**Purpose of the experiment:**

The purpose of this lab is to consider the free diffusion of a particles in both 1D and 2D and graph the motion of the random walk’s or a motion of a particle.

**Problem Description:**

In lab 7, the particles we will be start at the origin of an x-y plane (coordinates (0,0)). We then assume the motion is random walk. Each step will update the position of the x and y component of the particle randomly and independently.

Before I did anything, I imported the libraries necessary for plotting, mathematical functions and the random function as shown in figure figure (1).

The first part of the script is where I declared the following constants num\_steps = 500, num\_walkers = 1000. num\_walkers represented the number of times to repeat the situation and the number of walkers. num\_steps represented the length of the random walks.

import numpy as np

from numpy.random import random as rng

import matplotlib.pyplot as plt

num\_steps = 500

num\_walkers = 1000

endpoints = np.zeros(num\_walkers)

temp = np.zeros(num\_steps)

*(Figure (1) – Importing the important libraries useful for this script and declaring constants)*

I then created a random walk for each walker which would then create two arrays x\_step and y\_step that has 500 elements with each element between 0 and 1 within for loop. To create a binary distribution, I had to identify when each element is greater or less than 0.5. If the element is greater than .5, it will move in the positive direction. If it is smaller than 0.5, it will be in the negative direction.

x\_step and y\_step represent the steps taken by the random walker. I add the previous values of each step together to find the position of the random walker at each time step. I then plotted the position at each x and y value of all the walkers.  
  
After graphing all the steps of the walkers, I then calculated and graphed the distance of each walker from the origin at each time. I defined an array called plot that represents the cumulative sum of the . At each time, the displacement in my script is written as r2 = x\*\*2 + y\*\*2. I then added each of the previous values to r2 inside the loop. After this, all code was done outside the loop. The loop looks like figure 2 below.

for j in range(num\_walkers):

#initialization for number of elements in each step

x\_step = rng(num\_steps); y\_step = rng(num\_steps)

#Initialize the steps to be either zero or one

x\_step = 2 \* (x\_step > 0.5) - 1

y\_step = 2 \* (y\_step > 0.5) - 1

#Sum of each step

x = np.cumsum(x\_step)

y = np.cumsum(y\_step)

#Plotting of all steps in all direction

plot = x; title = "Ramdom walks in 1D"

plot2d(); plt.plot(plot, y)

#Calculations to plot the average

r2 = x\*\*2 + y\*\*2

temp = temp + r2

plot = temp/num\_walkers; title = "Average Walk in 1D"

plot2d(); plt.plot(plot)

*(Figure (2) – the for – loop that calculates and plots the walks and calculates the averages)*

All values of temp = temp + r2 were then normalized with respect to the number to the

number of walkers. This was then plotted to show the average displacement of all walkers.

For graphing in 2D, I repeated the steps above except, in the for loop. What I did differently is using a random variable theta\_step that was a range of six numbers different numbers in each iteration that were integers between 0 and 6 multiplied by 60. I then converted all values inside the array to radians. x\_step was the cosine of the theta\_step values and y\_step was sine of the theta\_step values. It is shown in figure (3)

temp1 = np.zeros(num\_steps)

for i in range(num\_walkers):

theta\_step = np.radians(np.random.randint(0,6,num\_steps) \* 60)

x1\_step = np.zeros(num\_steps); y1\_step = np.zeros(num\_steps)

#Initialize the steps to be either zero or one

x1\_step = np.cos(theta\_step)

y1\_step = np.sin(theta\_step)

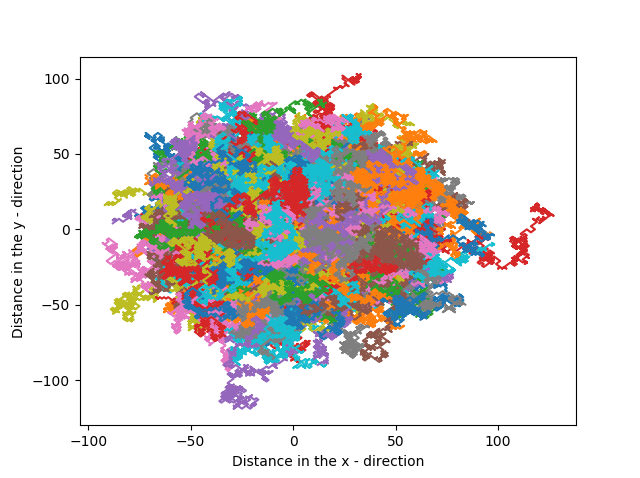
*(Figure (3) – The difference when calculating 2D values)*

