Project Report

Introduction to Probability and Random Variables - L1
Dr. Musabbir Abdul Majeed
Agha Syed Nasir Mahmood Azeemi (aa04377), Moid ul Huda (mh05205), Syed Muhammad Fasih Hussain (sh05204)

1 Tasks

1.1 Task 1

Solution:
Let S be a set such that $S = \{-1,\ 0,\ 1\}$
For this task we modeled a 1-D boundless random walks using turtle in Python. The turtle would start at origin (0, 0) and moved randomly in the x-plane. To ensure randomization we assigned each step in set S a probability. Utilizing Python's built-in library numpy we randomly chose a step from the set based on it's probability. The turtle was allowed to take a 1000 steps and then it's final displacement from the origin would be saved. After repeating the experiment 10,000 times all final displacement's were plotted using a normal distribution.

The mathematical model for this task can be constructed using multi-nomial probability. We have 3 steps each whose probabilities add up to 1 and each trial is independent of the other. Thus for n number of steps/trials, the mathematical model becomes:

$$\sum_{i=0}^{n} \frac{n!}{n_1! \times n_2! \times n_3!} \times p_1^{n_1} \times p_2^{n_2} \times p_3^{n_3}$$

where $n = 1000$, n_1 is the number of times step -1 is chosen, n_2 is the number of times step 0 is chosen, n_3 is the number of times step 1 is chosen and $p_1,\ p_2,\ p_3$ are the probabilities of each step, respectively. The expected displacement from the starting position after n steps is given below:

$$E[Displacement] = -P(S = -1) \times n + P(S = 1) \times n$$

Hence the expected displaced for equally distributed probabilities is 0 shown as the mean in figure 1 for 1000 steps, while for unequal probabilities given below:

$$P(S = -1) = \frac{1}{6},\ P(S = 0) = \frac{2}{6},\ P(S = 1) = \frac{3}{6}$$

$$E[Displacement] = -\frac{1}{6} \times 1000 + \frac{3}{6} \times 1000 = 333.33$$

The expected displacement is also the mean in figure 2.

1.2 Task 2

Solution: Building on task 1, to compute the expected time required we initialize two nodes, one at point x while other one at point 0, and update their positions according to given probabilities. Updating in this, the distance between the nodes keeps on increasing in many cases and they don't converge. Instead if reinitialize the node if they are at a distance of more then $2x$ from each other than it gives them room to move away as

well but makes sure that they converge at point. Figure 3 shows the mean time for the node to converge over a range of values of x (1000 runs each) and the trends can be fitted to a polynomial function. Following is the function for the expected value obtained using fitting a quadratic function in our data:

$$E[X = x] = 0.7769x^2 + 0.9367x + 0.6088$$

This function is based on our assumption that the expected time would be a second order polynomial based on it's trends as shown in figure 3.

1.3 Task 3

Solution:
Let S be a set such that $S = \{0,\ 0.5,\ 1\}$ & let O be a set of the closed interval $[0 - 2\pi]$
For this task we modeled a 2-D bounded random walk using turtle in Python. The turtle would start at origin $(0,\ 0)$ and moved randomly in the xy-plane. To ensure randomization in step, we assigned each step in set S with uniform probability. To ensure discrete orientation from the continuous set O we divided the set in 4 intervals (e.g 0-90) and fixed 1 integer value to be used for each interval. randomization in orientation we chose. Utilizing Python's built-in library numpy we randomly chose a step from set S and chose an orientation from set O based on their probability. The turtle was restricted to move in a circular region R of radius 100 units. After some time the turtle could end-up at the boundary of R, hence it needed a re-entry model to bounce back into R. The re-entry model is based on the laws of angular reflection. The angle of incidence of the turtle onto the tangent, where the turtle made contact with the boundary, was calculated and then was reflected of from the tangent with the same angle. The point of contact with the boundary was calculated by solving the equation of the circle (for radius 100 units) simultaneously with the equation of the line drawn by the turtle (calculated by using the last 2 co-ordinates of the turtle) to get the 2 points of intersection. The farthest point of intersection was discarded. The turtle was allowed to take a 1000 steps and then it's final displacement (Euclidean distance) from the origin would be saved. After repeating the experiment 10,000 times all final displacement's were plotted using a normal distribution which be seen in the figure 4.

