Lab 7
March 31, 2017

Random Walk
(Based on chapter 7 of "Computational Physics" by Giordano and Nakanishi)

Here we will consider the free diffusion of a particle in 2D and model its motion as a random walk.

In these examples, assume that the particle starts at the origin in the x-y plane with the coordinates (0,0). Assuming that the motion is a random walk, at each step, we will update the position of the particle, i.e. its x and y coordinates independently. Each step has a unit length in x and y direction and its direction is chosen randomly.

We will start with the python script for the 1D example (random_walk_1d.py) and modify it to simulate the motion in 2D. You can download the script along with these instructions from the blackboard. A brief summary of the script and necessary modification are provided below:

- Define the following constants:
  ```
  num_steps = 500
  num_walkers = 1000
  ```
  to store the length of the random walk (num_steps) and the number of times you want to repeat the simulation (num_walkers).

- To create a random walk for each walker, we will define a loop and execute the following steps for each iteration:
  - Create two arrays called x_step and y_step to store the random steps taken by the walker in each direction. We will do this in two steps.
    ```
    x_step = rng(num_steps)
    ```
    will define an array that has 500 (num_steps) elements and each element is chosen randomly between 0 and 1.

    Add a similar command to define an array called y_steps for the steps taken in the y-direction.

    The rng function will generate random numbers that are uniformly distributed between 0 and 1. To create a binary distribution of random steps with equal length, for each element that is > 0.5, we will assume a move in the positive direction and for every element that is < 0.5, we will assume a move in the negative direction. This is achieved by the following commands:

```
x_step = 2 * (x_step > 0.5) -1
```

where the value of the argument in parentheses is 1 if True and is -1 if False. Define a similar command for y_step.

The two arrays, x_step and y_step represent the steps taken by the random walker. To find the position of the random walker at each time step, for example at time n, we need to add the steps taken prior to that. The following command will create an array with the cumulated sum of all the elements in x_step:

```
x = np.cumsum(x_step)
```

define a similar command to define y.

Now, (x,y) will represent the coordinates of the random walker at each time, and you can plot it with the following command:

```
plt.plot(x,y)
```

- In addition to the position of the walker, we are also interested to plot the distance of the walker from the origin at each time. For that, we will define another variable (array) called r2ave that represents the squared displacement of the walkers from the origin. At each time, the displacement can be written as:

  ```
  temp = x**2 + y**2
  ```

  which we will store in a temporary variable called temp. temp will represent this distance for each walker. Modify the above line in the script to represent the squared distance of a walker in 2D (instead of 1D).

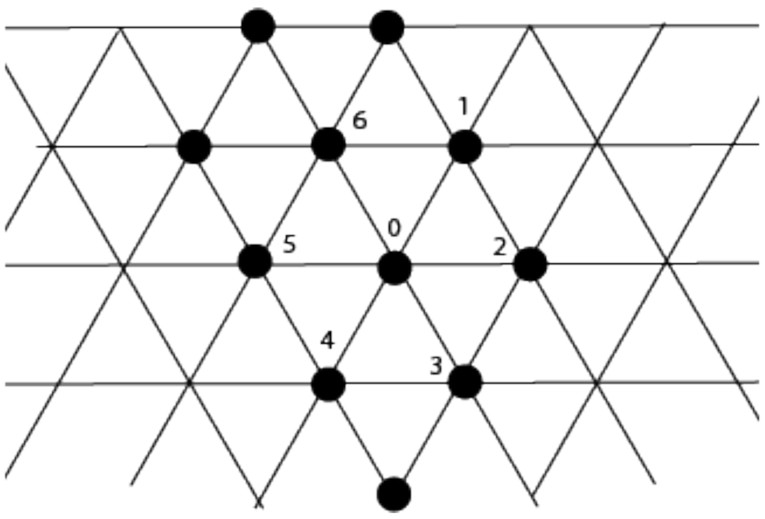
  To get the average displacement among all walkers (r2ave), we will add these values for each walker to r2ave, which at the end will be normalize by the number of walkers.

- The script will create two plots, one in which the random walks (x,y) are plotted for each walker and one in which r^2 , the squared distance from the origin is plotted.

To make sure that your script has created the random walks correctly, you can run the script for fewer walkers and shorter random walks and check your plots. However, the value of r^2 will not show the desired trend unless it is averaged over a large number of walkers.

- Run this script and calculate the diffusion constant in 2D.

In the next exercise, you will modify your script to simulate the random walk on a 2D triangular lattice. Such a lattice will look like the figure below:



Starting from point 0, the next move can end up in any of the 6 locations marked from 1 to 6.

To do this, you can define a variable called theta_step, which represents the direction of movement at each step and can have any of the following values with equal probability [0, 60, 120, 180, 240, 300].

You can do this in two steps similar to the first example:

theta_step = rng(num_steps)

theta_step = int(theta_step/ 6) * 60

Use these values to determine the values of x_step and y_step using geometrical arguments.

Then update the coordinates x and y by adding up these steps.

- You can then (1) plot the random walk and (2) the average distance from the origin similar to the first example.

- From the <r.r> plot, calculate the diffusion constant for a walker on this lattice.