PHY407 Lab10

(Yonatan Eyob Q2,Q3), (Kivanc Aykac Q1)
Student Numbers: 1004253309, 1004326222

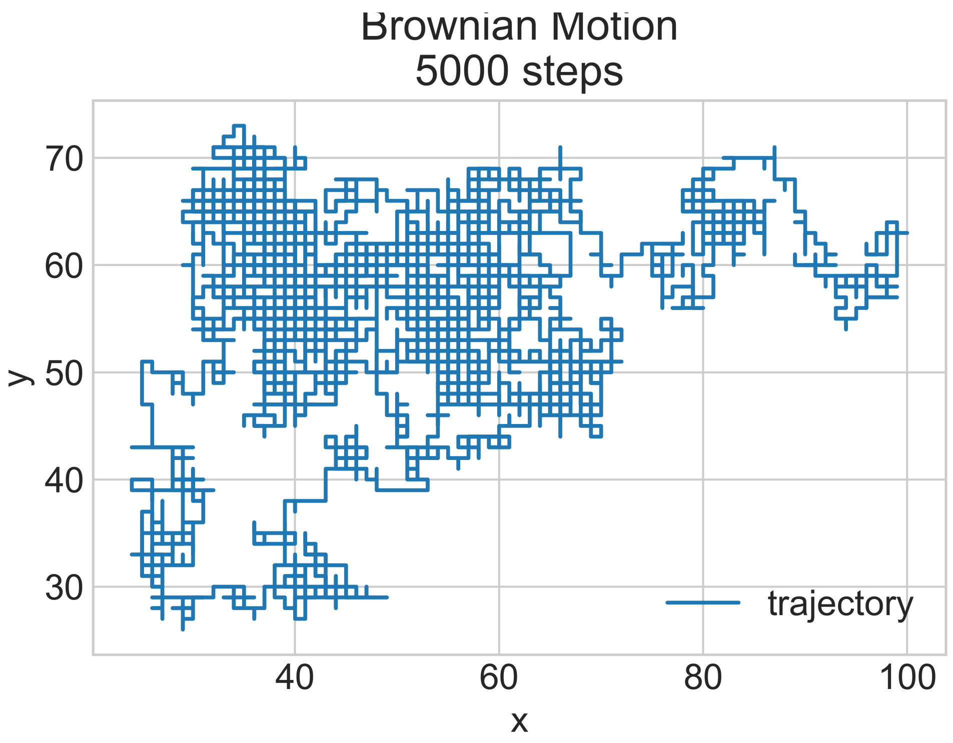
November 27th, 2020

1 Brownian motion and diffusion limited aggregation

In the below exercises the Brownian motion under diffusion-limited aggregation will be examined.

