To: Dr. Palmtag

From: Grayson Gall

Subject: Monte Carlo method for calculating square of neutron displacement using n groups

Date: November 22, 2021

Introduction

During this memo I will be presenting an overview of the theory and application of the Monte Carlo method to calculate the square of the magnitude of the displacement vector for a neutron emitted from a point source. The neutron will be born in the fast group and then subsequently scattered through a given number of energy groups until it is absorbed. To complete this simulation for the multiple energy groups I created a general program that will accept input for any given number of energy groups. To improve the run times for my program I also leveraged the parallel computing toolbox in MATLAB.

Theory

The Monte Carlo method is a simulation that uses random sampling and large sample sizes to find the average value of a given quantity. The first thing we need to do when we formulate random sampling for the Monte Carlo method is to link pseudo-random numbers to the event we are sampling.

Probability of Events

We often refer to the probability of something happening and we model these probabilities with something called a, probability distribution function or PDF. Say for our context our PDF is a function of a distance, r. It is important to note that we cannot simply evaluate a PDF at any given point as the probability of any event occurring is 0. Rather we calculate the probability of that and any previous events happening. This is will be our Cumulative Probability Distribution function of CDF and is given by integrating over the PDF. Lets suppose for our context our PDF is a function of distance from a neutron source

$$PDF = p(r) \tag{1}$$

If we want to find the probability of a particle existing a distance r from our source we simply integrate from 0 to that distance.

$$CDF = P(r) = \int_0^r p(r)\, dr \tag{2}$$

Sampling

We are sampling the path that a particle takes after it is emitted from a point source and then scattered through along its way to be absorbed. We can perform this task by randomly sampling the displacement of the particle after it is born and after each subsequent scattering event. We can perform this sampling in spherical coordinates as it is a convenient way to sample displacement in any direction. This means we need to sample 2 quantities, the magnitude of the

displacement and the direction of the magnitude. To sample the magnitude we start with the probability distribution function given by equation 4.18 in the Monte Carlo handout

$$p(r) = \Sigma_t \exp(-\Sigma_t r) \tag{4.18}$$

$$\Sigma_t = \Sigma_{\text{total}} - \bar{\mu}\Sigma_s \tag{3}$$

We then find the CDF by integrating to an arbitrary distance, r.

$$P(r) = \int_0^r \Sigma_t \exp(-\Sigma_t r)\, dr = 1 - \exp(-\Sigma_t r) \tag{4.20}$$

Now to actually sample r we set our PDF equal to our random number, ξ and then solve for r

$$\hat{r} = -\frac{1}{\Sigma_t} \ln(1 - \xi) \tag{4.21}$$

The other quantities we need to sample are our two angles in spherical coordinates. We have an azimuthal angle, ϕ and a polar angle θ. We now make the assumption that we have an isotropically scattering system, we can have any combination of azimuthal and polar angle. This allows us to simply sample these angles using the formulas provided in section 4.2.1 of the Monte Carlo handout.

$$\bar{\mu} = 0 \tag{4}$$

$$\Sigma_t = \Sigma_{\text{total}} \tag{5}$$

$$\hat{\phi} = 2\pi\, \xi \tag{4.16}$$

$$\hat{\theta} = \pi\, \xi \tag{4.17}$$

We now use these three quantities to sample the displacement of our particle after it is emitted or and after a scattering event. We can then transform this spherical displacement vector to our particles current position by performing a transformation to Cartesian coordinates and adding it to the particles current position.

Random Numbers

Another issue we need to address is the matter of our pseudo-random numbers. The way our random numbers are generated can play a big impact on the accuracy our simulation. If we do not have uniformly distributed random numbers our results will have a bias to whatever numbers have a peak in the random number PDF. I wrote my program in MATLAB and several options for random number generators exist. I used the default random number generator in MATLAB which uses the Mersenne twister algorithm. This algorithm has an approximate period of $2^{19937} - 1$, meaning you should have to generate approximately $2^{19937} - 1$ numbers before getting the same number twice.

What is an Event

Throughout the earlier sections I have been referring the particle under going some type of event. An event refers to what happens to a particle after it had reached a certain position. After a particle is born it will travel some distance before interacting with the material around it. When the particle does interact with the substance through which it is traveling that is called an event. There are two main types of events when the particle interacts with its surroundings; an Absorption event or a Scattering event. An absorption event is pretty self-explanatory the particle is absorbed by the medium that it has interacted with. After this occurs we consider the current position of the particle to be its final position. The other option is a scattering event. This means that the particle "bounces" off of the medium it is interacting with and keeps traveling until it is absorbed. When we model a particle with multiple energy groups there can be several types of scattering between the energy groups. For example a particle in energy group 1 can scatter to any of the energy groups below it and this is called down-scattering. A particle in an energy group can also scatter and stay in the same energy group, this is called self-scattering. Finally, a particle can scatter from a lower energy group to a higher energy group. For example, a particle in energy group 4 could scatter to energy group 3 and this is called up-scattering. However, this type of scattering is relatively unlikely and is often times neglected.

