PHY407 Lab10

Genevieve Beauregard (1004556045)
Anna Shirkalina (1003334448)

Oct 2020

Q1

a)

We are asked to simulate a random walk for 5000 times steps, with the initial starting point for a particle in the middle of a 101 by 101 grid. At each time step we generate a random number using `Numpy.random.random()` which returns a random float number between [0, 1), we then use the random number to move the particle right, left, up or down. When a particle hits the wall and tries to exit the boundaries we don't let it, and wait until the random number indicates a valid move. Fig 1. Our code is available in `Lab10_Q1a.py`.

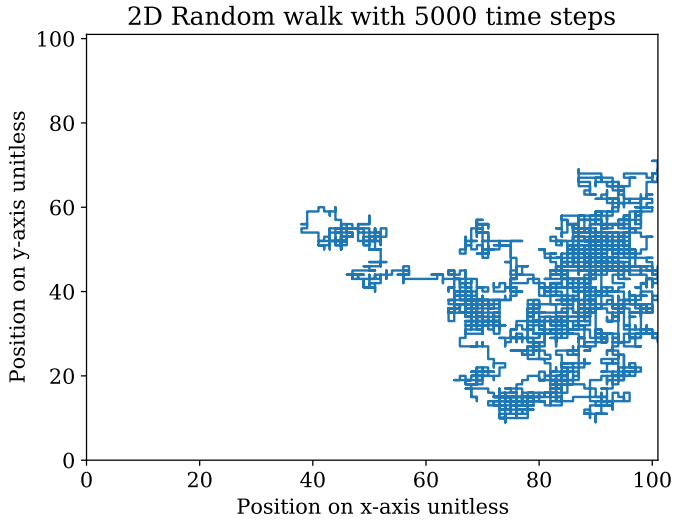

Figure 1: The Brownian random motion for a particle starting in the middle of the grid.

b)

We are asked to make an DLA algorithm where each particle starts in the middle of the board and stops moving when it comes into contact with another particle or the boundary. We run the simulation for 100 particles. Fig 2 Our code is available in `Lab10_Q1b.py`.

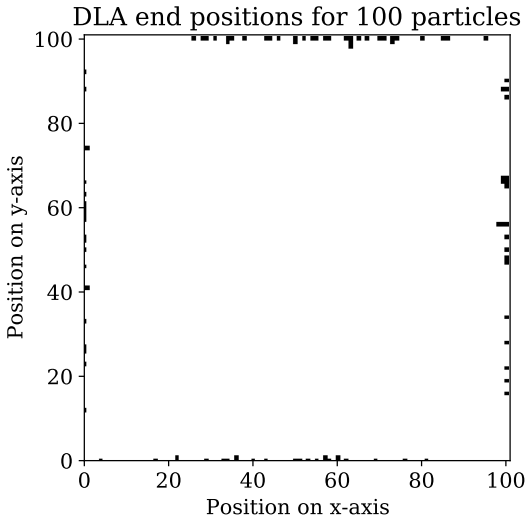

Figure 2: The end state of DLA algorithm with 100 particles each of which starts in the middle and sticks to one of the walls as the end point.

c)

We are asked to run a simulation until on particle freezes in the middle of the board (151 X 151 points) which is the starting point for all particles. Our code is available in `Lab10_Q1c.py`.

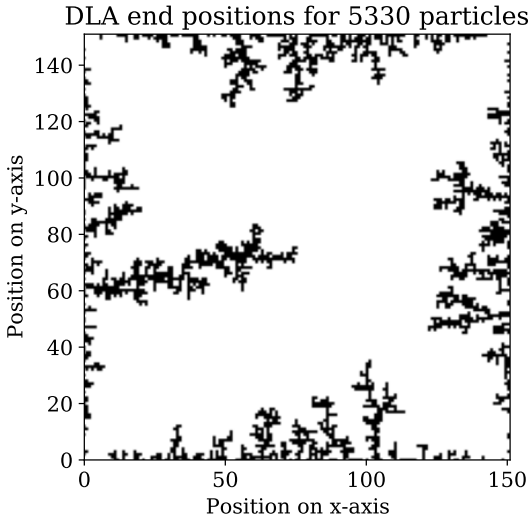

Figure 3: The end state of DLA algorithm when a particle gets stuck at the middle of the board.

