PH3010 MSci Skills Project
X-ray simulation with the Monte Carlo method

1 Introduction

In this project you will investigate the production of x-rays and their interactions with matter using the Monte Carlo method. This is a technique for simulating processes using sequences of random numbers. The first part of the problem is to simulate the production of individual photons, each having a different energy. The energies are chosen 'at random' in such a way that the values follow a given distribution.

We then allow each photon to hit a layer of matter (in the example below, 1 mm beryllium). There is a certain probability for the photon to be absorbed, and this probability depends on the photon's energy. So for each photon we 'toss a coin' (in the computer, of course) to decide whether the photon makes it through. We can then make a histogram of the energies of the photons which penetrate the layer.

2 The assignment

Your assignment is to simulate the production of x-ray photons from a synchrotron radiation source and to simulate their passage through a layer of matter. The tasks include the following:

- simulate random numbers uniformly distributed between 0 and 1;
- use the uniformly distributed numbers to simulate photon energies using the acceptance–rejection technique;
- simulate the absorption of photons in matter;
- make a histogram of the energies of the photons which are not absorbed.

3 Generating random numbers

Simulation of random processes by the Monte Carlo method can be broken down into two basic steps. First, random numbers are generated which follow a uniform distribution between 0 and 1. Next, these are used to generate random numbers which correspond to physical quantities (such as photon energies), which in general have a different frequency distribution.

There are many computer algorithms available for producing uniformly distributed random numbers. They produce a sequence of numbers according to a certain rule, and would produce the same sequence if you repeat the procedure. In this sense they are not truly random, and therefore are often called pseudorandom. For our purposes this is just as good.

A simple but effective algorithm is the multiplicative linear congruential generator (MLCG). This generates a sequence of integer values $n_1, n_2, \ldots$ according to the rule

$$n_{i+1} = (an_i) \bmod m. \tag{1}$$

Here the *multiplier* a and *modulus* m are integer constants and the mod (modulo) operator means that you take the remainder of an_i divided by m. The values n_i follow a periodic sequence in the range $[1, m-1]$. The initial value n_0 is called the *seed*. The transformation

$$r_i = n_i/m \tag{2}$$

then gives numbers in the interval $(0, 1)$.

The art of random number generation is to choose a and m so that the resulting values r perform well in various tests of randomness. For a 32-bit integer representation, for example, $m = 2147483399$ and $a = 40692$ have been shown to give good results.[1] You may wish to try implementing your own random number generator. If time constraints prevent this you can use the `Math.random` method from directly from java, e.g.,

```
double x = Math.random();      // assigns random value to x
```

4 X-ray photons from synchrotron radiation

When a charged particle such as an electron traverses a magnetic field, it emits electromagnetic radiation (photons) tangential to its direction of motion. The energy spectrum of these photons has a certain derivable form which depends on the electron's energy and the strength of the magnetic field. This is difficult to calculate, but fortunately we don't have to; spectra can be obtained from the Center for X-ray Optics at the Lawrence Berkeley National Laboratory at `http://www-cxro.lbl.gov/`. Follow the links to 'x-ray interactions with matter', and then 'synchrotron bend magnet radiation'. You can choose various parameters for the electron beam to produce energy spectra such as the one shown in Fig. 1. The data can be downloaded into a file for use in your project.

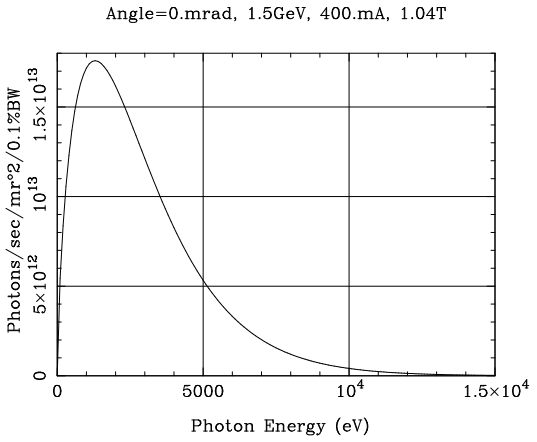

Figure 1: A photon energy spectrum from a synchrotron radiation source.

Once you have obtained a spectrum like Fig. 1, the next step is to generate photon energies such that the probability to obtain a given energy is proportional to the height of the curve. The easiest way to do this is with the *acceptance–rejection* technique.

