Computational Assignment 1: Random Walks and the Mean Free Path

Philip Cherian

August 31, 2023

In this computational exercise, we will deal with a topic central to statistical mechanics: the idea of the *random walk*. Let us consider an N-step random walk on a two-dimensional grid. In order to execute such a walk, a walker starts off at the origin and, at every time step, moves a fixed amount in either the x or y directions. Thus, at every time-step, she randomly chooses between one of four possibilities. This is then repeated N times. In this exercise, you will simulate such a random walk and see how the average "distance" the walker covers varies with time.

(a) In the Jupyter Notebook provided to you, complete the `walk` function so that it simulates a random walk with a number of steps given by `n_steps`. [4]

The basic algorithm that you use should be something like this:

(i) Start each walker at the coordinates $(0,0)$.

(ii) Loop over the total number of steps (in your case, this is `n_steps`).

(iii) At each step, draw two random numbers: the first number will decide if the walker is going to move along the "x" direction or "y" direction. The next random number will decide if the walker will move "forward" or "backward" in the selected direction.

(iv) Add the walker's new position to the array meant to keep track of their trajectory.

(b) Once you have completed this, you can run the provided `simulate` function which will use your `walk` function to simulate 50 random walks of 10,000 steps and plot the results.

Then use the `plot_evolution` function to show how the random walk grows with time. You should see a "bundle" of walks slowly evolve. Comment on any conclusions you arrive at by looking at the time evolution. [1]

(c) One way to estimate the "size" of the bundle is to compute the mean-squared displacement of the walker after some number of steps n. Mathematically, we can define $R_d(n)$ through

$$R_d^2(n) = |\mathbf{r}_n - \mathbf{r}_0|^2, \tag{1}$$

where $\mathbf{r}_n = (x_n, y_n)$ is the position of the walker at step n.

Complete the `get_Rsq` function which takes in a single walk and computes and returns an array containing the values of R_d^2 at every step.

Use the data that you obtained from the `simulate` function to find R_d^2 for each of the 50 walks. Plot a graph of the *average* value of $R_d^2(n)$ as a function of n. (Remember, this average is taken over multiple *walks*.) Try to figure out the functional dependence of R_d^2 on n. What does this tell you about $R_d(n)$? [3]

(d) Random walks can be used to model a range of phenomena. For example, if you put a drop of ink into water, one can model its diffusion by imagining the molecules of ink executing a random walk in the water.

In the previous part you should have shown that $R_d^2(n)$ increases with n indefinitely. Of course, in real systems we have *boundaries*. Incorporate the following simple boundary conditions (known as *periodic* boundary conditions) into your `walk` function:

(i) If x_i goes beyond $x = L$, change $x_i \longrightarrow x_i - 2L$.

(ii) If x_i goes below $x = -L$, change $x_i \longrightarrow x_i + 2L$.

(iii) Apply the same conditions along y.

Convince yourselves that the above rules will constrain each walk to a box of side $2L$. Now repeat parts (a)–(c) with these new boundary conditions. Comment on anything interesting you see. [2]