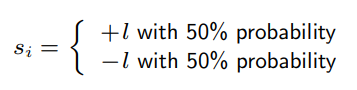
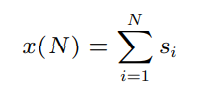
**-Discrete 1-D random walk:**

**-Unbiased Random Walk:**

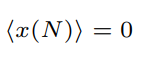
Let’s assume that a walker begins at x= 0, and the length of each step is equal and having equal probabilities of moving in the left or the right direction. Another important thing to note is that all the steps are independent of the previous step. Let’s represent the displacement of the walker at each step by si,



After N steps, it is easy to see, the position (and displacement) x of the walker will be;



Now, the average position for the current setup will be,

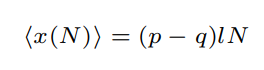


This is because for every +L we’ll have a –ve L due to equal probabilities. It means that the probability of finding the particle somewhere is centered at x = 0, but naturally the probability distribution gets wider with increasing numbers of steps N [1], as can be seen from the results of 1000 experiments and 1000 simulations for 5 steps and 10 steps respectively in figure 1.

These results were generated by Monte Carlo simulation. Random.choices () a library function of python generated a uniform random pick by the defined probabilities for the direction of the movement and then this experiment was repeated for large numbers for the representation of the average.

-**Biased Random walk:**

Now, let the probabilities be unequal for taking a step in the left and right direction. Let them be p and q where, q = 1-p, now the average position will be [1]:



This can be observed in the graphs in figure 2 where,

For figure 2a, L = 1, N = 50, and p = probability of moving in the right direction = 0.7 and q = probability of moving in the left direction = 0.3. (Theoretical value = (0.7 – 0.3)\*1\*50 = 20)

For figure 2, L =1, N = 50, and p = probability of moving in the right direction = 0.9 and q = probability of moving in the left direction = 0.1. (Theoretical value = (0.9 – 0.1)\*1\*50 = 40)

**Meeting of two random walks in 1-D:**

The difference of the two random walks can be modeled as a random walk [2]. So, if we have walker A stand at the origin and walker B stand at a distance d from the origin then the random walk obtained as a result of the difference of the two random walks will have a origin at the difference.

The figures 3a and 3b are of equal probabilities with difference equals 5 & 50 respectively.

**Biased probabilities:**

For figure 4a, L = 1, difference = 50, and p = probability of moving in the right direction for person 1 = 0.75 and q = probability of moving in the left direction = 0.25. The second person moves with equal probabilities.

For figure 4b, L = 1, difference = 50, and p = probability of moving in the right direction for person 1 = 0.75 and q = probability of moving in the left direction = 0.25. Probability for moving in the right for the second person is 0.6 and for moving left it is 0.4.

**Circular discrete walk:**

For the walker to walk in a circular path with varying step size and direction, random. Choices (), a python library function was used to generate the random step between {0, 0.5, 1} with equal probabilities, similarly different angles were picked at discrete intervals to determine the direction of the walker. These were then converted from the polar coordinates to the Cartesian coordinates by the relation:

X = r\*cos (theta); where r is the step size,

Y = r\*sin (theta); where r is the step size

And this process continued till the desired number of steps were not completed. This was the initial step. For maintaining the walkers within the 100 unit radius circle, a tangential rebound approach was used [3].

**Assumptions for simulations:**   
Starting Position will be the origin i.e. (x, y) = (0, 0), Radius = 100 units, Step Size is can be either {0, 0.5, 1} with equal probability, and Angle can be either {0, 90, 180, 270} with equal probability.

The simulations for different number of steps can be observed in figure 5.

**Continuous 1-D random walk:**

The theory is similar to the discrete random variables, applied to continuous variables.

It can be observed in the simulation results provided in Figure 06. Assumptions for the simulations are:   
 In figure 6a, n = 50, Step Size is Uniform Continuous Random Variable over (0, 1), Experiments = 1000, simulations = 1000, with equal probabilities of moving in either direction.