1.a

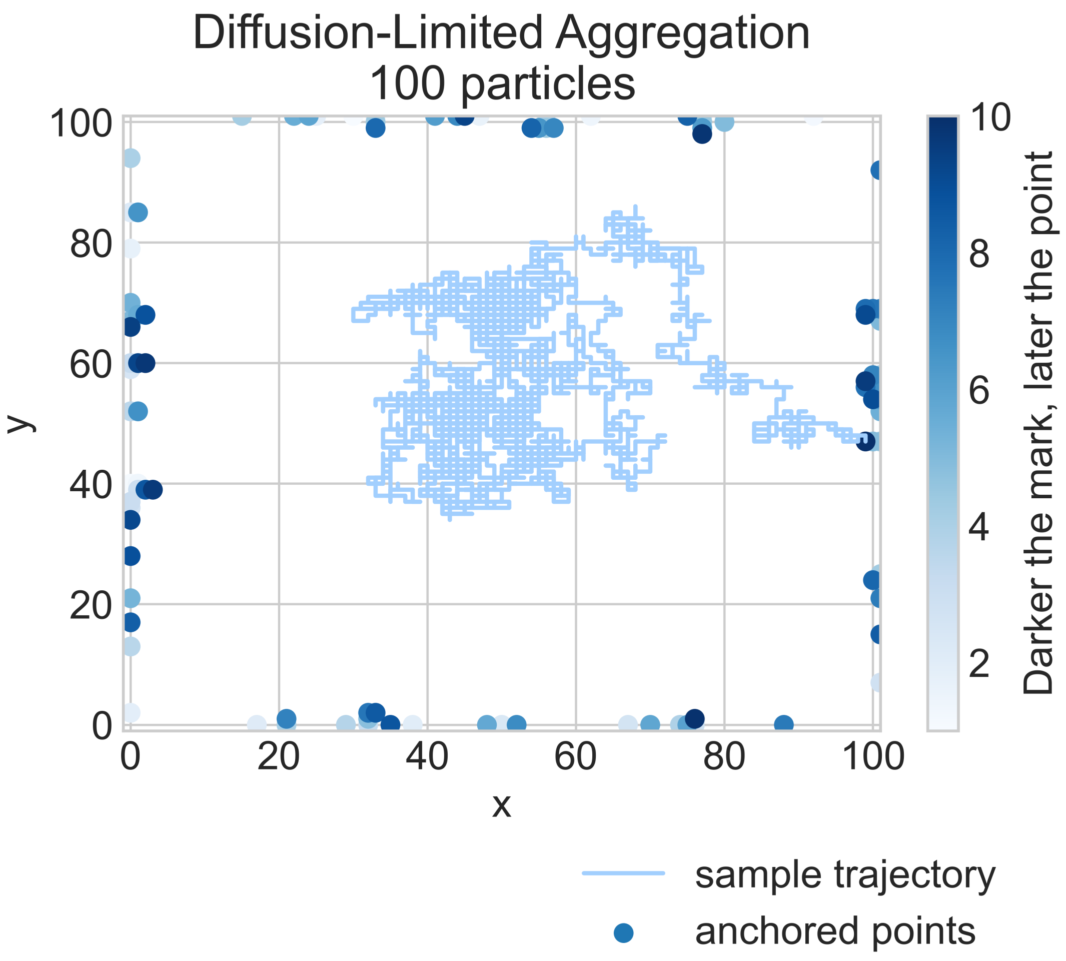
In this part Brownian motion for a single particle is going to be figured out under 5000 time-steps. The size of the domain is 101x101 grids. The `random` library uses *Mersenne twister*, so the trajectory at each execution of the code is different. To make sure particle stays within the boundaries, `checking()` function was used, which checks each iteration for its validity. If iteration is out of bounds, it is done again until it is valid. Below is the figure that shows the trajectory:



Since the step size is fairly large, the trajectory looks quite fuzzy. But if looked closely, it is visible that the trajectory, indeed, does not exceed the boundaries. In this execution the boundary that was significant was $x = 0$, and it is visible that trajectory doesn't go below it. So the key goal here was achieved!
From the output, we understand that main part of the code took 0.033 sec.