1.4 Task 4

Solution: For this task we repeat task 1 with the following change:

- set S is now continuous & contains the closed interval $[0 - 1]$

Hence step sizes are no longer discrete values but vary from 0 to 1.

1.5 Task 5/6

Solution: For this task we repeat task 3 with the following change:

- set S is now continuous & contains the closed interval $[0 - 1]$.
- set O is not being restricted to 4 intervals.

Hence step sizes are no longer discrete values but vary from 0 to 1 & orientation varies from $0 - 2\pi$

1.6 Task 7

Solution: For this task we repeat task 3 with the following change:

- set O is not being restricted to 4 intervals.

Hence orientation varies from $0 - 2\pi$

1.7 Task 8

Solution: Building on task 5, two nodes are distributed randomly in the circular region by random selection of angles between 0 and 2π. The distance from the center is also calculated at random. But as this distribution does not give uniformity, then a random number between 0 and 1 is chosen and is square rooted to make it a grow slower as the number reaches 1 and then is multiplied with the radius of the circle to give a uniform distribution of nodes around the circle. Details about the algorithm can be found *here*.

The two uniformly distributed nodes are then allowed to perform a random walk of similar to that of Task 5, but the experiment ends when the two nodes are within 1 unit radius of each other. The expected time taken for the two nodes to come this close depends on their starting point. If the distance between the two nodes is more than the radius of the circle, it takes almost infinite amount of steps for them to reach close to each other. However, when the distance is minimal it takes even steps in hundreds for them to meet. We set a limit of 1 million steps for a breakpoint where we restart the experiment. In the results obtained, we can clearly say that expectation rises exponentially with increase in radius.

$$E[X] = ae^{br}$$

1.8 Task 9

Solution: Building on task 8, we populate the region R with many nodes within some vicinity of each other. We randomly choose a node (make it red) to represent the patient. We allow every node to move randomly. When a patient comes in contact with a healthy node it makes it sick too (changes it's color to red). A sick node become healthy after some time (changes it's color to black).

We simulate 2 scenarios to visualize the effects of social distancing.

- In scenario 1 we populate the region R with many nodes within a small vicinity of each other (say 5 units).
- In scenario 2 we populate the region R with less nodes within a large vicinity of each other (say 50 units).

The simulations will help us understand how practicing social distancing amid a pandemic flattens the curve.

In scenario 1 due to a large number of nodes relatively close to each other we see that many nodes become sick quickly.

In scenario 2 due to a small number of nodes relatively far from each other we see that not many nodes become sick and the sick ones recover without infecting many. This is due to the fact that the expected distance of a node from it's origin after a large number of steps is close to 0. Hence, a node will not travel very far and cannot infect a far away node.

