PHY407 Lab-10 Report

Patrick Sandoval[1,2] and Chin Chong Leong[3]

[1] *Q1 - Brownian Motion and Diffusion Limited Aggregation*
[2] *Q2 - Volume of Hyper-sphere in 10 Dimensions*
[3] *Q3 - Time Dependent Schrodinger Equation*

1. Q1 - BROWNIAN MOTION AND DIFFUSION LIMITED AGGREGATION

For the python script containing the code and plots for this question and all its consequent sub-questions please refer to `Lab10_Q1.py` where we use some function defined in `MyFunctions.py`.

1.1. *Simulating 2-Dimensional Brownian Motion*

If we assume a particle undergoing Brownian motion has an equal probability to move in any spatial direction (up, down, left or right) then we can model its random walk through a random sampling method where we assign equal probability values some direction. Our simulation will conduct a random walk of 5000 steps where each step interval is equal to some time interval Δt. The results of the simulation can be seen in Figure 1

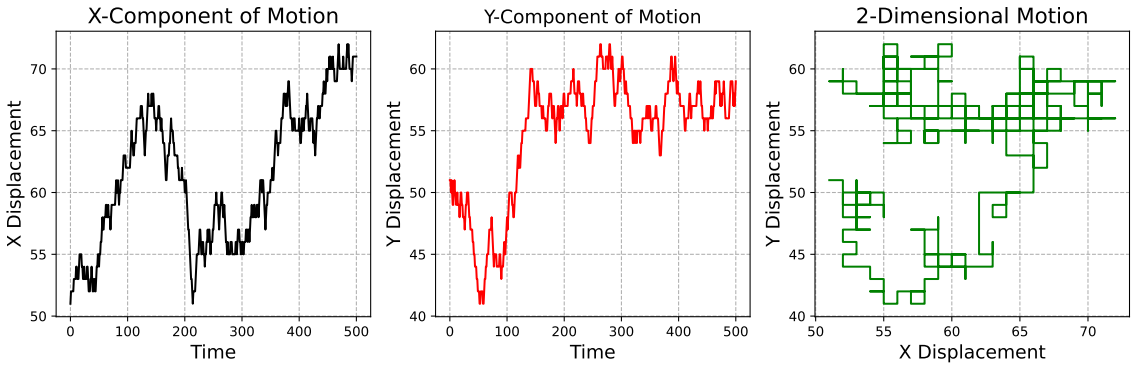

Figure 1: Random walk trajectory of particle undergoing Brownian motion. Where particle starts at the center of a 101×101 grid and cannot move outside the solid boundary.

1.2. *Simulating Diffusion-Limited Aggregation System*

Now we will simulate a similar system to that of subsection 1.1, but with the catch that particles can get anchored to the edges of the grid and to other particles. The algorithm in this case is a little different because we need to create a condition for the particles to stick together, and for them to also stick to the edges. We can do this by representing each particle with the number 1 and representing the walls of the grids with 1's too. This way we can create a generalizable condition that tells a particle to stop moving once there is a 1 in an adjacent box. Once this condition is triggered we generate a new particle in the center until the new particle in the center detects a 1 adjacent to itself.

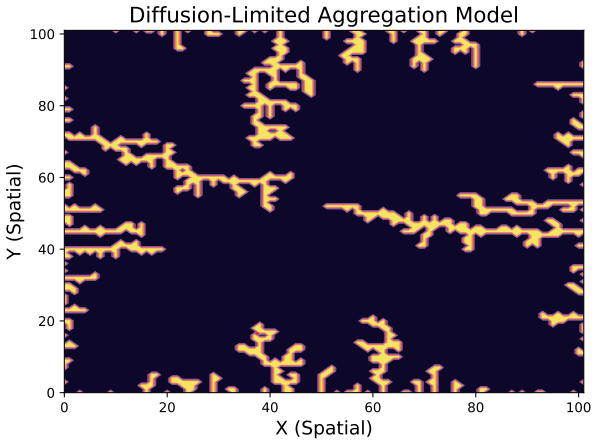

Figure 2: Diffusion-Limited Aggregation (DLA) system for particles undergoing random walk in a 101×101 grid.

2. Q2 - VOLUME OF HYPER-SPHERE IN 10 DIMENSIONS

For the python script containing the numerical calculations for the volume of the hypersphere please refer to the script `Lab10_Q2.py`.