[1] P.L. L'Ecuyer, Efficient and portable combined random number generators, *Commun. ACM* **31** (1988) 742.

- First generate two independent random numbers, r_1 and r_2, both uniformly distributed between 0 and 1.
- Use r_1 and r_2 to produce two more numbers: $E = r_1 E_{\max}$ and $f = r_2 f_{\max}$, where $E_{\max}$ is the maximum photon energy and $f_{\max}$ is at least as high as the maximum of the energy spectrum. For Fig. 1, for example, use $E_{\max} = 1.5 \times 10^4$ and $f_{\max} = 1.8 \times 10^{13}$. The two numbers E and f will be represented by a point somewhere on the energy spectrum plot.
- If the point (E, f) is below the curve, accept E as the generated photon energy, otherwise, reject the value and repeat the procedure until a value is accepted.

The probability to accept the point is proportional to the height of the curve, so the accepted energies will have the desired distribution.

5 Interaction of x-rays with matter

We will imagine that the x-rays hit a layer of material of a given thickness x. We would like to know how many of the photons make it out the other side and what their energies are. The probability for a photon to be absorbed by the photoelectric effect before penetrating a distance x can be expressed as

$$P(\gamma \text{ of energy } E \text{ absorbed before } x) = 1 - e^{-x/\lambda(E)} \tag{3}$$

where $\lambda(E)$ is the attenuation length, which is in general a complicated function of the photon's energy. Attenuation lengths measured as a function of energy can be downloaded from the Center for X-ray Optics for a variety of materials; an example for beryllium is shown in Fig. 2. (Follow the links 'x-ray interactions with matter', 'attenuation length'.)

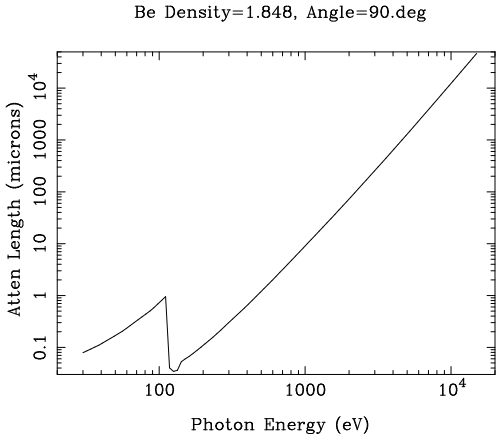

Figure 2: Attenuation length for x-rays in beryllium as a function of energy.

The strategy is thus to start by generating a photon with a certain energy. For this energy, determine the attenuation length $\lambda(E)$ and from (3) the probability P for the photon to be absorbed before making it through a given thickness of material, e.g. $x = 1$ mm beryllium. Then generate another random number r between 0 and 1. If $r < P$, we say that the photon

is absorbed, if not, it makes it through. (In fact, a real photon may make it through but its energy can be degraded by Compton scattering. For now we will ignore this effect.)

You will then have a subset of the photons which make it through the layer of material, and you should make a histogram of their energies. Generate enough events so that the energy spectrum of the photons getting through is reasonably well determined.

Should time permit, try varying the parameters of the problem such as the thickness of the material layer, its composition, the nature of the photon source, etc. You could, for example, determine the mean photon energy as a function of the thickness of a given absorber.

6 Software implementation

This section gives some advice on how to implement the calculation described above in a java program. At different stages it is necessary to read in numbers columns of numbers from a file that you get from the website `www-cxro.lbl.gov`. The numbers can be thought of as x and y values, i.e., you read in pairs of numbers (x_i, y_i) for $i = 1, \ldots, n$. Using these one needs an approximation for the function $y(x)$ for arbitrary x, that is, even at values of x that do not coincide with any of the x_i that were initially read in.

This task arises, for example, when obtaining the spectrum $f(E)$ of the photon energies by using the numbers (f_i, E_i) read in from a file. And one has essentially the same problem when obtaining the attenuation length $\lambda(E)$. So to carry out this task, it is useful to create a java class that can read in a file containing two columns of numbers, say, x and y, and the class should contain a method that returns the function $y(x)$, which can be based on an interpolation between the (x_i, y_i) points from the file. The basic idea is illustrated in Fig. 3.