2 Graphs/Figures

2.1 Task 1

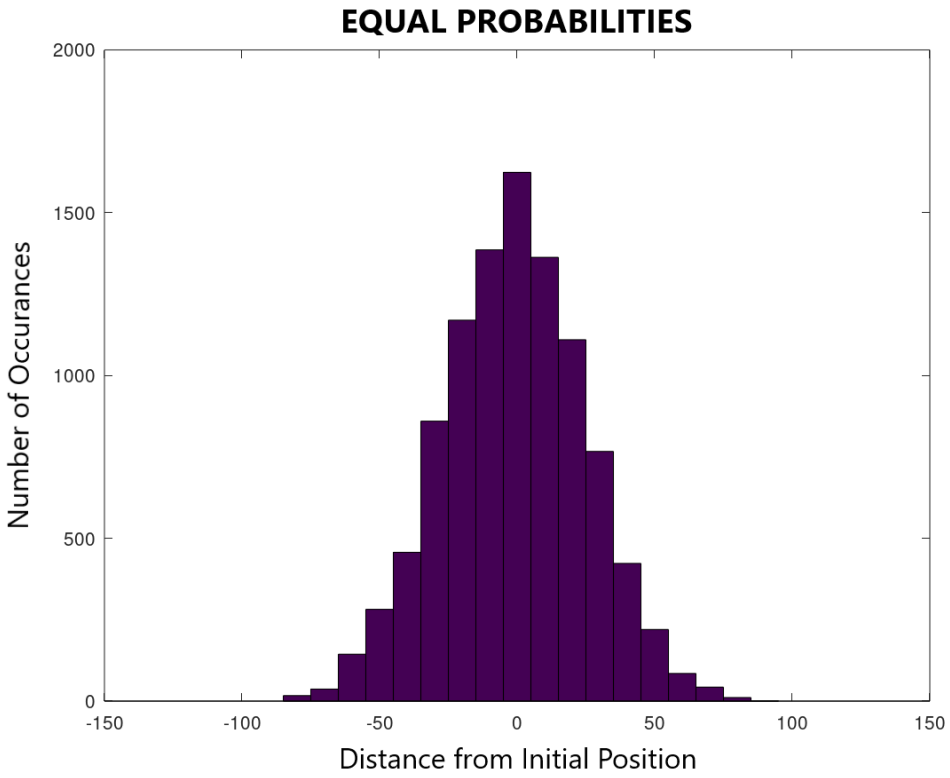

Figure 1: Task 1 For Equal

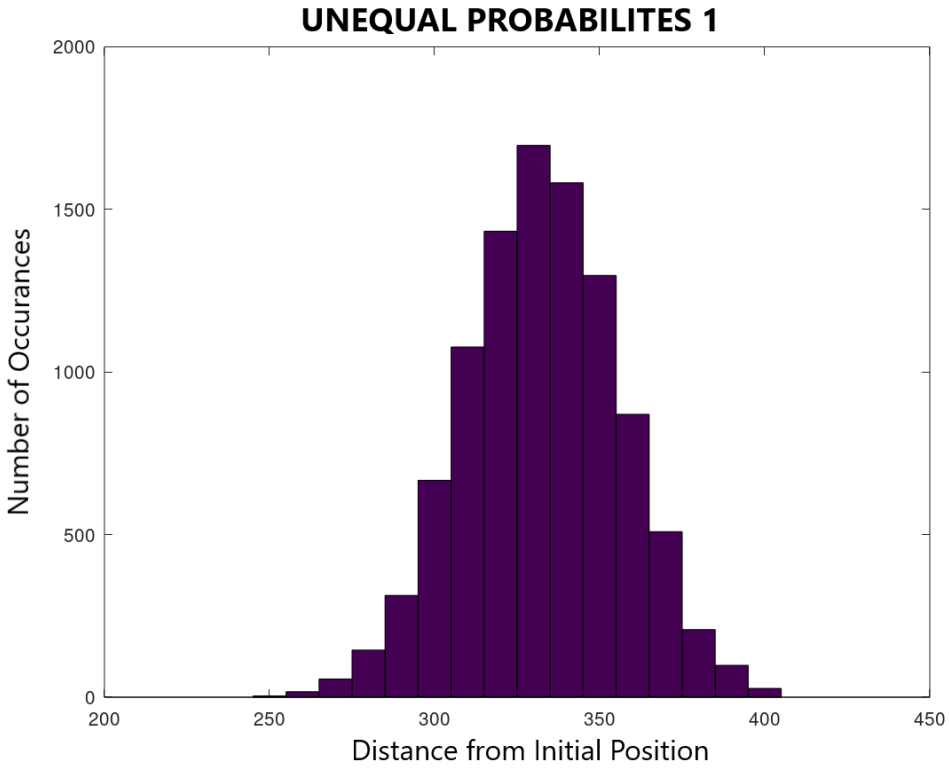

