PHY407 Computational Lab 10
Random Processes and Monte Carlo Integration

Computational Background

Monte Carlo Integration

- Monte Carlo methods refer to methods that make use of random numbers in evaluating something. In this lab you will be using three Monte Carlo integration techniques:
  - "hit or miss" Monte Carlo integration is described in section 10.2
  - "mean value" Monte Carlo integration is described in section 10.2.1
  - "importance sampling" Monte Carlo integration is described in section 10.2.3
- All three methods have associated expressions for estimating errors.

Physics Background

Diffusion-Limited Aggregation

- Most of the following info is from wikipedia.
- Diffusion-limited aggregation (DLA) is the process whereby particles undergoing a random walk due to Brownian motion cluster together to form aggregates of such particles. This theory is applicable to aggregation in any system where diffusion is the primary means of transport in the system.
- DLA is important in processes such as crystal growth (e.g. snowflakes), dielectric breakdown (e.g. lightning), electrodeposition (e.g. coating 1 metal with another by donating electrons to ions in a solution) and bacterial colony growth.
- The clusters formed in DLA processes are called Brownian trees. Here are cool pics:

Left: copper aggregate formed from a copper sulfate solution in an electrode position cell.
Right: bacterial colony grown under starvation conditions.

Hypersphere

- A hypersphere is the equivalent of a sphere in higher dimensions. In 'n' dimensions, a hypersphere of radius R with coordinate directions $x_i, \quad i = 1, ...n$, is given by the points x_i such that:

$$\sqrt{\left(\sum_{i=1}^{n} x_i^2\right)} = R \tag{1}$$

  Convince yourself that this makes sense for a 3D sphere (the one you are used to) as well as a 2D sphere (i.e. a circle).

Lab Instructions

Recommended reading: Sections 10.1-10.2 of Newman

For grading purposes, you only need to hand in solutions to the following parts:

- Q1: Hand in your code and a figure showing the full trajectory of your random walk after 1000 steps.
- Q2: Hand in your code.
- Q3: Hand in your code and a snapshot of your final DLA distribution. The easiest way to get a snapshot is to do a screenshot of your display window (rather than trying to save the figure in the python program. vpython is weird this way).
- Q4: Hand in your code, the value for your integral and the value for your error (along with how you calculated it).
- Q5: Hand in your code.
- Q6: Hand in a plot of your histogram.
- Q7: Hand in your code, a plot of your histogram and the requested comment.

Diffusion Limited Aggregation

1. Do exercise 10.3 on page 457. Notes: I started the code for this during the lecture. Also, don't try and run it for 1 million steps, that will take forever. Try 5000 steps and if your plot at the end demonstrates particles sticking to the boundaries and to each other working properly, then you are all set. If not, try running a little longer. Finally, there is information on how to do the animation in either pylab or vpython in the appendix.

2. Do exercise 10.13 part (a) on pages 499-500 of the Newman text. This exercise builds upon exercise 10.3 on Brownian motion which I demonstrated in the lecture. Use 100 particles. (You can remove the arrows representing the boundaries so you can see the points stuck to the boundaries).

3. Do exercise 10.13 part (b) on pages 500-501.You don't need to modify your lattice into the 201x201 grid or make the anchored particles shown in different shades. You don't need to do part (c).

Volume of a 10-dimensional Hypersphere

4. Do exercise 10.7 on pages 471-472 of the Newman text. Hint: the volume of an 'n'-dimensional hypercube of length L is L^n. You will want to use 1 million points to get a good estimate. Also calculate your error estimate.

Importance Sampling

5. Equation 10.34 on page 472 involves an integral whose integrand has a divergence. Section 10.2.3 discusses using importance sampling to evaluate this integral. Before that, use the regular mean value method to evaluate the integral in this question. Check the lecture notes where we did a similar example (exercise 10.5), you should only have to do a little modification. Use 10,000 points for the integral.

6. Now alter your code to calculate the same integral 100 times. Make a histogram of the resulting integral values using 10 bins from 0.80 to 0.88. For an array of values (say its called I), you do this with the command:

```
hist(I,10,range=[0.8, 0.88])
show()
```

   You will also have to import hist from pylab.

7. Now implement importance sampling to evaluate the integral. Use the weight function $w(x) = x^{-1/2}$. You will need to determine the transformation needed to non-uniformly sample your region. The probability distribution you will be drawing from is

$$p(x) = \frac{1}{2\sqrt{x}} \tag{2}$$

   Pages 458-459 of the text explain how to find the transformation for a given probability distribution. Calculate the same integral 100 times and plot the histogram with the same number of bins and limits as done in question 6. Comment on what your histograms from the mean value method and the importance sampling method tell you about the methods.

Appendix

You can choose to make your animation using either vpython or pylab (with a similar structure for what we did with the tsunami in Lab 08. See instructions for vpython below:

Vpython animation

- You will need the following commands to get an animation of your particle working in vpython:

  ```
  #call the visual module
  from visual import *

  #define the initial position of the particle to be at xp,yp here:
  ...

  #set up a graph display window
  graph=display(background=color.white, title='Brownian Motion',center=(xp,yp,0))
  #plot the particle as a sphere
  particle=sphere(color=color.black,pos=(xp,yp,0),radius=1.0)
  #for location, plot the x and y axes
  xaxis=arrow(pos=(0,0,0), axis=(Lp,0,0),color=color.blue,shaftwidth=0.0001)
  yaxis=arrow(pos=(0,0,0), axis=(0,Lp,0),color=color.blue,shaftwidth=0.0001)

  #write your loop to update the position of the particle
  for t in range(10000):
        rate(1000) #this controls the speed of the animation, larger=faster
        xp,yp=nextmove(xp,yp) #nextmove is a function to update the position of the p
        particle.pos=(xp,yp,0)
  ```

  That should produce the animation.

- For the graph of the trajectory, store the xp,yp each time through the loop in arrays xarray and yarray then after the loop has finished, plot the xarray vs yarray. You may not get the show() to work, but you can just save the figure and see it there.