Project 10: Random walk and the diffusion process: Numerical analysis and subsequent generation of the PDF of particle's position.

Gaurav Joshi[1] Adit Kaushik[2] Rutwik More[3] Arun Reddy[4] Aditya Deshmukh[5]

1 Project Abstract

In this project have developed a computer code to simulate the random walk process in 2 dimensions starting from $t = 0$ till $t = \tau$, where τ will be a user defined number and generated the PDF $P(X, \tau)$ of finding a particle at position X and time τ. We have also created various plots to visually show how the diffusion process works and how the particles move in the random walk.

2 Brief Theory about diffusion, and random walks

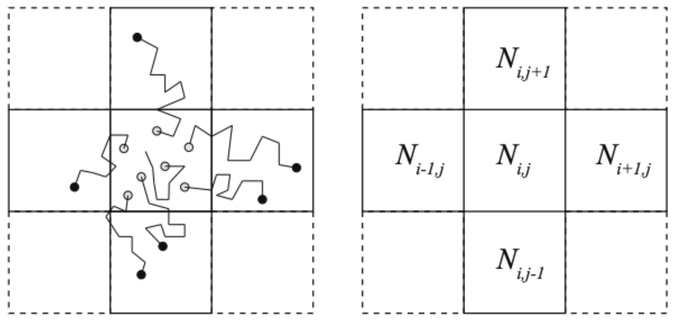


Figure 1: Random Walk in a mesh

Random walks are related to the diffusion process of various particles in a medium. This can be shown by observing random walkers in a mesh.

There is initially a number $N_{i,j}$ walkers in the center grid box, but after a few time steps, corresponding to some steps of the random walkers, some of the walkers have walked from $N_{i,j}$ and to its surrounding boxes, and some walkers have walked from the surrounding boxes and into $N_{i,j}$.

We can address this process in one dimension. In this case, we have a given number of walkers $N_i(t)$ in box i at time t. We assume that each walker has a probability $p = R\Delta t$ to leave the box and move into box $i + 1$ during a time interval Δt. This means that the number of walkers that moved from box i into box $i+1$ will be pN_i in the time interval Δt. Similarly, there is a number pN_i that moves from box i and into box $i-1$. There will also be walkers walking from box $i-1$ and into box i, and walkers moving from box $i+1$ into box i. Let us include these effects into an equation for N_i:

$$N_i(t+\Delta t) = N_i(t) - R\Delta t N_i(t) - R\Delta t N_i(t) + R\Delta t N_{i-1}(t) + R\Delta t N_{i+1}(t)$$

We collect terms on each side to get

$$N_i(t+\Delta t) - N_i(t) = R\frac{\Delta x^2}{\Delta t}(N_{i+1}(t) - 2N_i(t) + N_{i-1}(t))$$

We recognize this as the discrete form of a partial differential equation in $N(x,t)$:

$$\frac{\partial N}{\partial t} = R\frac{\Delta x^2}{\Delta t}\frac{\partial^2 N}{\partial x^2}$$

If we instead describe the system by the number of particles per box, $c(x,t) = N/V_b$, where V_b is the volume per box, we get the diffusion equation:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2}$$

where $D = R\Delta x^2$ is called the diffusion constant. [1, Intro to diffusion and random walks]

This can be extended to two dimesions also.

3 Outline of Methods used

For this project, we used a mesh simulation method in which we made the simulation of each particle at discrete time steps, and to get the probability density at any point, we divided the space into a mesh and the prob density = number of points that fell into the same box, as the required point over the total number of points.

4 Code and Results

The code has been made to simulate a diffusion process, in which the molecules are kept in a thin strip and then allowed to do a random walk across the board. There two types of boxes, that we will use

- Unboxed Diffusion: The particles are allowed to move out of the box mesh, producing simulation for a completely free environment
- Boxed Diffusion: The particles are not allowed to cross the box dimensions, and we bounce them back into the mesh

For each of the methods, we have created python classes named `Unboxed_Diffusion` and `Boxed_Diffusion` . Here we will show how to use these classes and run the simulation.

