PHY407 Computational Lab 10
Random Processes and Monte Carlo Integration

Recommended reading for this lab: Newman Sections 10.1-10.2.

Computational Background

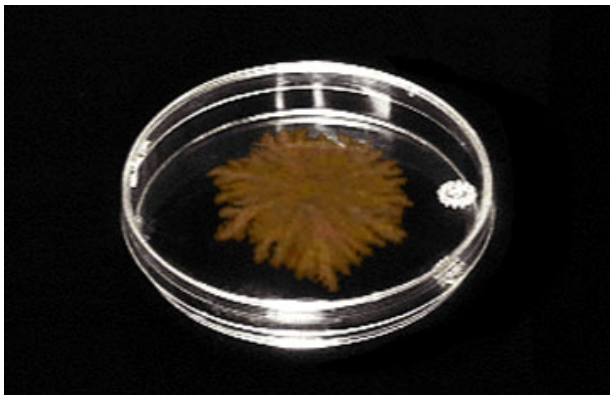
- **Monte Carlo integration** methods make use of random numbers to solve tricky integrals. In this Lab you will use "mean value" and "importance sampling" Monte Carlo integration techniques (Sections 10.2.1 and 10.2.3, with good background in Sectdion 10.2.1), along with their associated error estimates.

- **Monte Carlo simulations** provide a way to mimic natural behaviour of noisy systems with random behaviour. We will be using a Monte Carlo simulation in Question 2.

Physics Background

- **Diffusion-Limited Aggregation** (DLA) is the process whereby particles undergoing a random walk due to Brownian motion cluster together to form aggregates. 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.

Lab Instructions

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

- Q1: Hand in your code, a screenshot of the `visual` plot, and the analysis.

- Q2: Please pay attention to the following.

- For Part a, hand in the plot.
- **Do not hand in anything for Part b.**
- For Part c, hand in the plot.
- For Part d, hand in your code, the plots, any written answers.
- For Part e, hand in any plots and written answers.

- Q3: Hand in the code, plots, analysis.

1. **A random point on the surface of the Earth (from Newman 10.12, 30% of the lab):**

   Suppose you wish to choose a random point on the surface of the Earth. That is, you want to choose a value of the latitude and longitude such that every point on the planet is equally likely to be chosen. In a physics context, this is equivalent to choosing a random vector direction in three-dimensional space (something that one has to do quite often in physics calculations).

   Recall that in spherical coordinates θ, ϕ the element of solid angle is $\sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi$, and the total solid angle in a whole sphere is 4π. Hence the probability of our point falling in a particular element is

$$p(\theta,\phi) \, \mathrm{d}\theta \, \mathrm{d}\phi = \frac{\sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi}{4\pi}.$$

   We can break this up into its θ part and its ϕ part thus:

$$p(\theta,\phi) \, \mathrm{d}\theta \, \mathrm{d}\phi = \frac{\sin\theta \, \mathrm{d}\theta}{2} \times \frac{\mathrm{d}\phi}{2\pi} = p(\theta) \, \mathrm{d}\theta \times p(\phi) \, \mathrm{d}\phi.$$

   (a) What are the ranges of the variables θ and ϕ? Verify that the two distributions $p(\theta)$ and $p(\phi)$ are correctly normalized—they integrate to 1 over the appropriate ranges.

   (b) Find formulas for generating angles θ and ϕ drawn from the distributions $p(\theta)$ and $p(\phi)$. (The ϕ one is trivial, but the θ one is not.)

   (c) Write a program that generates a random θ and ϕ using the formulas you worked out. (Hint: In Python the function `acos` in the `math` package returns the arc cosine in radians of a given number.)

   (d) Modify your program to generate 500 such random points, convert the angles to x, y, z coordinates assuming the radius of the globe is 1, and then visualize the points in three-dimensional space using the `visual` package with small spheres (of radius, say, 0.02). You should end up with a three-dimensional globe spelled out on the screen in random points. This is a point where it is a lot easier to use the `visual` package. Example code is

```
from visual import sphere
...
#for each sphere to be plotted call the following
for i in range(500):
    #...stuff
    sphere(pos=[xpos,ypos,zpos],radius=0.02)
```

Take a screenshot to get your printout.

2. **Brownian motion and diffusion limited aggregation (40% of the lab):**

   (a) Brownian motion is the motion of a particle, such as a smoke or dust particle, in a gas, as it is buffeted by random collisions with gas molecules. Make a simple computer simulation of such a particle in two dimensions as follows. The particle is confined to a square grid or lattice $L \times L$ squares on a side, so that its position can be represented by two integers $i, j = 0 \ldots L - 1$. It starts in the middle of the grid. On each step of the simulation, choose a random direction—up, down, left, or right—and move the particle one step in that direction. This process is called a random walk. The particle is not allowed to move outside the limits of the lattice—if it tries to do so, choose a new random direction to move in.