Figure 2: Task 1 for Unequal

2.2 Task 2

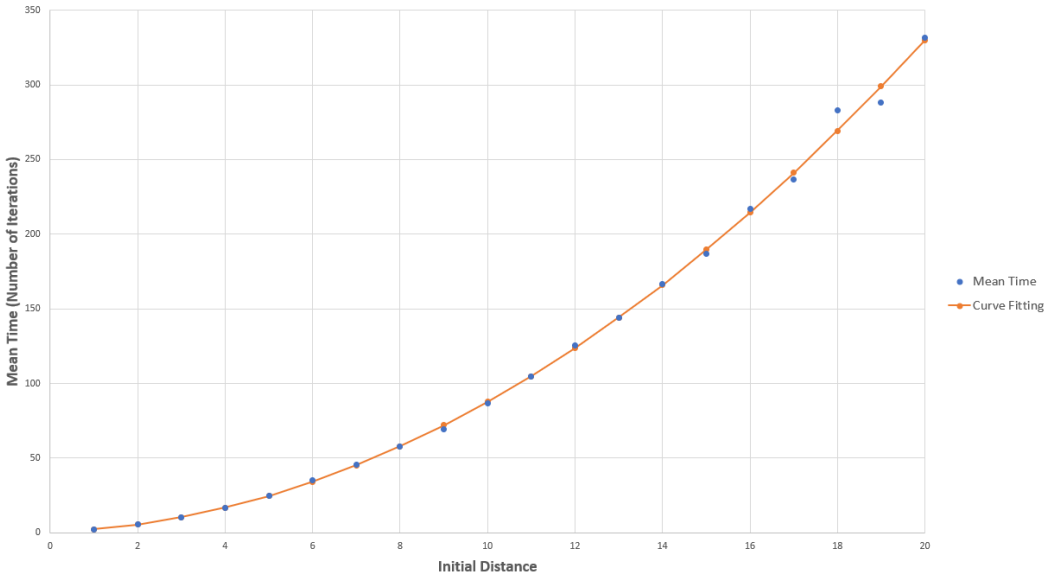

Figure 3: Task 2

2.3 Task 3