In figure 6b, n = 50, Step Size is Uniform Continuous Random Variable over (0, 1), Experiments = 1000, simulations = 1000, with probability of moving right = 0.9 and probability of moving left = 0.1.

It can be observed that the expected distance has reduced to half in comparison to the discrete random variables due to the uniform random distribution for each step.

**Continuous 2-D circular walk:**

Similar approach was taken as for the discrete continuous circular walk but this time for the continuous variables.

**Assumptions for simulations:**

Starting Position will be the origin i.e. (x, y) = (0, 0), Radius = 100 units, Step Size will be Uniform Continuous Random Variable over (0…1) inclusive and Angle will be Uniform Continuous Random Variable (0 … 2π) inclusive.

The simulations for different number of steps can be observed in figure 7.

**Mixed 2-D circular walk:**

Starting Position is the origin i.e. (x, y) = (0,0), Radius = 100 units, Step Size will be a discrete random variable from {0, 0.5, 1} with equal probability, and Angle will be the Uniform Continuous Random Variable between (0 … 2π) inclusive.

The simulations for different number of steps can be observed in figure 8.

**2 Random walks within a circle:**

Firstly two positions were uniformly and randomly picked within a 100 unit circle. They were picked by the equation:

r = 100 \* sqrt (random. Random ())

Where, random. Random () yields a uniform number between 0…1. Then by using the usual equation of the polar to Cartesian coordinates i.e.

x = r\*cos (theta)

y = r\*sin (theta)

The position was picked. An interesting question here is, that why multiply by the root of the random number, where it was possible to just get a random position by multiplying with the random number alone. So, by multiplying by the random number directly we get random points concentrated in the centre rather being uniformly distributed [4]. So for having a uniform distribution the average distance between points should be the same regardless of how far from the center we look. For example, on the perimeter of a circle with circumference 2 we should find twice as many points as the number of points on the perimeter of a circle with circumference 1. And since we know that the circumference grows linearly with R so naturally the random point should linearly grow with R too i.e. the desired PDF grows linearly [5]. For simplicity we’ll see for radius = 1.

PDF (x) = 2x

Now, we calculate the CDF:

*CDF*(*x*) = ∫*PDF* = ∫2*x* =*x*2

Upon mirroring the CDF across y = x, we get;

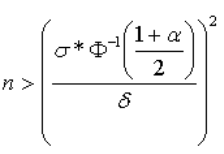
*CDF*: *y* = *x*2  
Swap: *x* = *y*2  
Solve: *y* = √*x*  
*CDF*-1: *y* = √*x*

*Hence,*

*CDF*-1(random ()) = √random ()

Further description can be found here[6].

Next Position was calculated in the same way as in Task 5. The number of simulation which give a 80% accuracy that the mean lies within the true mean value + 5000 or -5000 was calculated by the following equation [7]:



Where Ф-1 (•) is the inverse of the standard Normal cumulative distribution function (i.e. with mean 0 and standard deviation 1). α = 0.80, δ = 5000 and σ = 88147.37 (obtained through 50 simulations) shown in figure 9b.

The number of simulations should be: N> 517.199

For determining an 80% accurate mean.

Since radius = 100 required a lot of computation and time, below in figure 9a, the expected number of steps are shown for radius = 10. For radius = 10; α = 0.80, δ = 100 and σ = 615 (approx.). Therefore