Q2

We are asked to generalize a Monte Carlo routine used to compute the area of a circle to 10-dimensional hypersphere. Our code for this question can be found in `Lab10_Q2.py`.

We considered a n-dimensional hypersphere centred at the origin We define a function f that takes in a n-dimensional coordinate r, such that

$$f(r) = \begin{cases} 1 & |r| \leq 1 \\ 0 & |r| > 1 \end{cases}$$

It should be clear that $\int_{\text{hypercube}} f(r)\mathrm{d}r$ would return the volume of the hypersphere.

Using this function, we will numerically compute the volume of a unit sphere using a Monte Carlo Method.

Adapting the Monte carlo method, for an n-dim hypersphere,

$$I \simeq \frac{2^n}{N} \sum_{i=1}^{N} f(\vec{r})$$

We run this for $N = 10^6$ points for a circle (n = 2) and a 10-dimensional hypersphere (n = 10).

To obtain the error we simply use:

$$\text{error} = |I - \text{Actual Volume}|$$

Where the actual volume is the analytically computed volume, and I is the approximated volume by Monte Carlo.

We compute the actual volume

- For the unit circle, $V = \pi$.
- For the hyphersphere, $V = \frac{\pi^{10/2}}{\Gamma(10/2+1)} = \frac{pi^5}{120}$ (we just use the scipy function in the script).

For one particular run of the code, our results are summarised in table 1.

n	Approximated Volume	Actual Volume	Error
2	3.138364	3.141593	0.003229
10	2.456576	2.550164	0.093588

Table 1: Approximated volume for n-dimensional hypersphere using Monte Carlo methods with a million samples, and the associated error against the actual volume, to 6 s.f. for readability. This is based on the output below.

Generally the error for the Monte Carlo approximation is an order of magnitude higher for the 10-dimensional hypersphere than the circle (save for a few edge cases).

Pseudocode

```
Define function f

Define function ObtainVol:

    For loop over N samples
        Initialize count float variable = 0

        Obtain n-dim r vector with random entries between 0 and 1

        Count += f(r)

    Return n*2/N * count
```

Output

```
Using N=1000000 points for Monte Carlo integration
The approximated volume of a unit circle is 3.1383639999999997
The associated error is 0.0032286535897934066

The approximated volume of a 10-dim unit hypersphere is 2.456576
The associated error is 0.09358803987734499
```

Q3

Our code for this question can be found in `Lab10_Q3.py`.

a)

We are asked to numerically determine the integral

$$\int_0^1 \frac{x^{-1/2}}{1+e^{-x}}\mathrm{d}x$$

using both importance sampling and mean value Monte Carlo methods. We implement both routines for $N = 10,000$ samples.

Importance sampling

The weighting function we are asked to draw from is:

$$w(x) = x^{-1/2}$$

The associated probability distribution is:

$$p(x) = \frac{w(x)}{\int_0^1 w(x)\mathrm{d}x} = \frac{1}{2\sqrt{x}}$$

for $x \in [0,1]$ ($p(x)$ is implicitly zero otherwise).

For importance sampling, the integral $I = \int_a^b f(x)\mathrm{d}x$ can be approximated using weighting function $w(x)$ as follows:

$$\begin{aligned} I &\simeq \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{w(x_i)}\int_a^b w(x) \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \end{aligned}$$

where x_i is the sampled points in $[a,b]$ from probability distribution $p(x)$. We create this routine for a general computation $\int_a^b f(x)\mathrm{d}x$ in a function `ImportSampMontecarlo`.