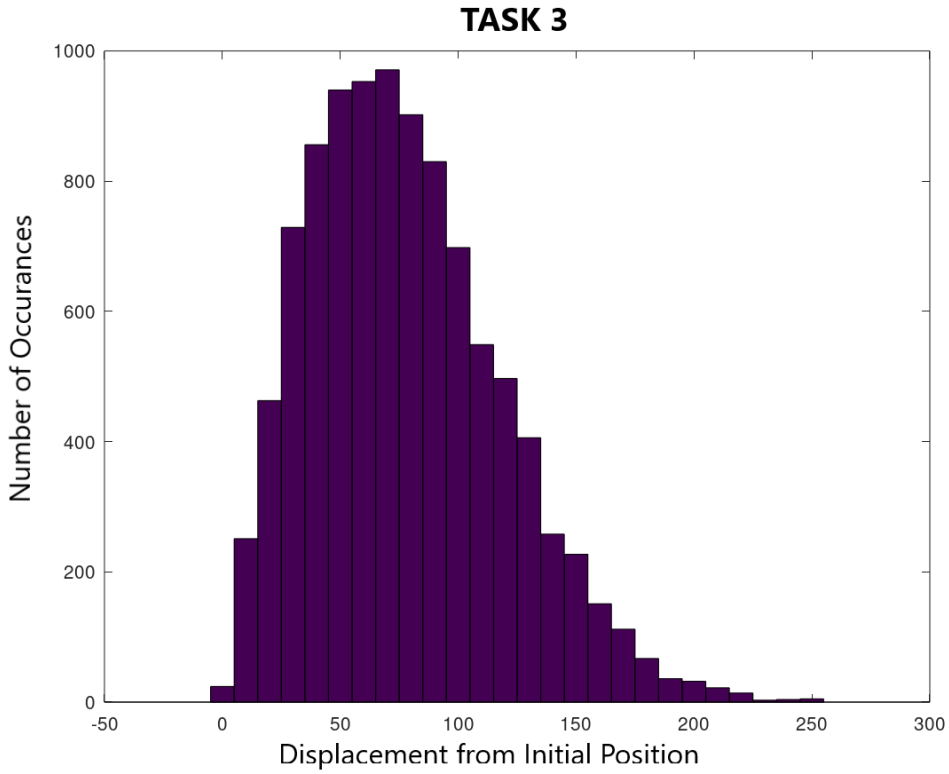

Figure 4: Task 3

2.4 Task 4

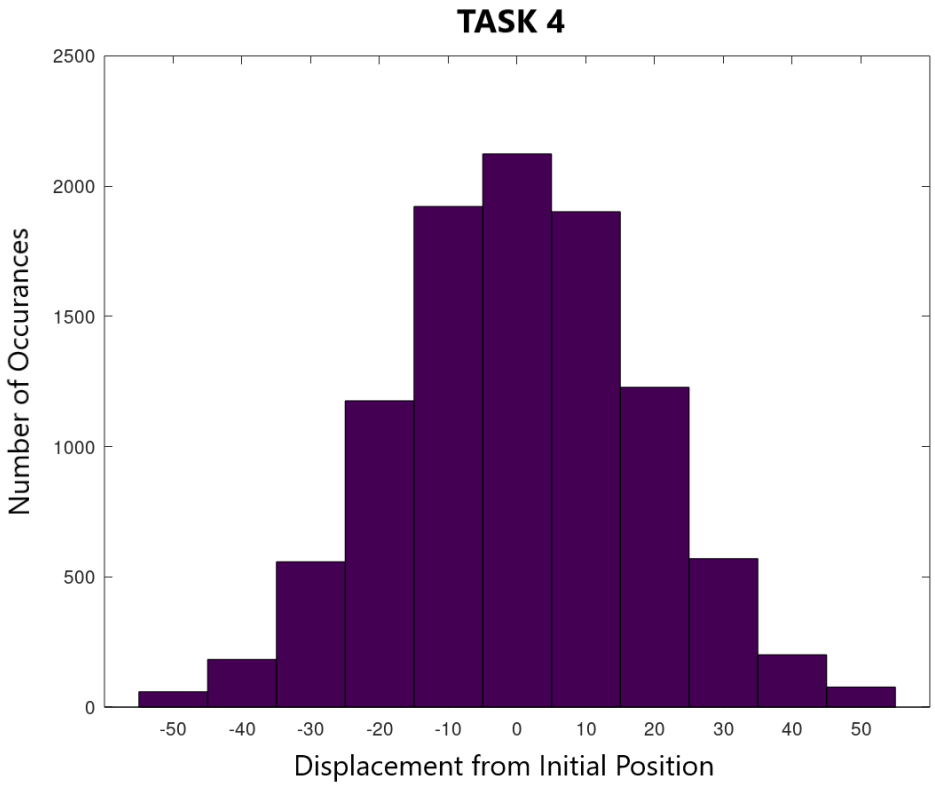

Figure 5: Task 4

2.5 Task 5/6