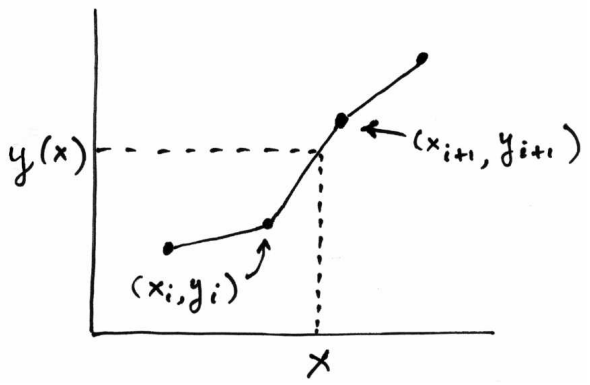

Figure 3: Illustration of the interpolation between points to approximate the function $y(x)$.

The suggested method for dealing with this problem is to create a class called, say, `FunctionFromTable`, since its purpose is to take as input a table of numbers, i.e., the points (x_i, y_i), $i = 1, \ldots, n$ and to provide a function $y(x)$, that could be evaluated with any value of x. For example, one could define the class `FunctionFromTable`, starting in the following way:

```
public class FunctionFromTable {

// data members

  int numPoints;
  double[] x;
  double[] y;
  double yScale;

//  constructor

  public FunctionFromTable(String file) {
    final int MAXPOINTS = 1000;
    x = new double[MAXPOINTS];
    y = new double[MAXPOINTS];
    this.readData(file);              // sets x[], y[], numPoints
    yScale = 1;
  }
```

The data members are an integer valued variable `numPoints` that will store the number of (x, y) points that are read in, and the values themselves are stored in the arrays `x` and `y`. In addition we have defined a scale factor `yScale` which will allow us to scale the function by a constant; for now ignore this.

The class's constructor, which in java always has the same name as the class itself, takes in a single argument of type `String`, which is the name of the input file. In the program that uses this class, we create an object of type `FunctionFromTable` with, e.g.,

```
  FunctionFromTable f = new FunctionFromTable("myFile.txt");
```

The code for the constructor for this function is a somewhat complicated, and involves using java's `BufferedReader` class to parse the numbers that it reads in from the file. This code will be provided to you.

The class must also define a function that can return our approximation to $y(x)$, e.g.,

```
  public double val(double xVal){

// Returns the function value.  Outside allowed range of x set function
// to zero, otherwise interpolate between points.

    double fVal = -9999.;

// YOUR CODE TO CALCULATE FUNCTION VALUE GOES HERE:


    return fVal;

  }
```

Part of your task is to implement the interpolation illustrated in Fig. 3 in the code above. Partial code for the `FunctionFromTable` class can be found on the project web page listed below.

To test the `FunctionFromTable` class you should write a short test program that, say, creates a `FunctionFromTable` object using one of the data files from `www-cxro.lbl.gov`, and tests that the values returned for selected input values of the argument agree with what you expect.

Once this class has been completed, it can be used twice in the main program that you will write to simulate x-ray energies: first, to obtain the energy spectrum $f(E)$, and second to obtain the attenuation length versus energy $\lambda(E)$. That is, your program will contain lines of the form

```
FunctionFromTable eSpec = new FunctionFromTable("energySpectrum.txt");
FunctionFromTable lambda = new FunctionFromTable("attenLength.txt");
```

and then to use these functions for a given value of the argument you will write, say,

```
double E = ...               // assign value of E
double f = eSpec.val(E);     // returns energy spectrum value f(E)
```

In fact, it is convenient to scale the energy spectrum numbers by a constant so that their values are close to unity. There is an additional constructor supplied in the file mentioned above that allows for this.

To make histograms of the energies that you generate with the Monte Carlo method, you can use the `Histogram` class that you used in the PH2150 course last year. There is an extended version of this class on the project web page listed below that provides methods to obtain the contents of a histogram along with the bin boundaries. These can be written to a to a text file and then plotted using, e.g., Excel.

7 References

This script and links to other resources can be found on the project web page: `www.hep.ph.rhul.ac.uk/~cowan/msci_skills.html`

X-ray data can be obtained from the web site of the Center for X-ray Optics at the Lawrence Berkeley National Laboratory: `www-cxro.lbl.gov/`.

The Monte Carlo method is described in:

S. Brandt, *Statistical and Computational Methods in Data Analysis*, Springer, New York (1997);

L. Lyons, *Statistics for Nuclear and Particle Physicists*, Cambridge University Press, Cambridge (1986);

G. Cowan, *Statistical Data Analysis*, Clarendon Press, Oxford (1998).

G. Cowan, September 2010