2.1. *Using Monte Carlo Integration*

The analytical expression for the volume of a sphere of radius R in n dimensions is given in eq.1, which will be used to compare the accuracy of our method.

$$V_n(R) = \frac{R^n \pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} \tag{1}$$

Additionally, to further test our program we generalized to compute the 'volume' for an arbitrary degrees of freedom such that we could test our program for computing the area of a circle i.e d.o.f = 2 and the volume of a sphere i.e d.o.f = 3. From Table 1 we see that the percent error for the MC integration method is 0.009%, 0.076%, and 0.934%

d.o.f	Volume/Area (MC)	Volume/Area (Analytical)
2	3.140	π
3	4.192	$(4/3)\ \pi \approx 4.188$
10	2.574	$\pi^5/5! \approx 2.550$

Table 1: Result of Monte Carlo integration of sphere in two, three and ten dimensions. Values where computed by sampling 1 million points

for 2, 3 and 10 dimensions respectively. Lastly we note that the function used for the integration scheme was the equation of a sphere generalized to higher dimensions.

$$f(r) = \begin{cases} 1 & \text{if } \sum_i^n x_i^2 < R^2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

3. Q3 - IMPORTANCE OF SAMPLING

For the python script containing the code that implements the mean value method and importance sampling method for this questions please refer to `Lab10_Q3.py`

3.1. *Q3a - Mean Value Method*

For this exercise, we define the integrand as:

$$f(x) = \frac{x^{-\frac{1}{2}}}{1+e^x} \tag{3}$$

Using the `random()` function in python, we generated the $N = 10000$ random numbers x_i in the range between $a = 0$ and $b = 1$. Since the sampling is uniform, we follow equation 10.30 in the textbook to estimate the value of the integral via mean value method.

$$I \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \tag{4}$$

By repeating the calculation 100 times, we take the average and obtained $I = 0.84010$

3.2. *Q3b - Importance Sampling Method*

For this part of the exercise, we used the sampling function $w(x) = x^{-\frac{1}{2}}$ in order to remove the singularity. From this we can define the probability function:

$$p(x) = \frac{w(x)}{\int_a^b w(x')dx'} = \frac{1}{2\sqrt{x}} \tag{5}$$

Since we are now sampling $N = 10000$ points x_i non-uniformly with this density function, we need to choose the sample points x such that it satisfies equation 10.7 in the textbook:

$$z = \int_0^z dz' = \int_{-\infty}^{x(z)} p(x')dx' = \int_0^{x(z)} \frac{1}{2\sqrt{x'}}dx' = \sqrt{x} \tag{6}$$

$$\implies x = z^2 \tag{7}$$

(i.e. we have to use sample the random numbers using `random()**2` in python, see the code for implementation). From this we again generated $N = 10000$ random numbers between $a = 0$ and $b = 1$. By equation 10.42 in the textbook we estimate the value of the integral using the importance sampling method:

$$I \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{w(x_i)}\int_a^b w(x)dx = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{8}$$

By repeating the calculation 100 times, we take the average and obtained $I = 0.83917$

3.3. *Q3c - Comparison of Histogram*

We compared the distribution of the 100 calculated integral values from either method in Figure 3 below. From the histogram we can see that the calculated values from the mean value method is much more spread out around the actual value of the integral (which is around 0.839), while the calculated values from the importance sampling method is much more concentrated and accurate. (This is due to the nature of the integrand $f \to \infty$ as $x \to 0$, so more unweighted sample the program takes from the small x regions, the less accurate the calculated value is.)

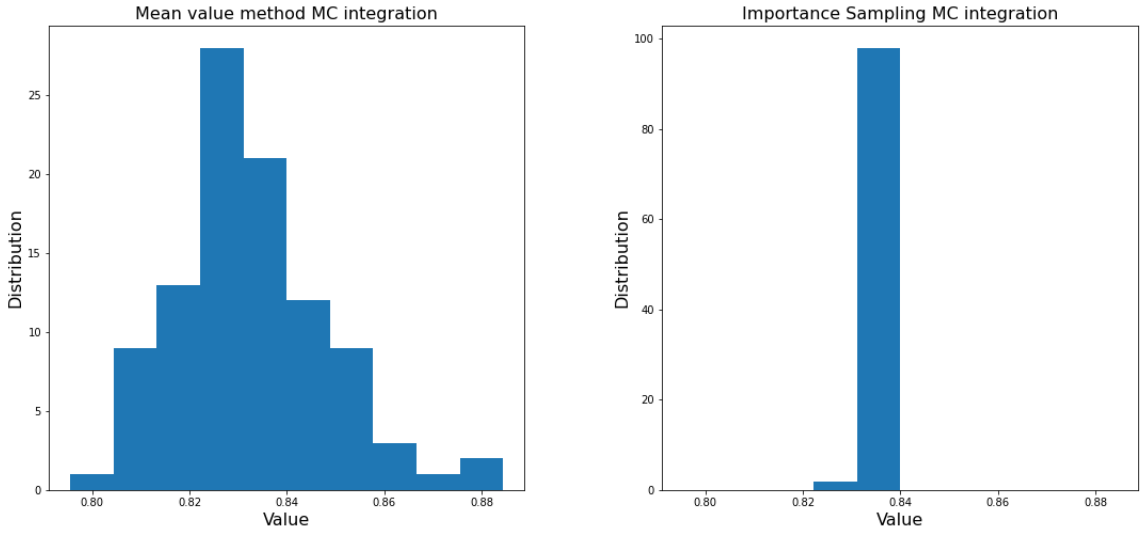

Figure 3: (Left) Distribution from mean value method Monte Carlo integration; (Right) Distribution from the importance sampling method Monte Carlo integration.