**Equations solved and algorithms used:**

In this lab, to determine the placement of the walkers, or particles, I made a for loop that would repeat for each walker that would make a random path. Within the for loop for the 1D and 2D part of the labs, I took the cumulative sum of each step and plotted them. I then took the average step of each walker by adding the square of the cumulative sum of each x and y steps and add it with a temp variable. After, stored the resulting sum into the same temp variable and divided it by the number of walkers outside the loops. The plotting of the averages was done outside of the loops

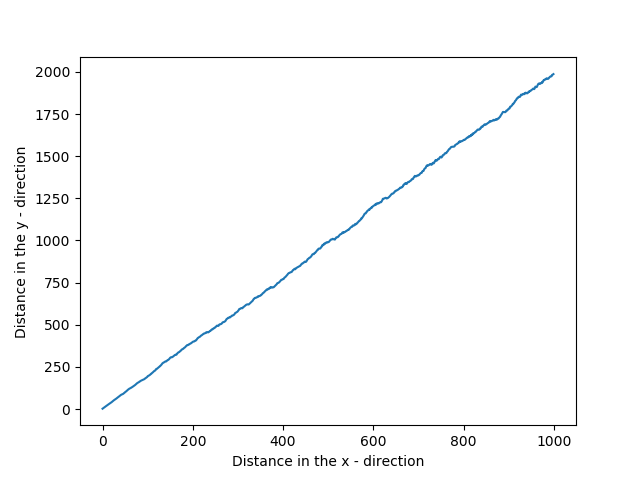
**Results & Analysis**

In figure 4, shows the path each walker has taken in the x and y positions in 1D. The walks were mostly resembled small quadrilateral shapes.



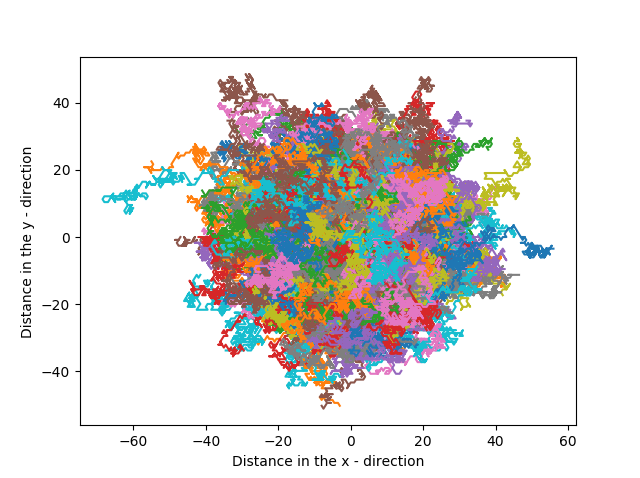
*(Figure (4) – 1D direction plotted on the xy plane)*

The average walk in 1D, show below in figure 5, has a slope of 2 which means each walker or particle moved two units the y direction for one unit moved in the x direction. The slope is also the diffusion constant, which is 2.



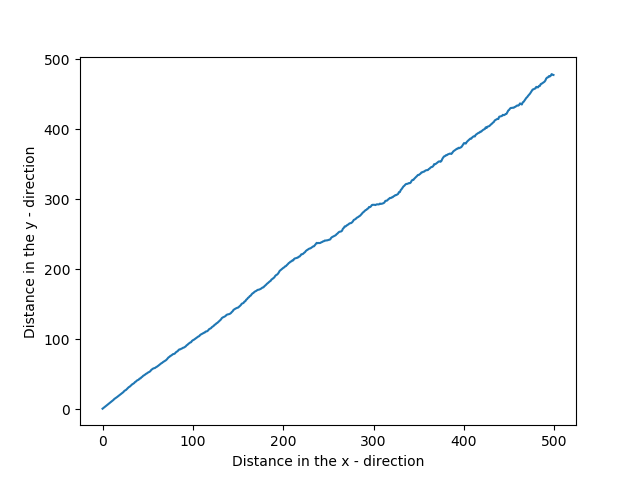
*(Figure (5) – Average walk of a 1D walker)*

For the 2D plots as show below on figure 6, the paths taken by the walkers were triangles. The paths were also plotted on the x y plane.



*(Figure (6) – motion of particles or walkers in 2D)*

The slope of the average walker happened to be one as show from the graph in figure 7



*(Figure (7) – The average motion of particles or walkers in 2D)*

The average walk in 1D, show below in figure 7, has a slope of 1 which means each walker or particle moved one units the y direction for one unit moved in the x direction. The slope is also the diffusion constant, which is 1/2.