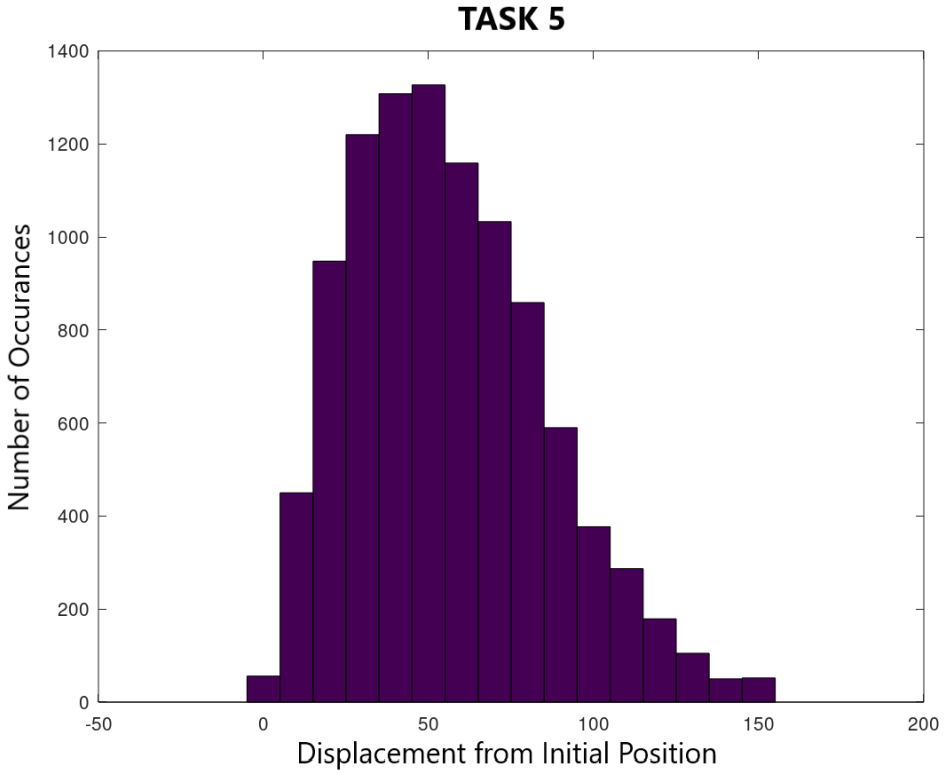

Figure 6: Task 5/6

2.6 Task 7

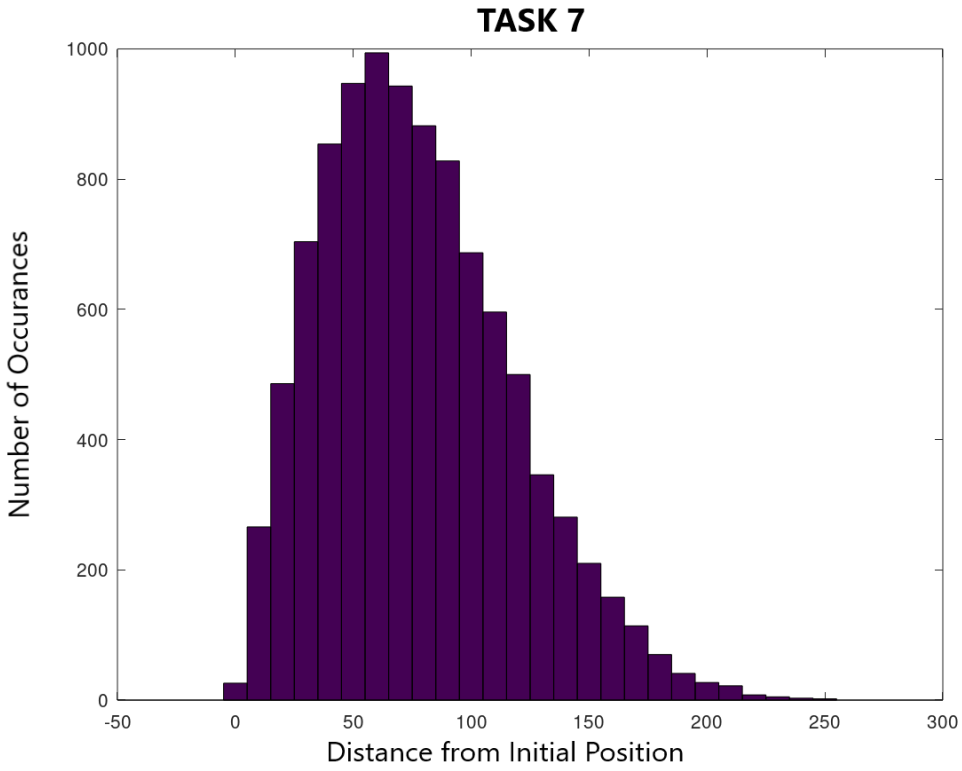

Figure 7: Task 7

2.7 Task 8