   Write a program (or simplify the program in Part b) to perform a a few thousand steps of this process on a lattice with $L = 101$ and use `pylab` (or `visual`) to make an animation on the screen of the position of the particle. (We choose an odd length for the side of the square so that there is one lattice site exactly in the center.) Then create a plot that plots the path of the particle.

   Note: If you choose to use `visual`, note that this package doesn't always work well with the `random` package, but if you import functions from `visual` first, then from `random`, you should avoid problems.

   (b) In the program DLA-example.py is a program that extends the Brownian motion simulation of Part a) to carry out a DLA simulation of a single particle. This will serve as a starting point for your next DLA program in Part c). The program is set up to to perform the DLA process on a 101×101 lattice—we choose an odd length for the side of the square so that there is one lattice site exactly in the center. It repeatedly introduces a new particle at the center and has it walk randomly until it sticks to an edge or an anchored particle.

   Run this program for a while and observe what it does. You don't need to hand in anything for this code.

   (c) In the interests of speed, change the program in Part b) so that it shows only the anchored particles on the screen and not the randomly walking ones. That way you need update the pictures on the screen only when a new particle becomes anchored.

   Set up the program so that it stops running once there is an anchored particle in the center of the grid, at the point where each particle starts its random walk. Once there is a particle at this point, there's no point running any longer because any further particles added will be anchored the moment they start out.

   Run your program and see what it produces. Hand in this plot.

   (d) The main part of this lab is to alter the program in Part b) by simulating the behaviour of the aggregation process in a doubly periodic domain with a single anchor point in the centre. Make making the following changes:

   - The centre point of the domain is the only initial anchor point. Particles are no longer anchored at the edges of the domain.

- Particles are launched randomly anywhere in the domain that is not an anchor point.
- If a particle reaches the top of the domain immediately map it to the bottom of the domain. Similarly, if a particle reaches the right edge of the domain immediately map it to the left edge.
- Otherwise the rules of aggregation are the same. Particles move until they end up adjacent to an anchor point, where they become anchored.

Thus the first particle has to wander around until it reaches the centre anchor point. This can potentially take a long time, so it's a good idea to experiment with a smaller domain (e.g. 51 x 51) and a few particles (e.g. 10-20) to see how the simulation works. It's useful to print something out when a particle becomes stuck, to monitor progress.

With the remapping suggested above, the program doesn't work well if the cluster grows too close to the edge, which can happen as you add points. So if you encounter this behaviour you can increase the size of the domain a bit to accommodate a larger cluster. One could alter the program to allow the cluster to connect to itself across the domain boundaries, but don't do that for this lab.[1] Start with a simulation

on a 101 x 101 grid and increase the number of points until you get a robust structure. If you want, you can also increase the domain size further (to, for example, 201 x 201). Describe what you see. Why is the centre of the domain not more "filled in"?

(e) The DLA structures arising from this exercise have a so-called *fractal dimension*. To explain, if the growth were linear with just a few kinks, you would find that the total number of points within a circle of radius l or square of side l would be proportional to r. Call the total number of points m (for *mass*), we would find that, to a rough approximation, $m \propto l^k$, $k = 1$. If the growth filled out all the points in the domain, then we would expect $m \propto l^k$, $k = 2$. The fractal dimension of the cluster is a scaling parameter k such that $m \propto l^k$. In DLA, the cluster is not quite linear and not quite space filling, and this means that we expect k to fall somehwere between 1 and 2. A property of fractal systems is that we observe $\log m = k \log l + C$ behaviour, with k being fairly constant over a wide range of l. In this final part of the lab we will explore this behaviour.

Save the output from one of your runs of the model in Part c). Starting at the centre, create a square of length l units centered at the center of the box, and count the number of anchored points within the square. *Hint: which array in the program keeps track of the anchored points?* This gives you m for one value of l. Now create a loop that calculates m for each l, and plot this on a log-log scale. You will find that for some $l > l_c$, m will not change any more: the square that is l_c on each side contains all the points in your simulation. Finally, plot $\log m$ versus $\log l$ and find, either graphically

[1] You can imagine the doubly periodic domain as a connected torus or a doughnut. Then the cluster can be imagined as the growth of some mould from an initial spore, which could extend and potentially wrap around the doughnut and connect with itself. Ok, perhaps that's not the most appealing thought ... must be getting late in the semester.

or by a least squares fit, the fractal dimension k. The dimension is sensitive to the range of r you use for the fit, so choose a reasonable one where l is quite a bit less than l_c and the slope is reasonably constant in the log-log plot. The dimension will also be somewhat sensitive to the domain and the number of points you choose. You will not have time to thoroughly explore these sensitivities, but try a couple of tests to see how your answers differ.

3. **Importance Sampling (30% of the lab):**

   (a) 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.

   (b) 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 it's 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.

   (c) 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

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

   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 above. Comment on what your histograms from the mean value method and the importance sampling method tell you about the methods.