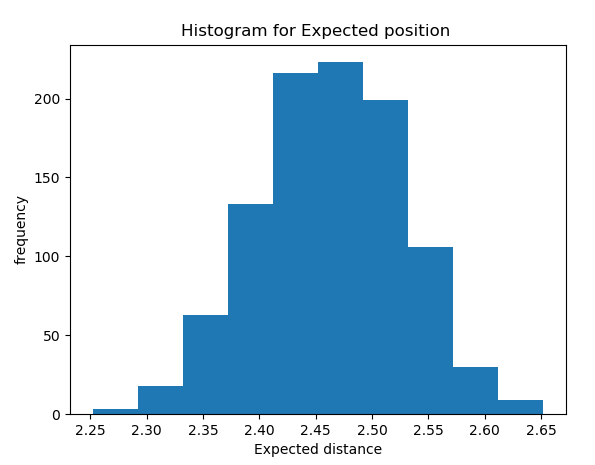
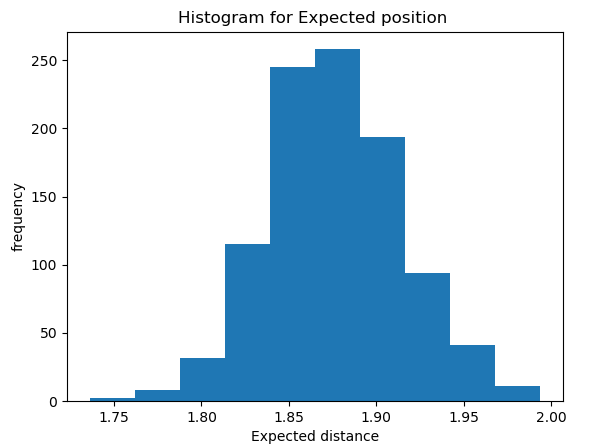
N > 62.94

**Link to the GitHub repository:**

<https://github.com/AnushaRehman/ar04025-randomwalk>

**Figures:**

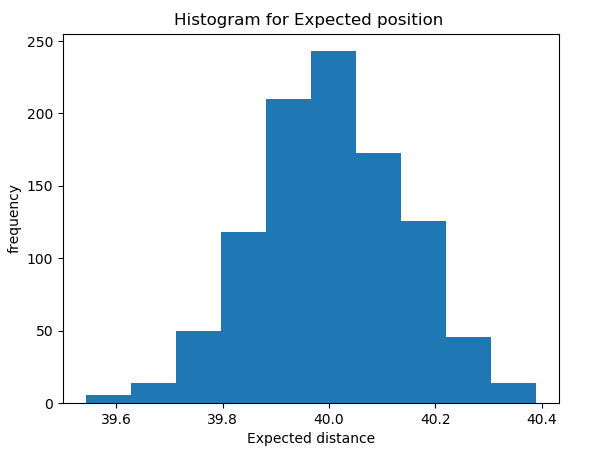
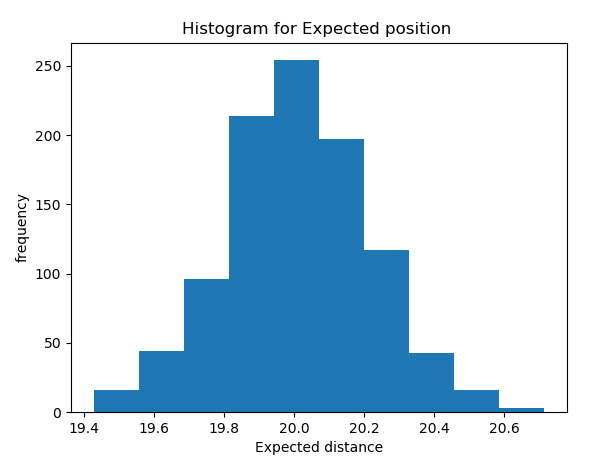
**Un-biased random walk:**