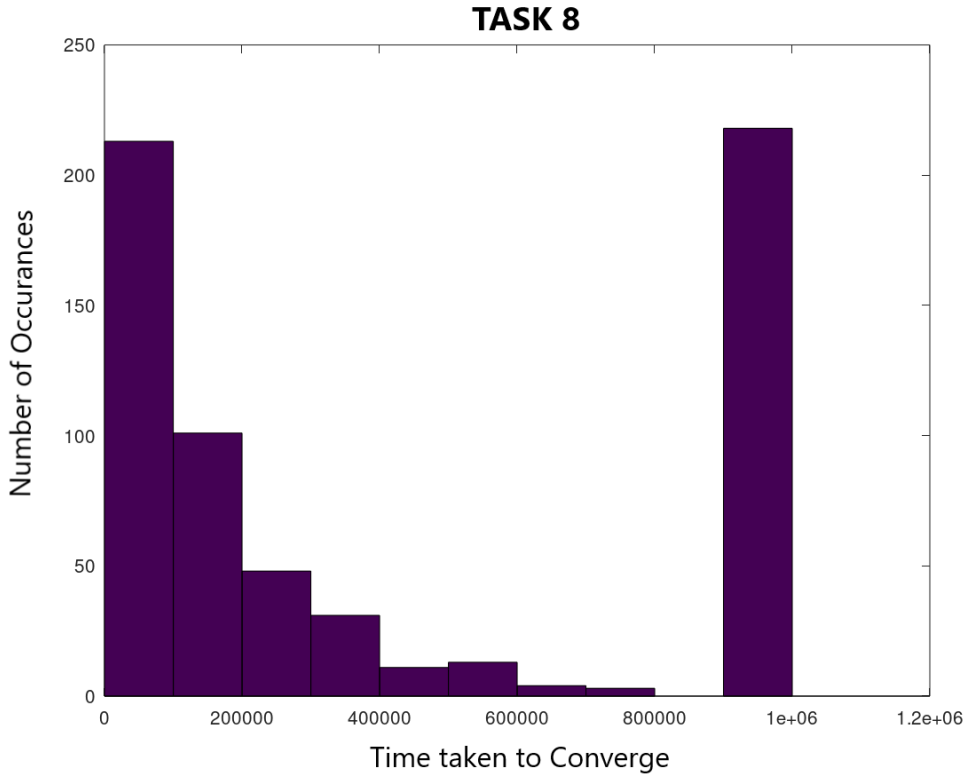

Figure 8: Task 8

2.8 Task 9

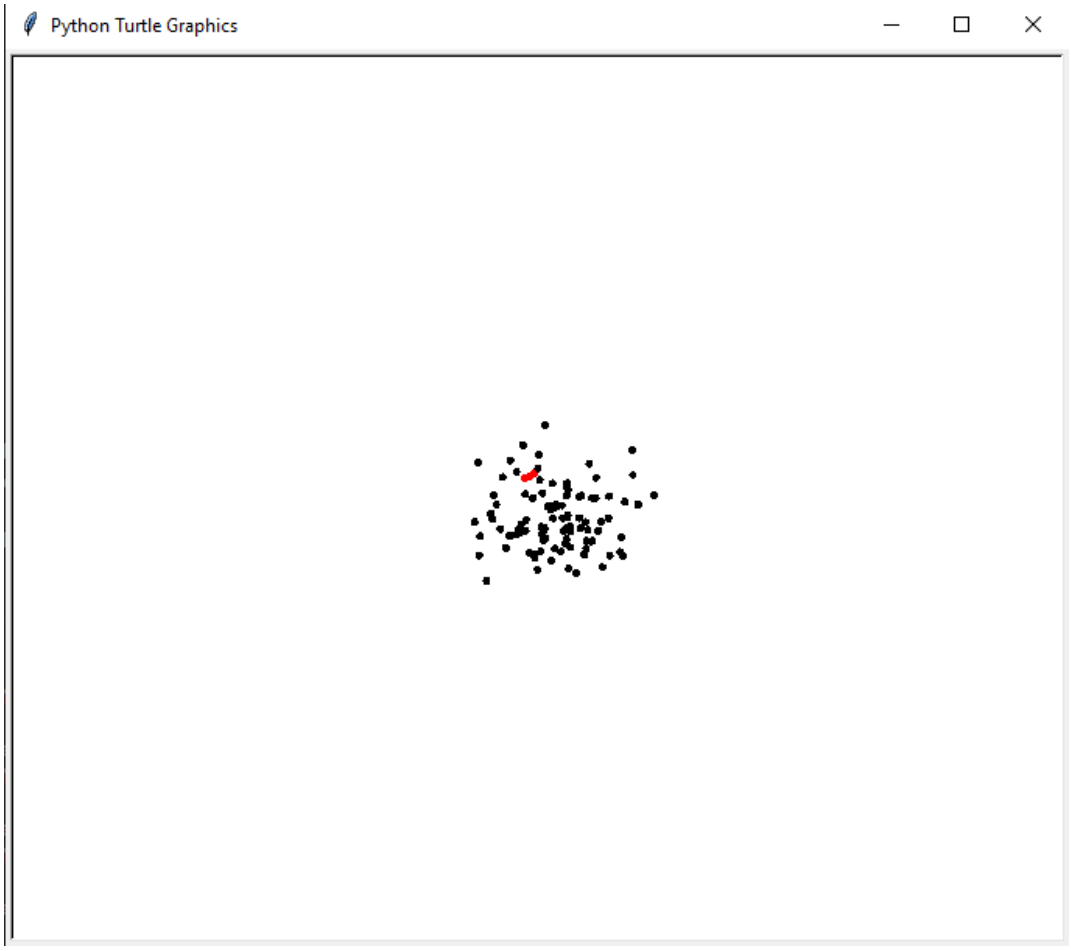

Figure 9: Task 9 (Close)