1.b

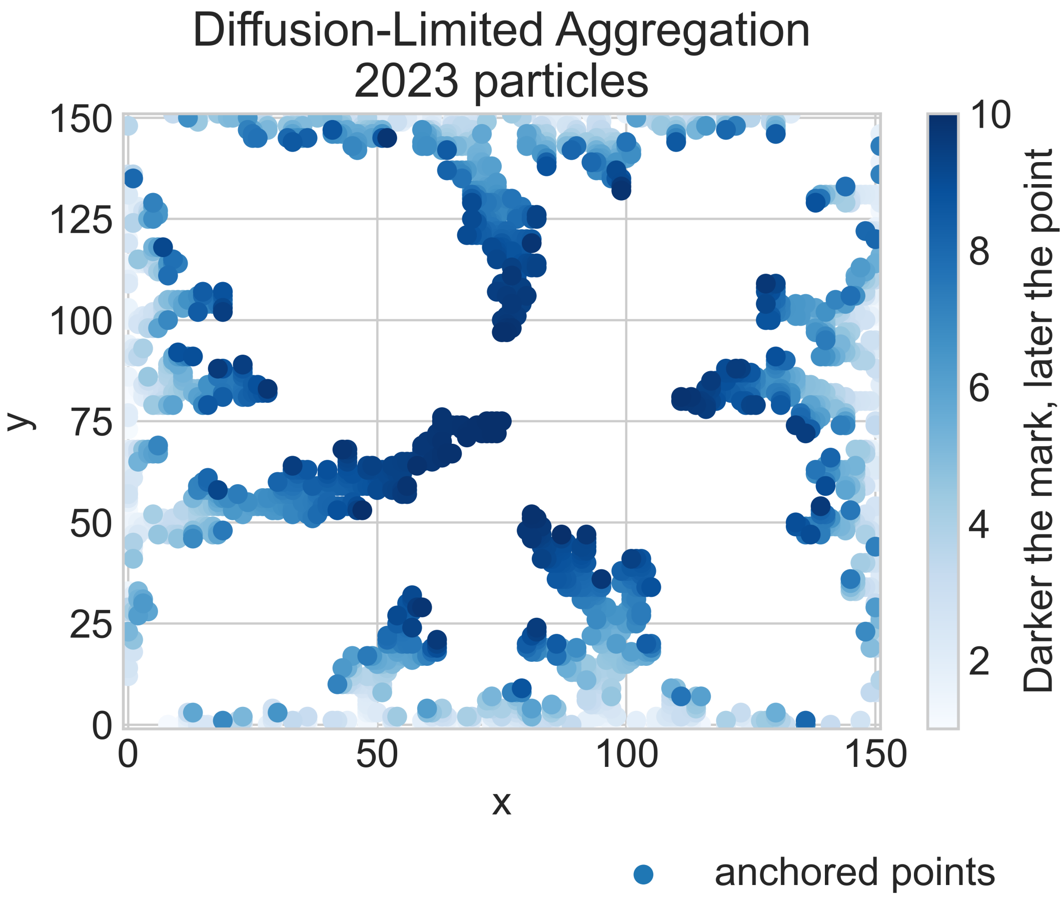
In this part, the same principle of motion in 1.a will be followed. But this time, there will be 100 particles and they will stop when the hit the boundaries or other anchored particles. Below is the figure of this case:



For a better reading, shading was used here. So the darker blue marks correspond to later generated values. This will come handy in 1.c. Also the trajectory of the last particle was plotted in pale blue to give a sample for trajectory.
As can be seen all 100 particles stopped their trajectory when they hit the boundaries (The one dot that is not on the boundary is actually the legend, it shall not misguide the reader).
Finally this part's main part (figuring out the trajectories and the anchored points) took 1.133 sec. This more than the run-time of 1.a, as expected, because now the `checking()` function has been expanded and there are now 100 particles, not 1. Also, when looked closely it could be seen that some of the anchored points actually accumulated on top of each other, which proves that the anchoring function works fine.

1.c

In this part the above part is going to be repeated, except this time the number of particles will not be defined. The program will terminate when an anchored point is at the middle of the grid. Below is the figure that shows the final result with all anchored points:



This part's code's main part (figuring out the anchored points) took 41.382 sec. This is normal because the for loop that was used to go over `N` particles is now a while loop that checks if the final anchored point is in the middle and there are also many more particles (2023). Looking at the figure, the final point was at the centre, therefore the program does what it was meant to to do. Also is very interesting to stare at as well with the shading of the scatter plot. The darker points mean they were generated later.

2 Volume of a 10-dimensional Hypersphere

In this section we used The Monte Carlo integration method to estimate the integral of a ten dimensional hypersphere. Our method was to generate random ten dimensional points $\vec{\boldsymbol{r}}$ =(q,w,e,r,t,y,u,i,o,p) that were all bounded inside a ten dimensional hypercube centered around the origin with side lengths equal to 2, count all the points that landed inside the radius of $R = \sqrt{q^2 + w^2 + e^2 + r^2 + t^2 + y^2 + u^2 + i^2 + o^2 + p^2}$ and divide that by the total number of random points (in this case we used 1 million points), and then multiplied this division with the volume of the hypercube to get our estimated integral.
After using this method we find that the estimated solution to the volume of the Ten dimensional hypersphere is 2.54976 units, with an error of 0.051033841359850625.