We can generate from a random variable z from a uniform distribution using the function `random()`. To obtain $x(z)$ sampled from the distribution $p(x) = \frac{1}{2\sqrt{x}}$, we need to do the transformation method. Adapted from the method in the textbook pg 459:

$$\int_0^{x(z)} p(x')\mathrm{d}x' = \int_0^z z'\mathrm{d}z' = z$$

$$\int_0^{x(z)} \frac{1}{2\sqrt{x'}}\mathrm{d}x' = z$$

$$-\sqrt{x} = z$$
$$x = z^2$$

We create function `generaterandom_a()` which implements the above transformation from $z =$ `random()`, sampling between 0 and 1.

We then calculate the integral using the importance sampling routine as shown earlier:

$$I = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

where $f(x) = \frac{x^{-1/2}}{1+e^{-x}}$ and $p(x) = \frac{1}{2\sqrt{x}}$.

More information and documentation can be seen in our python script for this question. We compute the integral 100 times and plot the number of occurrences integral values I against computed integral values I, between $I = 0.8$ and $I = 0.88$ into 10 bins. Our plots can be seen in Figure 4.

Mean Value Method

We implement mean value routine. From the textbook, for an integrand $f(x)$:

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

for x uniformly sampled within the integration region.

In this case, $f(x) = \frac{x^{-1/2}}{1+e^{-x}}$, $N = 10,000$, $a = 0$, $b = 1$.

We obtain x from `random()` as per a uniform distribution. We implement the mean value routine in the function `MeanValueMontecarlo`.

We run this routine 100 times and plot our results into a histogram between $I = 0.8$ and $I = 0.88$ with 10 bins. Our plot can be seen in Figure 5.

Figures

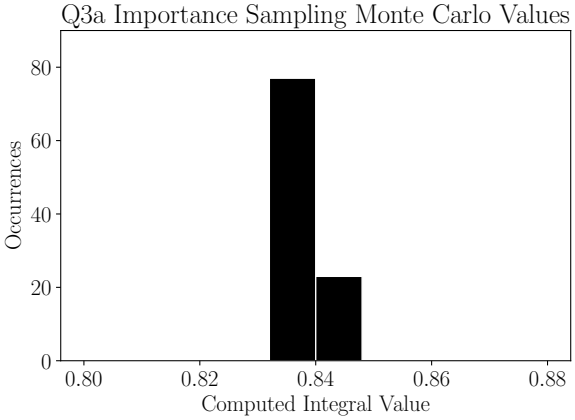

Figure 4: The integral values obtained using 100 tries of the Importance Sampling for Monte Carlo integration.

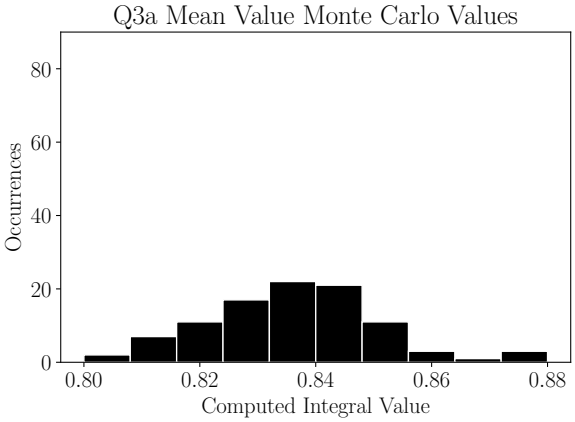

Figure 5: The integral values obtained using 100 tries of the importance sampling for Mean Value Monte Carlo methods.

In Figure 4, we note that there is a clear spike of computations obtaining values between I = 0.832 and 0.84, and some runs that obtained between 0.84 and 0.48. We can see that there is a clear consistency and smaller variance in values as compared to the the mean value runs in Figure 5. For the mean value method, we see a high variance in the numerical integral approximations. This is to be

expected - the integrand diverges at certain points and thus the mean value method does not work.

b)

We are now asked to numerically compute the integral:

$$\int_0^{10} \exp(-2|x-5|)\mathrm{d}x$$

using same routines as part (a).