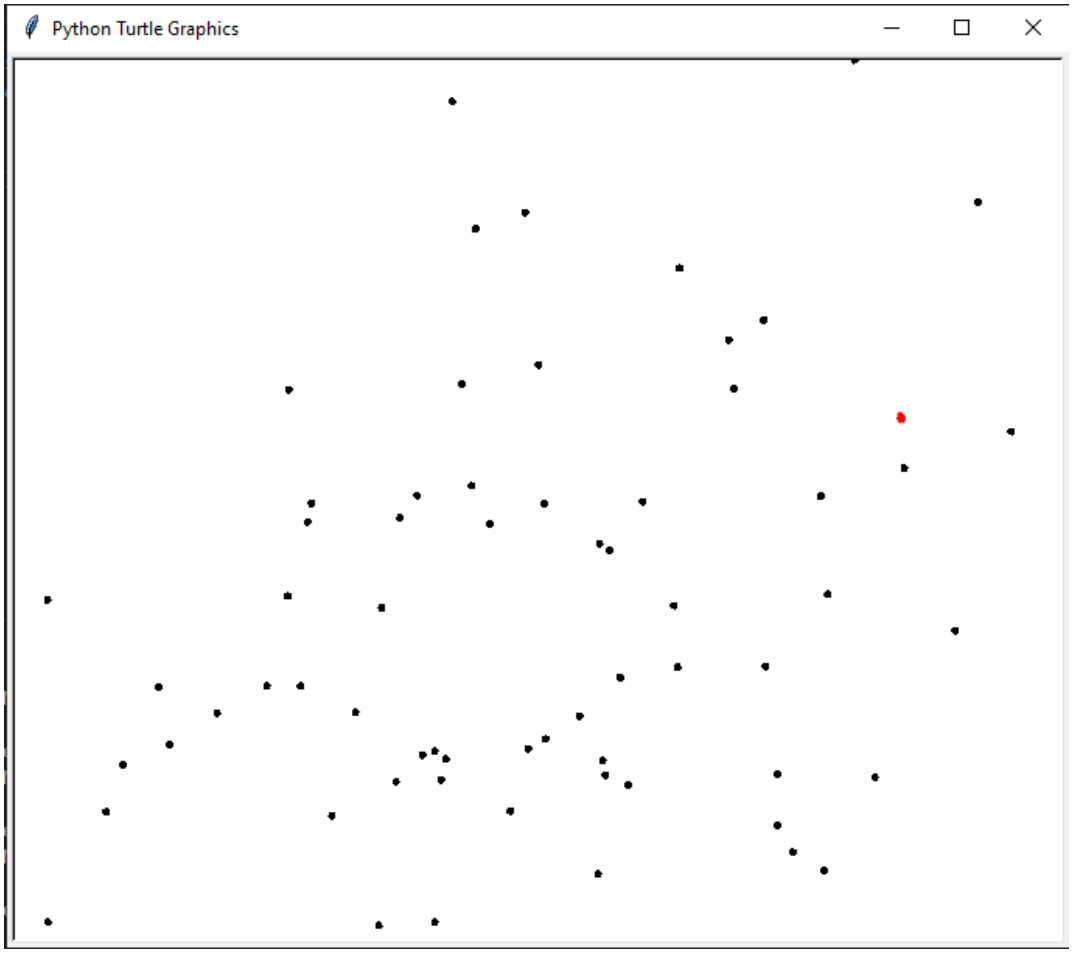

Figure 10: Task 9 (Scattered)

3 GitHub Repository

Please refer to our *GitHub repository* for simulation codes

4 References

- https://www.stat.berkeley.edu/ aldous/RWG/Book$_R$*alph*/*book.html*
- https://blogs.sas.com/content/iml/2016/03/30/generate-uniform-2d-ball.html