3 Importance sampling

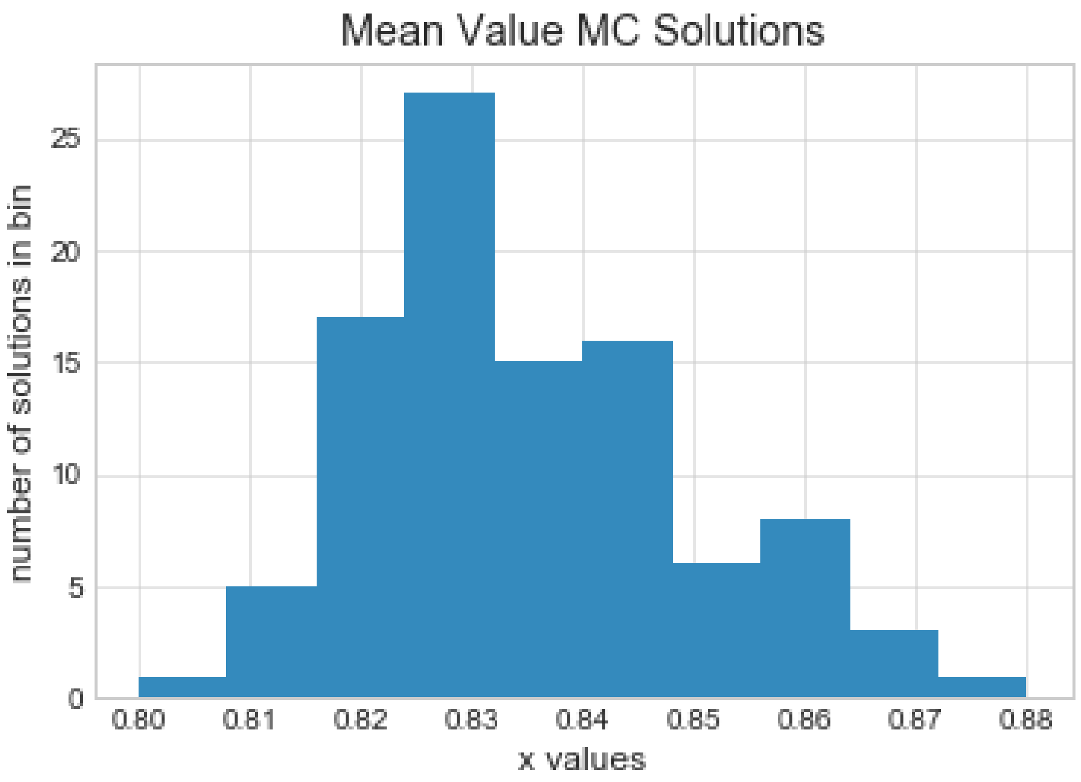
In this section we find the integral of a function using the Mean value MC and Importance sampling MC, and then compare the accuracy of both methods by running each method 100 times and plotting their solutions on a histogram to see how much the solutions vary for each method.