Of course we can compute this integral analytically for reference:

$$\begin{aligned}
\int_0^{10} \exp(-2|x-5|)\mathrm{d}x &= \int_0^{5} \exp(2(x-5)) + \int_5^{10} \exp(-2(x-5)) \\
&= \frac{1}{2}\exp(2(x-5))\Big|_0^5 - \frac{1}{2}\exp(-2(x-5))\Big|_5^{10} \\
&= 1 - \frac{1}{e^{10}} \\
&\simeq 0.99995 \\
&\simeq 1
\end{aligned}$$

Importance Sampling

We sample x_i from a normal distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2}$$

with the associated weighting function

$$w(x) = \frac{1}{\sqrt{2\pi}} e^{-(x-5)^2/2}$$

where the mean is 5 and the standard deviation is 2. We use function `numpy.random.normal` to generate the random number from the normal (we encoded this into a random number generator function `generaterandom_b()`.

Using the same functions and routine as part a) we obtained our histogram for 100 runs of the routine. Our x-axis is I = [0.96, 1.04] with 10 bins. Our histogram can be seen in Figure 6.

Mean Value Sampling

This is basically the same routine as part a). We use the same routine for 100 tries and plot it against the same histogram as the Importance Sampling method, where our x-axis is I = [0.96, 1.04] with 10 bins. Our figure can be see in Fig 7.

Figures

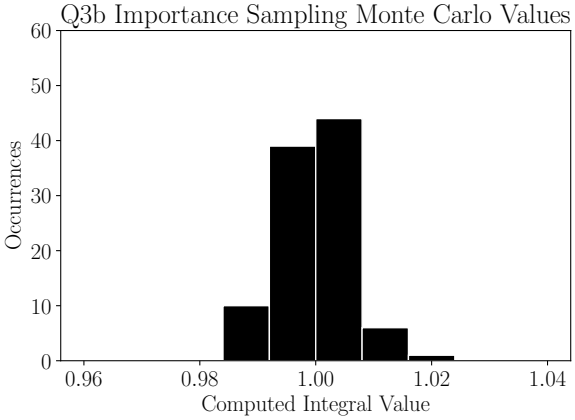

Figure 6: The integral values obtained using 100 tries of the Importance Sampling for Monte Carlo integration.

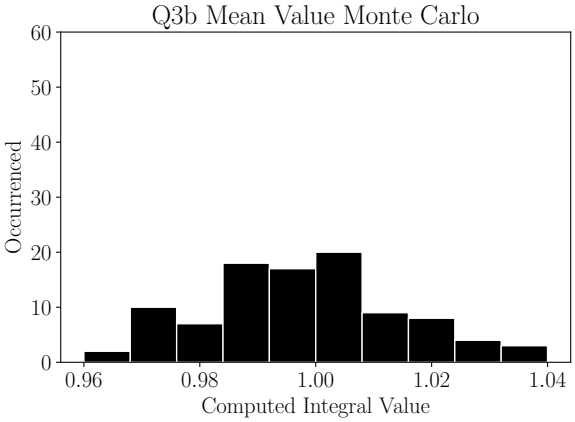

Figure 7: The integral values obtained using 100 tries of the importance sampling for Mean Value Monte Carlo methods.

We see a similar pattern to question 3a) but to a larger extreme: there is far less variance in the computed integrals for the importance sampling method as compared to the computed integral value. In fact, it seems nearly impossible to have an accurate deduction of the integral from the Mean Value method: there are no clear spikes in the bins.

However, there are clear spikes of data in the importance sampling method but, it seems to also have increases variance (and spike to the left of 1.00 which is slightly inaccurate). However, more runs and averaging the computed integrals may mitigate this effect.