Determining the Type of Event

Now that we have discussed what an event is we need to use CDF and sample with random numbers. So how do we create this CDF? The answer lies with our cross section data. The first step is to find the PDF for a given set of cross-sections. To do this we simply divide each cross section by the total cross section for that group

$$p_i = \frac{\Sigma_i}{\sum_i \Sigma_i} \tag{6}$$

Here i denotes the cross section for reaction i in the data set

We use this to find the piece-wise continuous PDF for our data set. Now we can build the CDF and this will be what we use to determine the type of Event that occurs. Our CDF for any given event is given by the sum of the PDF's of all previous events

$$P_n = \sum_{1}^{n} p_i \tag{7}$$

Here n denotes nth reaction, for which we are calculating the CDF and i denotes the ith reaction

The way we actually determine which event occurs is of course with our random numbers. We compare our random number to the CDF's of each event and we determine the type of event based on the range in which it falls. Say we have an event i and a second event j. If the value of our random number is greater than the CDF for i and less than the CDF for event j, event j will occur. If our random number is less than the CDF of event i then event i will occur. For a more concrete examle consider the following. We have three events each with the CDF's given by

$$P_1 = 0.3 \tag{8}$$

$$P_2 = 0.7 \tag{9}$$

$$P_3 = 1.0 \tag{10}$$

If we have a random number, 0.1 then event 1 will happen.
If we have a random number, 0.5 then event 2 will happen.
If we have a random number, 0.8 then event 3 will happen.

One Group Theory

For the case of one energy group our it is actually possible to find the analytic solution to find the average square of the distance traveled by multiplying our PDF, given in section 4.1 of the Monte Carlo handout, by r^2 and then integrating from 0 to infinity

$$p(r) = \frac{r}{L^2} \exp\left(-\frac{r}{L}\right) \tag{4.5}$$

$$\langle r^2 \rangle = \int_0^\infty r^2 p(r)\, dr = 6L^2 \tag{4.10}$$

We can use this analytic solution and compare it to our generated solution to see if our program gives the correct answer for the simple case of one energy group.

Error Analysis

We now have all the tools to create a simulation for the square of the displacement for our particles and that is great but it is not very useful unless we have a way to quantify the possible error in our solution. When we have the analytic solution this is pretty easy we can simply find the difference between our solution and the actual solution.

$$\text{Error} = |\text{actual} - \text{prediction}| \tag{11}$$

However, we do not have an analytic solution for the square of the distance for more than 1 energy group. So we need a way to quantify error using only our predicted values. To do this we start with the sample variance of our solution

$$s^2 = \frac{N}{N-1}\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2 - \langle x \rangle^2\right) \tag{4.33}$$

Now we can find the standard deviation of our sample set by taking the square root of the variance

$$s = \sqrt{s^2} \tag{4.31}$$

Finally, we find a prediction for our error by dividing by the square root of N

$$R \approx \frac{s}{\sqrt{N}} \tag{4.32}$$

N = Number of particles

Results

The first part of this project was to match the analytic solution for one energy group. To do this I hard-coded a program that ran the calculation for a variable number of particles and compared the calculated values of $\langle r^2 \rangle$ and the estimated error to analytic solution and the actual error. The simulation was ran for 6 different number of particles, 1e2, 1e3, ... 1e7.

$$\bar{\mu} = 0 \tag{12}$$

$$\Sigma_a = 0.006 \text{ cm}^{-1} \tag{13}$$

$$\Sigma_s = 0.050 \text{ cm}^{-1} \tag{14}$$

$$\langle r^2 \rangle = 5952.4 \text{ [cm}^2\text{]} \tag{15}$$

Analytic Solution

Run Time [s]	N	MC $\langle r^2 \rangle$ [cm^2]	s [cm^2]	R [cm^2]	Actual Error [cm^2]	Actual in Range?
0.053650	1e2	4270.0	6509.1	650.91	1682.4	no
0.045657	1e3	5318.5	8953.0	238.12	633.85	no
0.063843	1e4	6053.6	10226	102.26	101.17	yes
0.093347	1e5	5958.5	9836.0	31.104	6.1476	yes
0.633919	1e6	5961.3	9944.2	9.9442	8.9200	yes
5.870700	1e7	5954.2	9921.5	3.1375	1.7699	yes

Table 1: Hard Coded 1 Group

The next thing I did was run the one energy group calculation with my general n energy group code to make sure that the two gave the same result