4.1 Simulation

We call the two respective function and input a dictionary containing the input variables. Then we can use the .simulate() function to run the simulation.

```python
Hyperparameters = {'W': 1, 'L': 5, 'N': 10**5, 'T': 10, 'D': 1,
        'delt': float(1/50), 'delx': float(2*(10**(-2))),
        'dely': float(10**(-2)), 'strip': float(0.05)}

from diffusion import Unboxed_Diffusion, Boxed_Diffusion

DM = Unboxed_Diffusion(Hyperparameters)
BDM = Boxed_Diffusion(Hyperparameters)
```

Listing 1: Simulation

This will start a progress bar telling the progress, of our program.

In the simulation process ,we initialise, an $N \times (T/delt) \times 2$ matrix to store the position of the particles at any time t. We initialise the matrix, by filling the points in the area of the strip. Then in each iteration (after time delt), we allow each particle to perform a random walk. This continues till time T/delt iterations are completed. The change in the boxed method is that we do not allow the particles beyond the boundry and reverse their motion, if they go beyond it in any step.

The parameters in the dictionary are:-

- W = width along the y axis
- L = length along the x axis
- N = number of particles
- T = total time of the diffusion process (in s)
- D = diffusion coefficient
- delt = small time interval between succesive random walks (in s)

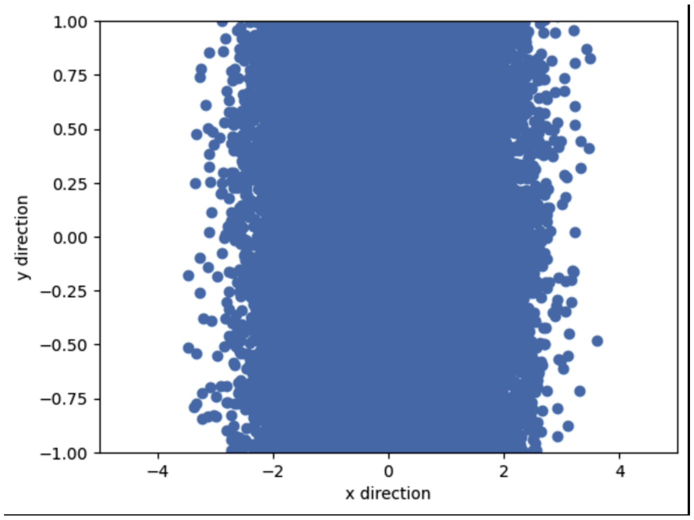
- delx = size of the mesh along x axis for calculating the probability
- dely = size of the mesh along y axis for calculating the probability

4.2 Plotting the diffusion

By calling .plot_diff, we can check the diffusion process at any step. Here is an example for the Unboxed diffusion. The time is not in seconds rather the number of iterations done

```
DM.plot_diff(time=20)
```

Listing 2: Main code

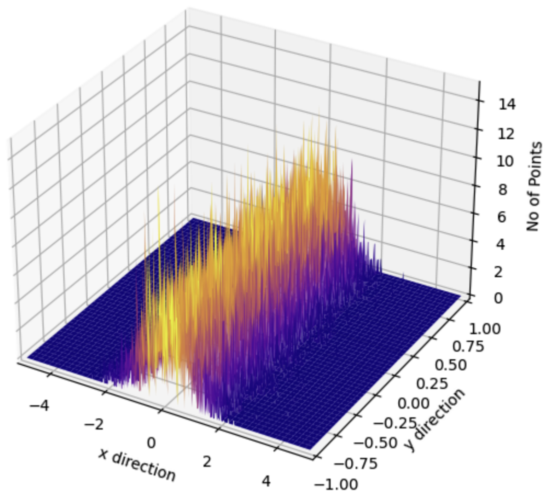


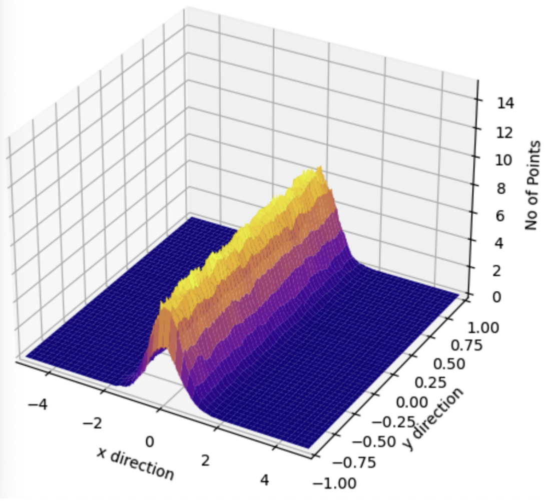
4.3 Plotting the probablity distribution