1. *(b)*

Figure 1

**Biased random walk:**

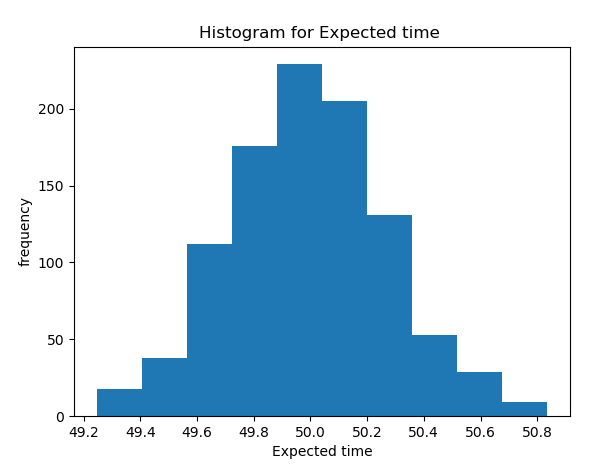
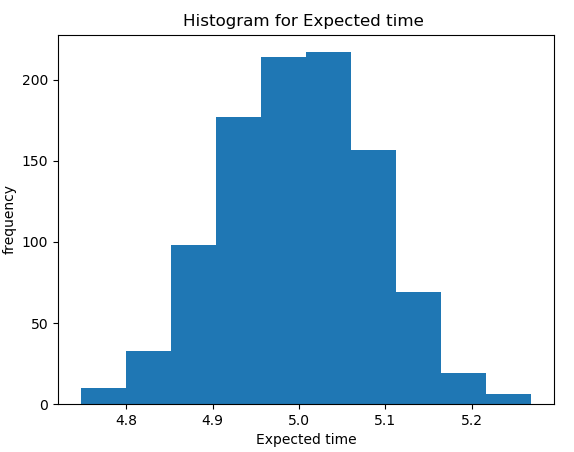


1. (b)

Figure 2

**Two- walks:**

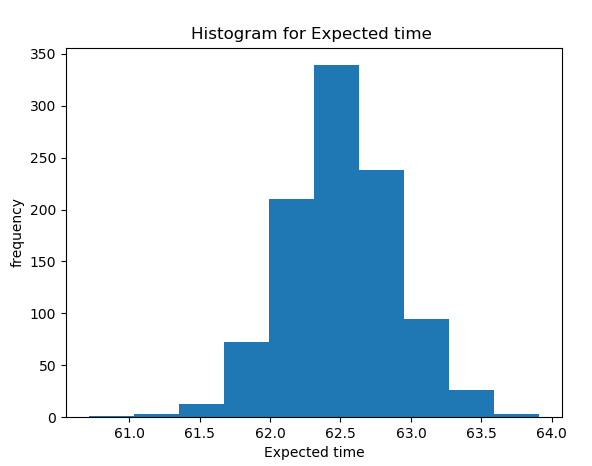
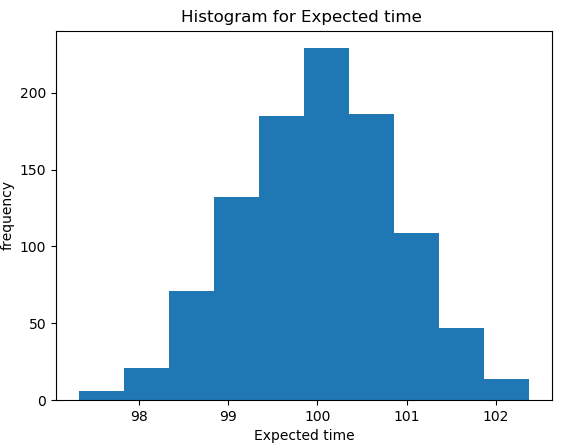
**Un biased :**



1. (b)

Figure 3

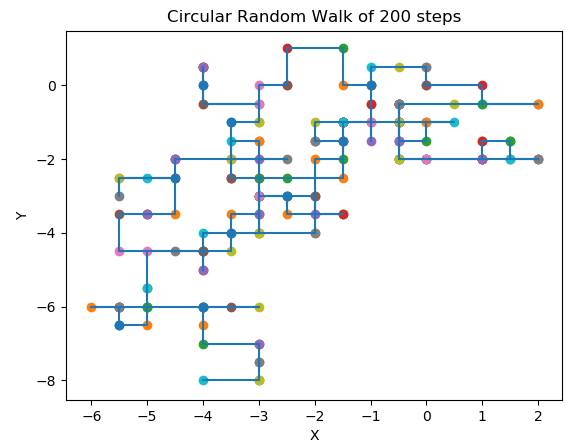
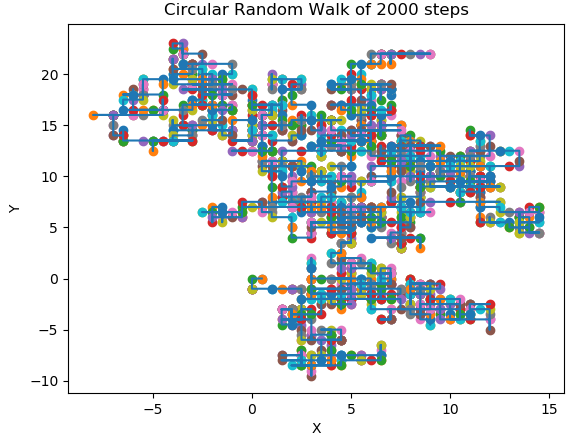
**Biased :**