Run Time [s]	N	MC $\langle r^2 \rangle$ [cm^2]	s [cm^2]	R [cm^2]
0.040154	1e2	5813.8	9382.3	938.23
0.061330	1e3	5783.0	9507.0	300.64
0.093417	1e4	5856.5	9360.3	93.603
0.196331	1e5	5983.7	9839.0	31.114
0.878507	1e6	5948.4	9886.3	9.8863
7.317530	1e7	5947.0	9888.4	3.127

Table 2: General Solution 1 Group

After Confirming my results I moved onto multi-group problems

4-group Absorption Cross Section (cm^{-1})

Group	Absorption
1	4.3708E-04
2	5.7416E-03
3	1.5010E-02
4	3.7309E-02

4-group Macroscopic Scattering Matrix (cm^{-1})

	To Group 1	To Group 2	To Group 3	To Group 4
From Group 1	1.82647E-01	8.77074E-02	3.71651E-05	7.00404E-07
From Group 2	0	4.19655E-01	1.63779E-01	4.15248E-03
From Group 3	0	1.46487E-05	6.34985E-01	4.39694E-01
From Group 4	0	0	1.32440E-01	2.48066E+00

Figure 1: 4 Group Cross Section Data

Run Time [s]	N	MC $\langle r^2 \rangle$ [cm^2]	s [cm^2]	R [cm^2]
0.038482	1e2	134.18	163.51	16.351
0.043499	1e3	126.25	149.55	4.7291
0.102598	1e4	136.65	188.92	1.8892
0.609245	1e5	135.56	184.53	0.58353
5.783653	1e6	135.82	186.37	0.18637
62.04317	1e7	135.58	186.17	0.058872

Table 3: 4 Groups

7-group Absorption Cross Section (cm^{-1})

Group	Absorption
1	6.0105E-04
2	1.5793E-05
3	3.3716E-04
4	1.9406E-03
5	5.7416E-03
6	1.5001E-02
7	3.7239E-02

7-group Macroscopic Scattering Matrix (cm^{-1})

	To Group 1	To Group 2	To Group 3	To Group 4	To Group 5	To Group 6	To Group 7
From Group 1	4.44777E-02	1.13400E-01	7.23470E-04	3.74990E-06	5.31840E-08	0	0
From Group 2	0	2.82334E-01	1.29940E-01	6.23400E-04	4.80020E-05	7.44860E-06	1.04550E-06
From Group 3	0	0	3.45256E-01	2.24570E-01	1.69990E-02	2.64430E-03	5.03440E-04
From Group 4	0	0	0	9.10284E-02	4.15510E-01	6.37320E-02	1.21390E-02
From Group 5	0	0	0	7.14370E-05	1.39138E-01	5.11820E-01	6.12290E-02
From Group 6	0	0	0	0	2.21570E-03	6.99913E-01	5.37320E-01
From Group 7	0	0	0	0	0	1.32440E-01	2.48070E+00

Figure 2: 4 Group Cross Section Data

Run Time [s]	N	MC $\langle r^2 \rangle$ [cm^2]	s [cm^2]	R [cm^2]
0.040154	1e2	233.43	365.53	36.553
0.069961	1e3	194.34	246.89	7.8074
0.130769	1e4	202.39	276.75	2.7675
0.745308	1e5	198.12	278.02	0.87918
7.058546	1e6	198.63	279.08	0.27908
72.04141	1e7	198.71	278.67	0.088123

Table 4: 7 Groups

Summary

The Monte Carlo technique is a great tool in the world of simulation. This random sampling technique allows us to solve complex problems for which an analytic solution may be too difficult or even impossible to find. However, this technique is not without its own unique challenges. There can be considerable difficulty in creating PDFs and CDFs as well as with the computational resources it can take to run a program like this when a program is not properly optimized. Working with some of my peers who did not use a parallel loop for their particles run times could be on the order of 10-15 minutes for the 1e7 particle case. While this is not a ridiculous run time it does tell us that finding the minimum number of samples to take for a large project may be important. Overall this project was a lot of fun and I feel like I have a new tool in my computational tool belt both for personal and professional projects.

Future Work

One big thing that could be improved in my code is the issue of the number of simulations. The approach of doing one large Monte Carlo simulation does give good results but one way to optimize for speed could be to run more smaller simulations. This could be an interesting thing to compare to the large simulation as I would assume that the error would be within a similar range but time could be significantly decreased. Additionally, if I need to run a simulation like this with a much larger sample size it may be worth the effort to move everything to a faster language such as C++ or other variations of C. With some considerable difficulty it would also be possible to put this simulation directly into assembly but I think the returns from speed would not be worth the pain of coding in assembly.