NE155 Project Interim Report

Milos Atz

April 15, 2016

1 Introduction

Monte Carlo (MC) methods offer an alternative to deterministic methods of solving numerical problems. Rather than directly solving discretized equations, MC utilizes repeated random sampling to obtain statistically likely solutions. Given enough repetitions, the MC result will converge to the true value. MC methods can be applied to any problem that has a probabilistic interpretation and are most are most useful when that problem is impossible or very slow to solve using other methods. Nuclear computing has a long history of utilizing MC to solve numerical problems. MC and digital computing were originally developed together to perform calculations pertaining to thermonuclear weapons; the grandfather of the modern digital computer, ENIAC, was programmed to carry out those types of calculations [1] [2]. Today, MC methods and codes (such as MCNP) are still widely used due to their ability to solve complicated phase-space problems without discretization or averaging of material properties [3]

This project aims to design and implement an MC method to solve for the neutron flux in a 1D, multiregion space. Constant energy and material properties are assumed. The code will use a collision tally to resolve the flux shape. In addition, the relative error will be determined. The effects of various variance reduction methods will be explored. This interim report represents a first draft of the final report and includes sections highlighting the mathematics behind MC, the algorithm implemented to solve the problem, and plans for completion as well as outstanding questions.

2 Mathematics

Monte Carlo simulations rely on the probabilistic interpretation of physical phenomena. Values for certain properties and outcomes of events can be obtained based on probability distributions. Every variable has a probability density function and a cumulative distribution function. The probability density function (PDF) describes the probability $\in [0, 1]$ that some value of the random variable will be achieved. For a continuous random variable, the PDF, $p(x)$, has the following characteristic [4] [5]:

$$p(a \leq x \leq b) = \int_a^b p(x)dx \tag{1}$$

The probability of a variable is always greater than 0. Integrating the PDF from $-\infty$ to ∞ accounts for all possible values and thus should return a value of 1. The integral of the PDF is called the cumulative distribution function (CDF). The CDF, $F(x) = P(X \leq x)$, describes the probability that a continuous random variable X will be less than or equal to the value x.

$$P(X \leq x) = F(x) = \int_{-\infty}^{x} p(x)dx \tag{2}$$

The above equations are for continuous random variables; for discrete random variables, the integrals can be replaced with summations. Individual values for physical phenomena can be resolved by repeated random sampling according to the CDF. Based on a uniformly distributed random variable $\xi \in [0, 1]$, values for physical phenomena can be found using the associated CDF: $F(x) = \xi$. The challenge involves identifying the PDF/CDF for a given physical phenomena and finding F^{-1} such that $x = F^{-1}(x)$. This is called inverting the CDF [6].

There are many ways to sample random variables. Some simple methods include direct discrete sampling (i.e. to determine which reaction will take place), direct continuous sampling (determine the number of mean free paths), and rejection sampling (for non-invertible CDF). In this MC problem, as with all MC transport problems, each particle is followed from it's birth to its death. Through its life, interactions and outcomes are randomly sampled from probability distributions using transport data. In this project, sampling is used to determine:

1. The direction of a particle when it is born or scatters.
2. The type of collision that might occur, either absorption or scattering.
3. The path length from the current particle location to it's next event.

The sampling of these values is described in more detail in the next section. As more and more particles are followed, the distributions of particles become more resolved and the quantities requested by the user, called tallies, converge to a statistically expected value. The goal is to accumulate a PDF for the quantities of interest that is the same as that of the real thing. The expected value, or average, of the accumulated PDF should be the same as that of the real value.

$$E(x) = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3}$$

Assuming the simulation has been designed correctly and N is sufficiently large, the expected value should be the solution to the problem.

3 Algorithms

The algorithm for this code can be described by the following steps.

1. Create a neutron at $x = -1$. Assume the interface between the two regions is at $x = 0$. Distance units are all in cm. Vacuum boundary conditions are considered at $\pm a$.

2. Sample direction: generate a random number ξ on the uniform distribution [0,1) The direction is positive (right) if $\xi \geq 0.5$. The direction is negative (left) if $\xi < 0.5$.