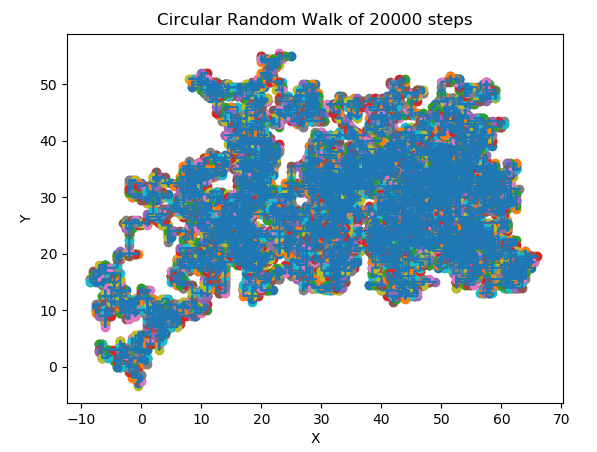
1. (b)

Figure 4

**-Discrete 2-D circular walk :**

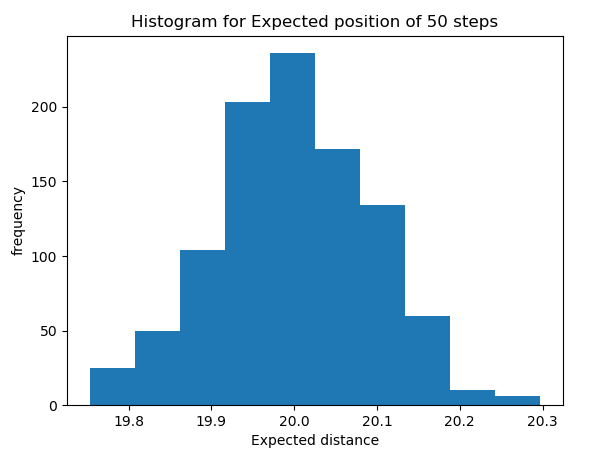
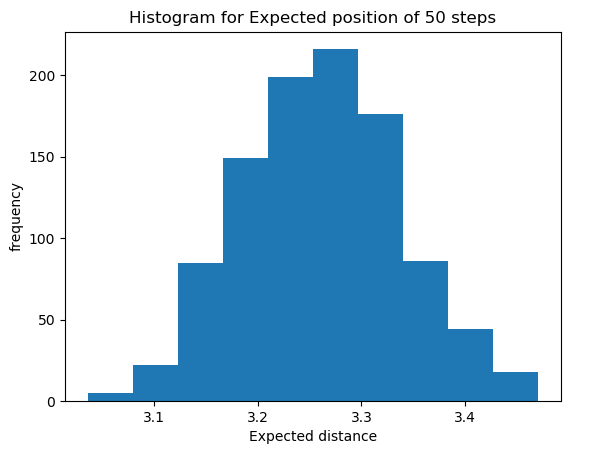
1. *(b)*



(c )