3.a

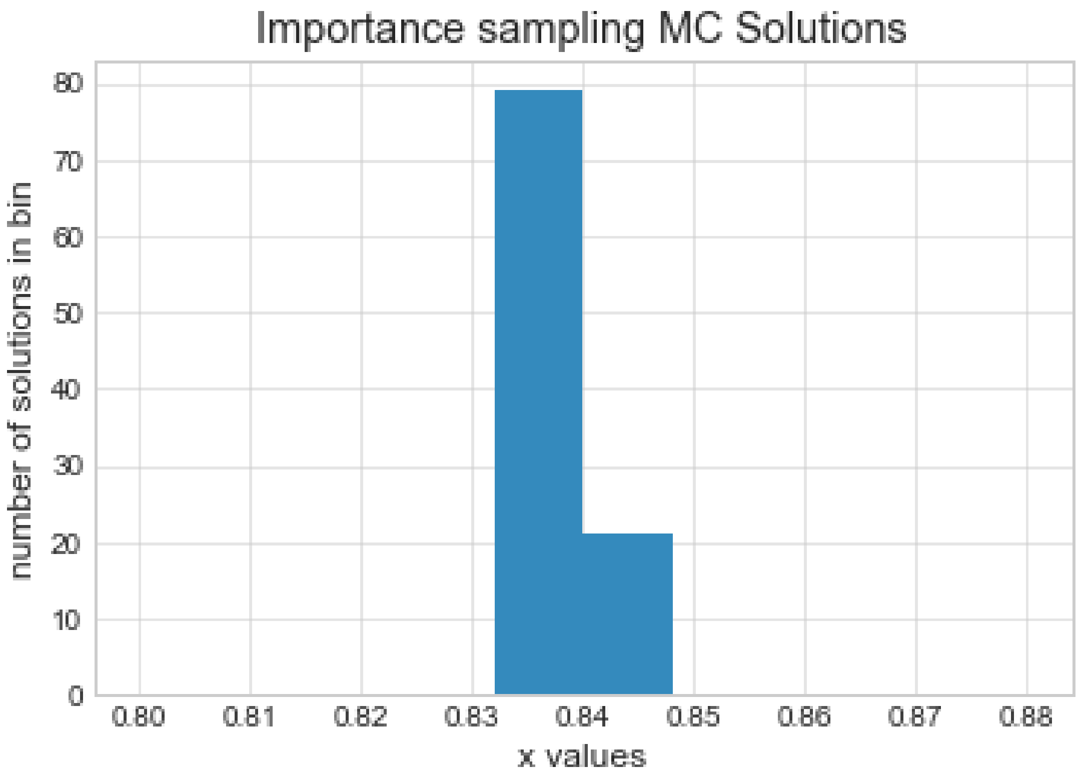
The first integral we will deal with is:

$$\int_0^1 \frac{x^{-\frac{1}{2}}}{1+e^x}\,dx$$

first we will create the histogram of 100 solutions using the Mean Value MC:



Now we will create the histogram of 100 solutions using the importance sampling MC:



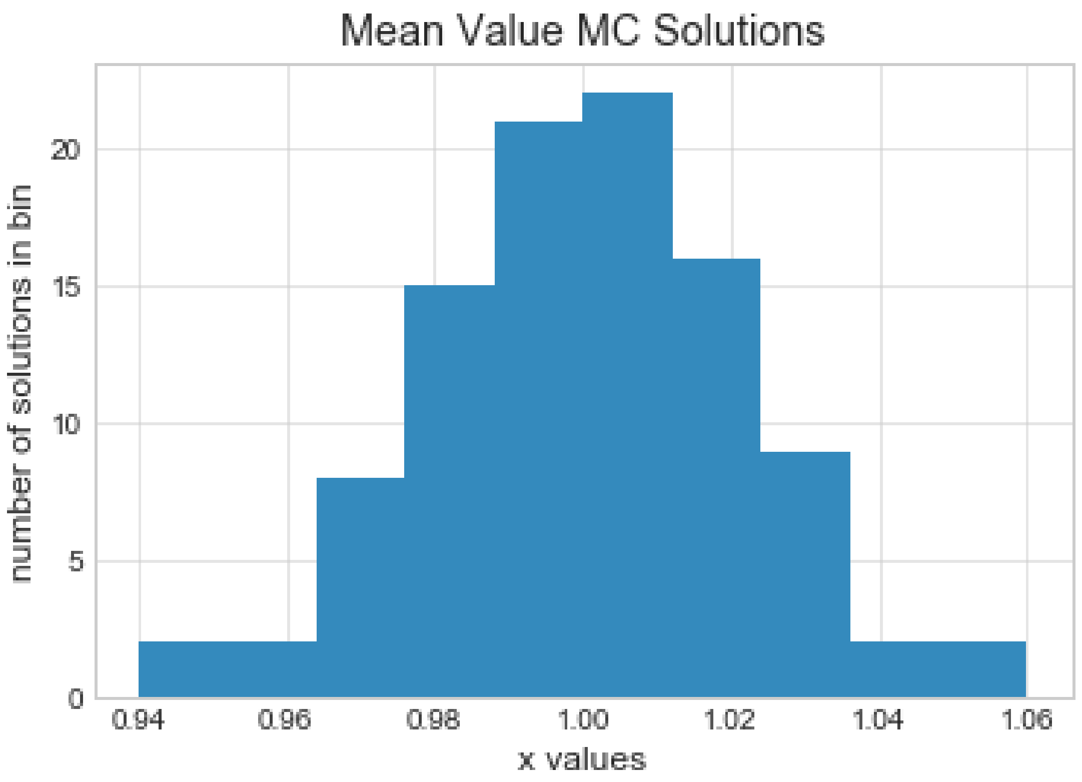
If the histogram requires more bins to hold all of the solutions, then that means the solutions vary from each other more (which means the solutions are less consistent). When looking at the two histograms, it is clear that the importance sampling method has more consistent solutions than the mean value method because the mean value histogram has ten bins and the importance sampling histogram has two bins.