By calling the .plot_prob_3d, we can make a contour map of the surface of the probablity distribation at any time. This is done by first divinding the mesh into sub-blocks of size delx and dely. Then for each point, the probability of the box, it is in is incremented by one. This gives us the density, but it very rough and spicky as the points close to each have varying probability. To mitigate, this issue, we do a convolve over the probability matrix, to smoothen out the function.

```
BDM.plot_prob_3d(time=4, convolve=False)
BDM.plot_prob_3d(time=2, convolve=True)
```

Listing 3: Main code





4.4 Getting Probability

By using the .get_prob_3d, we get the probability in any box. We input the (x,y) positon of the point whose probability we want and then, we locate the box, it is in and return the probablity.

```
DM.get_prob_3d(xpoint = 0, ypoint = 0.5, time = 34, convolve = False)[0]
```

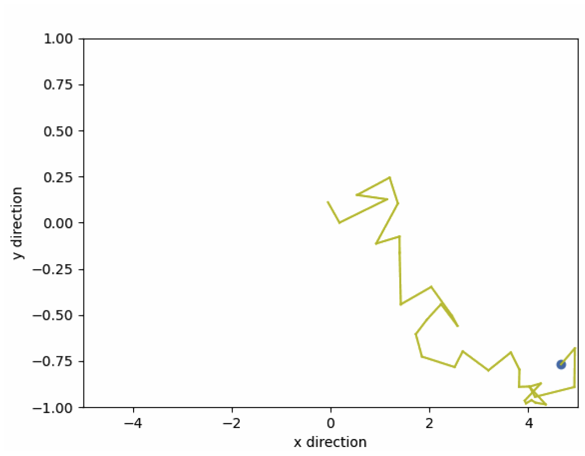
Listing 4: Main code

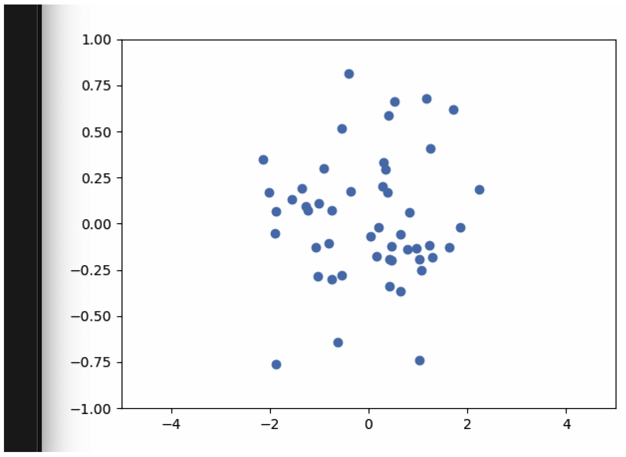
4.5 Animated Path

By using the .animated_path and entering the index number of the point, we want to track, we get a video of the motion of the particle. It traces where the particle goes using a yellow line an follows the particle. We can also use the .animate_manypoints to make a video of how the points go and diffuse through the medium.

```
BDM.animated_path(5550, name = "plots/animated_boxed_path.gif")
BDM.animate_manypoints()
```

Listing 5: Main code





4.6 Video of Probability density evolution

Using the .animate_prob, we get the images of the probabilty distribution plot at many intervals of time, and then we can use an online movie maker to convert it into a video gif.

```
BDM.animated_prob()
```

Listing 6: Main code

5 Code

You can either use the ipython notebook, given in the repository (testing.ipynb) (recommended), or use the python file (trial.py).

Note: If you are using the python file, then uncomment all the neccesary operations, you want to perform using the file.

6 Team Details

Name:	**Roll No**	**Branch**
Gaurav Joshi	21110065	Cse
Rutwik More	21110133	Cse
Adit Kaushik	21110010	Cse
Arun Reddy	21110029	Cse
Aditya Deshmukh	21110014	Me

References

[1] Intro to random walks and diffusion Link