Figure 5

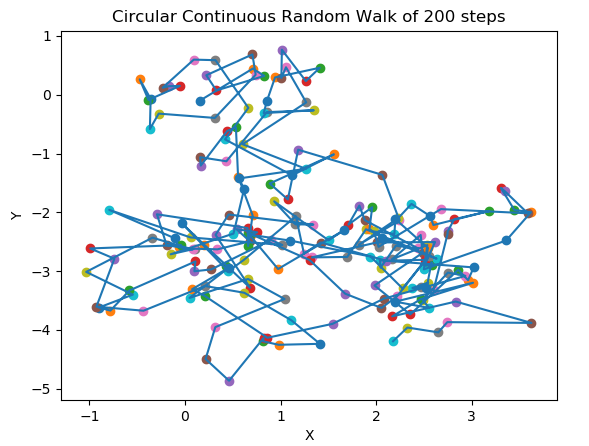
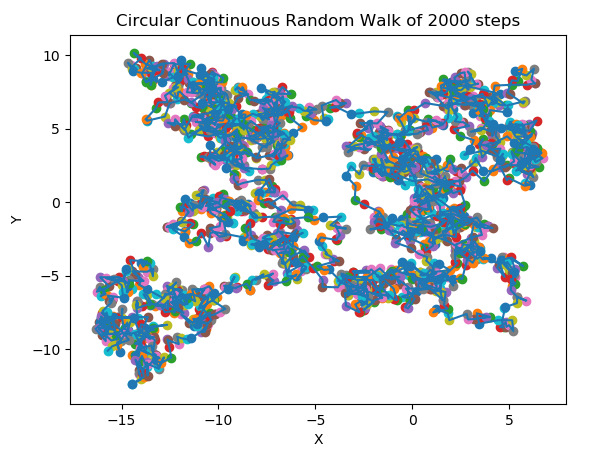
**Continuous 1-D random walk:**



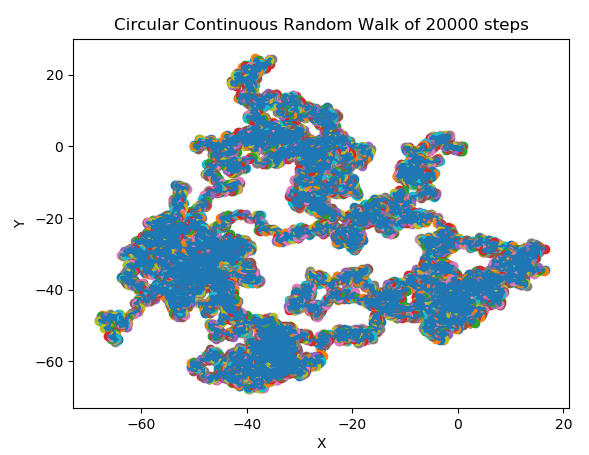
1. (b)

Figure 6

-**Continuous 2-D circular walk:**

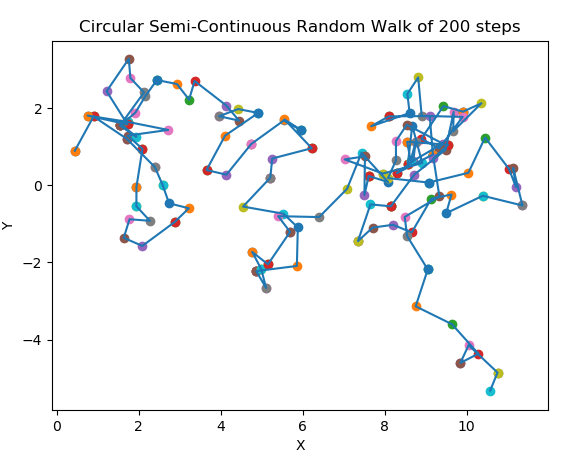
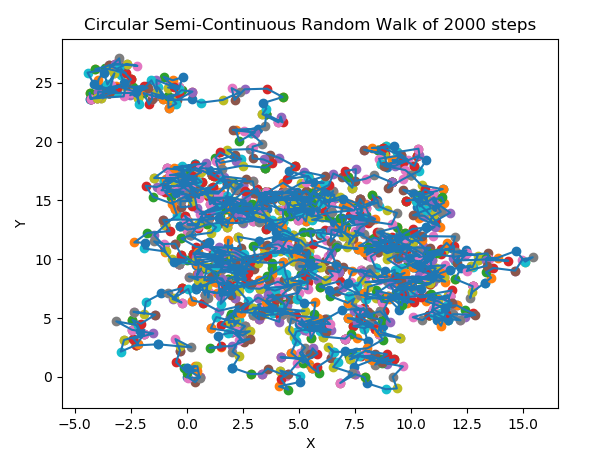
1. (b)



