of computation to be effective, and such computational ability was not previously available. He also discusses the development, at Los Alamos, of thermonuclear weapons, research which he calls "the spark" that led to the development of MC. Conventional tools, including PDEs, for modeling the feasibility of a thermonuclear weapon required too much simplification, and the model did not give strong support for feasibility. However, Ulam suggested a statistical approach and Von Neumann submitted a proposal for that approach (to be called MC) to his superiors.

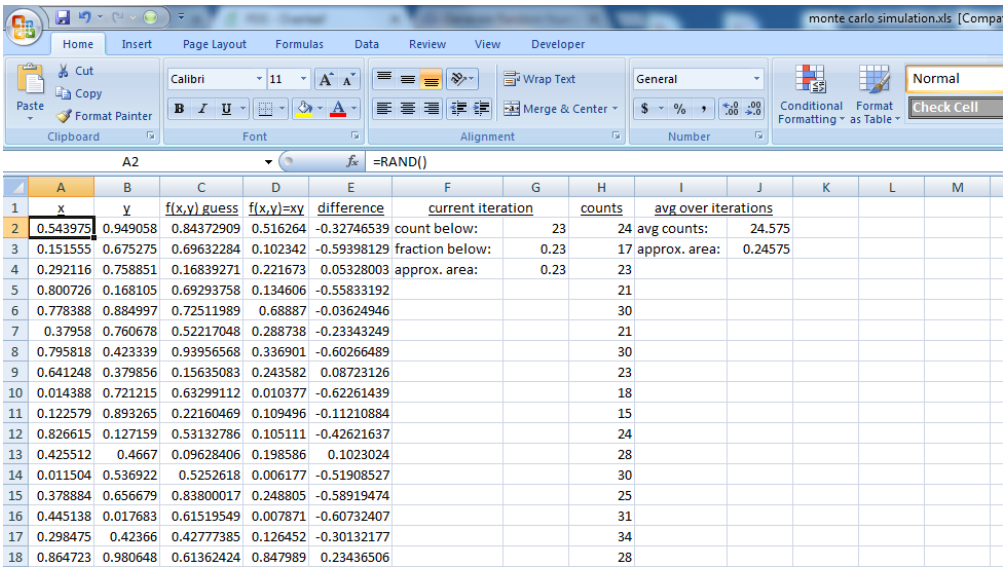
In 1947, Monte Carlo was first tested on ENIAC. 9 problems were considered, and the results of MC were compared to the results from other approaches and statistical analyses on the 9 problems. MC proved comparable to other methods, thus was developed a "new approach" to modeling (or so they thought). In fact, Fermi had independently developed MC but had never published it. Apparently, he had been using it for 15 years with only a mechanical calculator. In 1947, later in the year that Von Neumann had submitted his proposal, Fermi was visiting ENIAC and recognized the applicability of electronic computing to MC. He and King built the "Monte Carlo Trolley" (FERMIAC), an analog device implementation of a MC.

From there, others learned of MC and applied it to their work. They visited ENIAC and its successor MANIAC for the required computation power. Today, thanks to widespread computing power, MC is widely applied in many fields, ranging from climate change to finance.

2 Results - include at least two unique examples

2.1 Numerical Integration

Consider $f(x, y) = xy$ and note that $I = \int_0^1 \int_0^1 f(x, y)\, dxdy = \int_0^1 \int_0^1 xy\, dxdy = \frac{1}{4}$. So, when we run a Monte Carlo simulation to approximate I, we expect the approximation to be close to $\frac{1}{4}$. Keep in mind that the max value of $f(x, y)$ is 1 and the cube containing $f(x, y$ has volume 1.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	x	y	f(x,y) guess	f(x,y)=xy	difference	current iteration		counts	avg over iterations				
2	0.543975	0.949058	0.84372909	0.516264	-0.32746539	count below:	23	24	avg counts:	24.575			
3	0.151555	0.675275	0.69632284	0.102342	-0.59398129	fraction below:	0.23	17	approx. area:	0.24575			
4	0.292116	0.758851	0.16839271	0.221673	0.05328003	approx. area:	0.23	23					
5	0.800726	0.168105	0.69293758	0.134606	-0.55833192			21					
6	0.778388	0.884997	0.72511989	0.68887	-0.03624946			30					
7	0.37958	0.760678	0.52217048	0.288738	-0.23343249			21					
8	0.795818	0.423339	0.93956568	0.336901	-0.60266489			30					
9	0.641248	0.379856	0.15635083	0.243582	0.08723126			23					
10	0.014388	0.721215	0.63299112	0.010377	-0.62261439			18					
11	0.122579	0.893265	0.22160469	0.109496	-0.11210884			15					
12	0.826615	0.127159	0.53132786	0.105111	-0.42621637			24					
13	0.425512	0.4667	0.09628406	0.198586	0.1023024			28					
14	0.011504	0.536922	0.5252618	0.006177	-0.51908527			30					
15	0.378884	0.656679	0.83800017	0.248805	-0.58919474			25					
16	0.445138	0.017683	0.61519549	0.007871	-0.60732407			31					
17	0.298475	0.42366	0.42777385	0.126452	-0.30132177			34					
18	0.864723	0.980648	0.61362424	0.847989	0.23436506			28					

Figure 3.1

Refer to Figure 3.1. This is a portion of an excel sheet created to run a Monte Carlo simulation to approximate I (area). Columns A (x), B (y) and C (f(x,y) guess) each consist of 100 rows of randomly generated numbers between 0 and 1. It is clear why x and y are between 0 and 1 (the limits of integration). To keep the problem simple, I chose $f(x, y)$ such that its max value is 1. Column D (f(x,y)=xy) calculates $x * y$ and column E (difference) calculates $D - C$.
Each iteration generates new random numbers, and Column G (current iteration) determines the approximate area for that iteration. In column H (counts), I manually entered the results of 40 iterations. In column J (avg over iterations), I averaged the approximate areas to show that, although the individual iterations varied around 25, the average over all the iterations approached 25. In this example, counts is divided by 100 and multiplied by 1 (volume) to approximate the value of I.