3.b

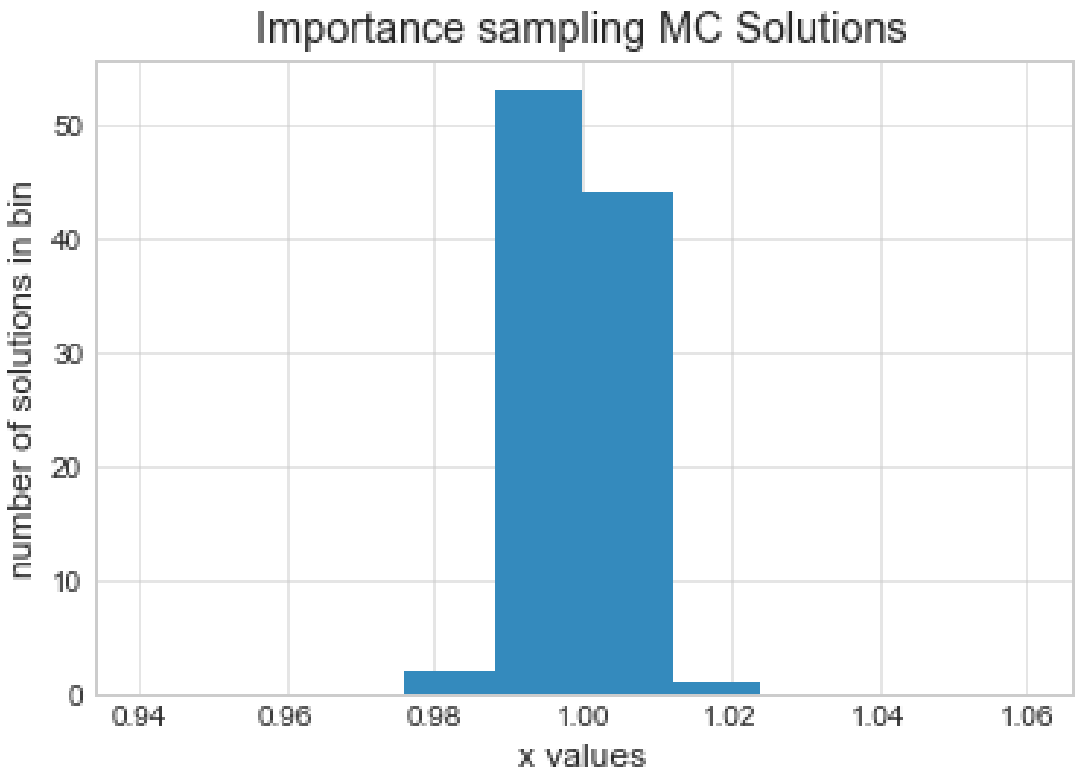
The second integral we will deal with is:

$$\int_0^{10} e^{-2\|x-5\|}\,dx$$

first we will create the histogram of 100 solutions using the Mean Value MC:



Now we will create the histogram of 100 solutions using the importance sampling MC:



Again, when looking at the two histograms, it is clear that the importance sampling method has more consistent solutions than the mean value method because the importance sampling histogram has four bins and the mean value histogram has ten bins (same reason as 3a).