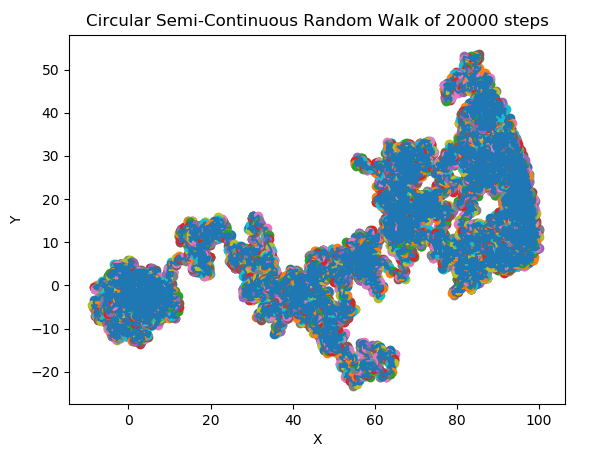
(c )

Figure 7

**-Semi-Continuous 2-D circular walk:**

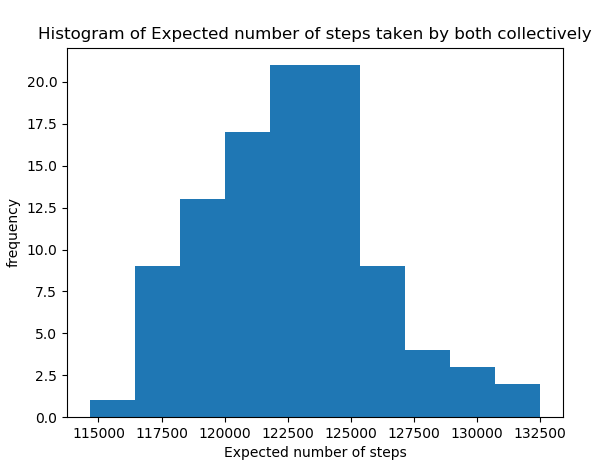
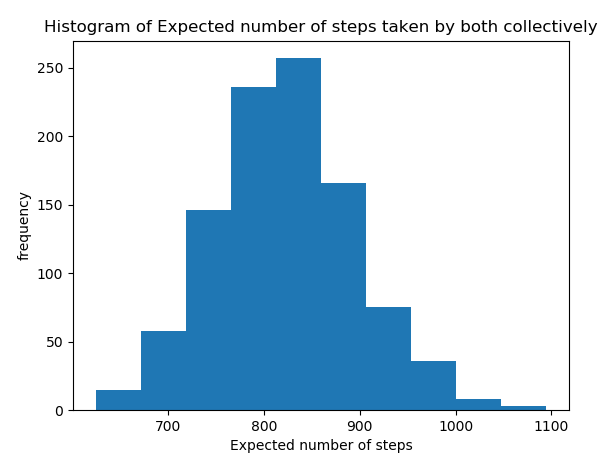
1. (b)



(c )

Figure 8

**-2 random walks within a circle:**

****

1. (b)

Figure 9

-References:

[1] http://www.acclab.helsinki.fi/~knordlun/mc/mc7nc.pdf

[2]

[3] <https://stackoverflow.com/questions/36013940/making-particles-move-randomly-within-a-circle>

[4] <https://blogs.sas.com/content/iml/2016/03/30/generate-uniform-2d-ball.html>

[5] <https://programming.guide/random-point-within-circle.html>

[6] <https://programming.guide/generate-random-value-with-distribution.html>

[7] <https://www.vosesoftware.com/riskwiki/Howmanyiterationstorun.php>