2.2 Numerical PDE

Consider the following Dirichlet problem in a square:

$$\mu_{xx} + \mu_{yy} = 0 \text{ (PDE)}$$

$$\mu(x, 1) = sin(\pi x) \text{ and } \mu(x, 0) = \mu(0, y) = \mu(1, y) = 0 \text{ (Boundary Conditions)}$$

$$0 < x < 1 \text{ and } 0 < y < 1 \text{ (Domain)}$$

Let's consider the solution to the problem when $x = 0.5$ and $y = 0.5$.

2.2.1 Exact Solution

An exact solution to the problem can be found using example 1 in [4]:

$$\mu(x, y) = \sum_{n=0}^{\infty} c_n sin(\tfrac{n\pi x}{a}) sinh(\tfrac{n\pi y}{a}) \text{ with } c_n = \tfrac{2}{a sinh(\frac{n\pi b}{a})} \int_0^a f(x) sin(\tfrac{n\pi x}{a}) dx$$

a and b are the max x and y values, so let $a = 1$ and $b = 1$. $f(x)$ is the boundary value at $\mu(x, 1)$ so $f(x) = sin(\pi x)$. Substituting these values in and solving the integral gives:

$$c_n = \tfrac{2}{sinh(n\pi)} \int_0^1 sin(\pi x) sin(n\pi x) dx$$
$$\text{So, for } n = 1,\ c_1 = \tfrac{1}{sinh(\pi)} \text{ and for } n \neq 1,\ c_n = 0$$
$$\text{Thus, } \mu(x, y) = \tfrac{1}{sinh(\pi)} sin(\pi x) sinh(\pi y)$$
$$\text{The solution is } \mu(0.5, 0.5) = 0.199268407669193$$

2.2.2 Monte Carlo Approximate Solution

The approximation closely follows the explanations of the Tour du Wino game explained on page 347 of [1]. I changed the boundary condition $\mu(x,1) = 1$ to $\mu(x,1) = sin(\pi x)$ so I could calculate an exact solution in the previous section.
Refer to Appendix A for the Octave code I used to run the Monte Carlo simulation. I ran the simulation 10 times and the results are give in the table below. If you compare the approximate results to the exact result from the previous section, you will notice that the approximations are close.

simulation	1	2	3	4	5	6	7	8	9	10
approximation	0.20370	0.20624	0.18586	0.20512	0.19005	0.21909	0.21292	0.19912	0.19508	0.20897

The approximation is based on discretization of the domains of x and y. To better approximate the solution, increase the number of discrete parts of the domain. In the code, I used 9 parts. Increasing the value of variable *num_partitions* will improve the approximation. Notice that if you wish to approximate the solution when $x = 0.5$ and $y = 0.5$, the value of *num_partitions* must be odd and the values of variables r and c must be changed to the median partition. For example, using 15 partitions, $r = c = 8$.
In this example, the probability of moving to any of the neighboring points on the grid is equal. The calculation of where to move next is given by $unifrnd(0, 4)$, which is a uniform random number generator function. If the coefficients in the PDE aren't 1, other random number generators may have to be used, or specific equations may have to be programmed. Page 350 of [1] shows an example with a variable coefficient.

3 Conclusion

References

[1] Farlow, Stanley J. Partial Differential Equations for Scientists and Engineers. Dover, 1993.

[2] Metropolis, Nicholas. THE BEGINNING of the MONTE CARLO METHOD. Los Alamos Science, 1987, pp. 125130., library.lanl.gov/cgi-bin/getfile?00326866.pdf.

[3] Owen, A. (n.d.). Monte Carlo theory, methods and examples [In Progress]. Retrieved April 29, 2018, from http://statweb.stanford.edu/ owen/mc/

[4] Two-Dimensional Laplace and Poisson Equations. math.la.asu.edu/ kuiper/502files/Laplace.pdf.

Appendix A Numerical PDE Monte Carlo Octave Code

```
num_walks = 0;
num_partitions = 9;

results = zeros(num_partitions);

while(num_walks < 1000)
    r = 5;
    c = 5;

    while (true) %run until a boundary is hit
        rn=unifrnd(0,4);

        if(rn<=1) %move left
            c = c - 1;
        elseif(rn<=2) %move right
            c = c + 1;
        elseif(rn<=3) %move up
            r = r - 1;
        else %move down
            r = r + 1;
        endif

        %check if hit boundary
        if(c==1) %left boundary
            break;
        elseif (c==num_partitions) %right boundary
            break;
        elseif (r==1) %top boundary
            break;
        elseif (r==num_partitions) %bottom boundary
            break;
        endif
    endwhile

    results(r,c)++;
    num_walks = num_walks + 1;
endwhile

results = results(1,:)./num_walks;
rewards = [0; sin(pi*0.125); sin(pi*0.25); sin(pi*0.375); sin(pi*0.5); sin(pi*0.625); sin(
results*rewards
```