3. Sample the path length (s) for the region that the neutron is in. If the neutron is on the boundary, sample the path length for the region that the neutron is entering.

$$F_s(s) = \Sigma_t exp(-\Sigma_t s) \tag{4}$$

$$s = -ln(\xi)/\Sigma_t \tag{5}$$

4. If the particle is moving away from the interface between the two regions, check whether the particle is going to collide in the region or leak by comparing the sampled path length with the distance between the particle location and the boundary. If the sampled path length is greater than the distance to the boundary, the particle leaks and is terminated. If not, the particle will collide within the region. Move the particle to the new location and sample what type of collision will occur - scattering or absorption. If the particle is absorbed, it is terminated. If the particle scatters, return to step 2 and resample direction and path length.

5. If the particle is moving toward the interface, measure the distance between the particle and the interface to determine whether the particle changes region. If the distance between the particle and the interface is less than the sampled path length, the particle will enter the other region. Move the neutron to the boundary and return to step 3 to resample the path length for the region the particle is entering. If the distance between the particle and the interface is greater than the sampled path

length, the particle will collide before it reaches the interface. Move the particle to the new location and sample what type of collision will occur - scattering or absorption. If the particle is absorbed, it is terminated. If the particle scatters, return to step 2 and resample direction and path length.

Every time a collision occurs, a score is added to the tally. Each score has the same weight. The collision tally is related to the flux by the following equation:

$$\overline{\phi_V} = \frac{1}{V}\frac{1}{N}\sum_i x_i \tag{6}$$

If the algorithm is considering weighted particles as a part of a variance reduction method, x_i can be replaced with the particle weight w_i. In order to resolve the flux over the entire space, the collision tallies are binned according to the location at which the collision occurs. The space between $\pm a$ is discretized into $2a$ bins. This means that each bin has a width of 1 and $\frac{1}{V}$ reduces to 1.

4 Code Use

In it's current form, the code is easy to use. All that is required is that the user run the Python script from the Terminal. The user is prompted for the number of particles that they would like to simulate (I recommend at least 10,000, but 100,000 is better). After that, the user is prompted for the width of the bin size used to discretize the tally to obtain spatial resolution of the flux. Based on that information, the code is run and two plots are generated. The first is the total collisions discretized in the space (at this point, this is done only using a step size of 1.0). Once that plot is closed, a second is generated, plotting the flux in each bin according to Equation 6 based on the number of particles simulated and the bin size input by the user.

5 Plans for Completion and Questions

Moving forward, there are many things I'd like to implement. I would like the code to prompt the user for more information, including the source point, the interface boundary between the two regions, and the location of the vacuum boundaries. I'd like to investigate whether it's feasible to tally the number of particles moving between the regions (current). In addition, I still need to implement the considerations of relative error and test the effects of various variance reduction techniques (implicit capture, splitting, rouletting).

To test the code, I plan to analytically solve the diffusion equation for a 1D planar source with vacuum boundary conditions and implement that problem in code. I can adjust the problem I've aimed to solve here to be similar by eliminating the second region and equating all boundaries and material properties. I will then attempt to compare the solutions.

In addition, to test the Python code itself and it's behavior, I plan to implement a series of tests and aggregate them using nosetest (or similar) [7]. These tests are currently TBD but I have some minor experience with testing code, having written the tests for an R package for adaptive rejection sampling developed for a statistical computing class last semester.

I have some general questions at this stage. I am hoping to receive some feedback on my methodology to discretize the collision tally to resolve the spatial behavior of the flux. Is this sound? In addition, is the scope of this (proposed) work sufficient? If not, how should I plan to augment it?

References

[1] R. Eckhardt. *Stan Ulam, John Von Neumann, and the Monte Carlo Method* Los Alamos Science Special Issue 1987.
`http://library.lanl.gov/cgi-bin/getfile?00326867.pdf`

[2] R. Slaybaugh *UCBNE155 Class Notes Spring 2016*. April 6, 2016

[3] *MCNP5 - A General Monte Carlo N-Particle Transport Code, Version 5*. Volume 1: Overview and Theory. X-5 Monte Carlo Team, Los Alamos National Laboratory (rev. 2008).

[4] R. Slaybaugh *UCBNE155 Class Notes Spring 2016*. April 8, 2016

[5] R. Slaybaugh. *UCBNE250 Class Notes Fall 2015*. October 21, 2015.

[6] R. Slaybaugh. *UCBNE250 Class Notes Fall 2015*. October 26, 2015.

[7] *nose is nicer testing for python* Usage documentation v.1.3.7.