# PHY566 Group Assignment

Group A

Tong Zhu, Jiyingmei Wang, Aritro Pathak, Ankur Manikandan

April 2, 2015

**Abstract**

A random walk is a mathematical formalization of a path that consists of a succession of random steps. The term random walk was first introduced by Karl Pearson in 1905. Random walks have been used in many fields: ecology, economics, psychology, computer science, physics, chemistry, and biology. Random walks explain the observed behaviors of many processes in these fields, and thus serve as a fundamental model for the recorded stochastic activity.

One of the main applications of random walk is to trace the path of a molecule as it travels in a liquid or gas. These molecules are constantly buffeted by the molecules around them, a consequence of the molecular nature of matter. They exhibit random walks, where each "step" is in a random direction relative to the one that preceded it.

The type of motion that describes such random motion of particles is known as, Brownian motion. Brownian motion is among the simplest of the continuous-time stochastic (or probabilistic) processes, and it is a limit of both simpler and more complicated stochastic processes.

## 1 2D random walk

### 1.1 Introduction

In this part, we would do a problem about random walk in 2 dimensions, taking steps of unit length in ±x or ±y direction on a discrete square lattice.

### 1.2 Results

In this part,we plot $\langle x_n \rangle$ and $\langle {x_n}^2 \rangle$ up to n=100 by averaging over at least 104 different walks. The result are shown in Fig.2.

You can see that the $\langle x_n \rangle$ is always zero which is agree with our expectation. For the random walk is walking randomly along +x direction and -x direction, so the average of that would be zero after many times of random walks. As you can see that correlation

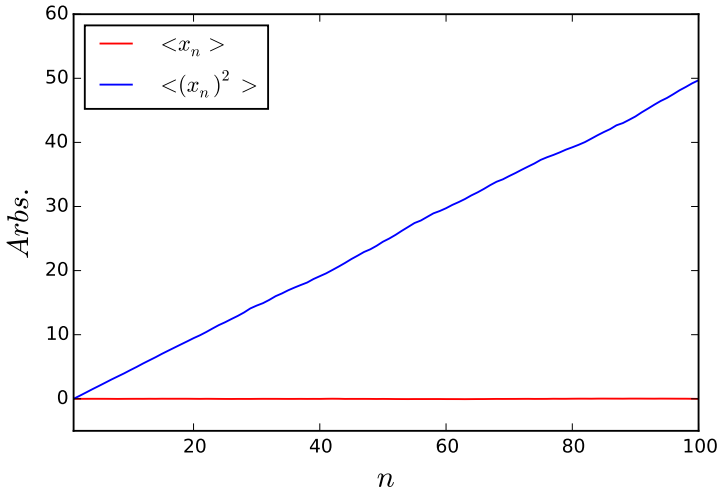


Figure 1: $\langle x_n \rangle$ **and** $\langle {x_n}^2 \rangle$ **up to n=100 by averaging over at least 104 different walks.**

between $\langle {x_n}^2 \rangle$ and n is linearly, and the slope is like 0.5. The reason for that is that the random walks in two axis, neither ±x direction or ±y direction, so the probability of walking in ±x direction is almost 0.5 which is also agree with our results.

By seeing the mean square distance from the starting point ¡ r2 ¿ up to 100 times by averaging over at least 104 different walks, shown in Fig.**??**, we can see that the motion is diffusive and the value of the diffusion constant after an curve fit is like 0.99995 which is very close to 1. The reason for that the mean square distance from the starting point stands for the walk of two axis (x and y), so the probability of that is equal to 1 which is agree with our plot.

# 2 Diffusion Equation

## 2.1 Introduction

In this section, we will solve the diffusion problem both analytically and numerically given the initial density distribution.

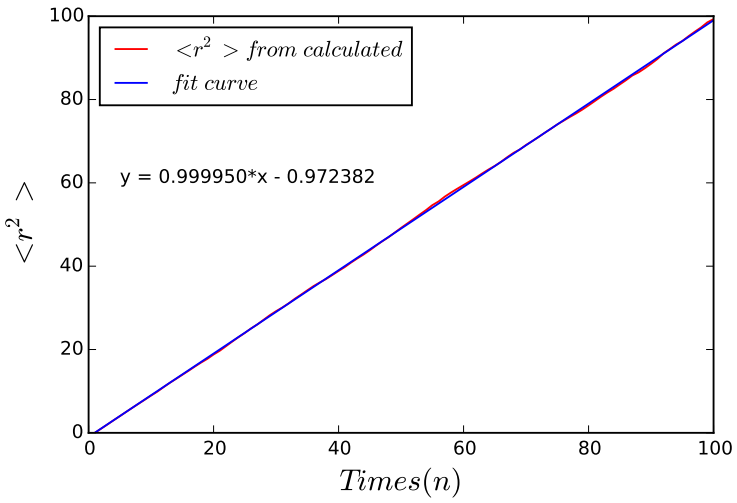


Figure 2: $\langle r^2 \rangle$ **up to n=100 by averaging over at least 104 different walks.**

## 2.2 Method

### 2.2.1 Part(a)

In this section, the spatial expectation value for $\langle x(t)^2 \rangle$ is determined analytically, given a 1D Normal Distribution$\rho(x,t)$ shown in Eq.(1)

$$\rho(x,t) = \frac{1}{\sqrt{2\pi\sigma(t)^2}} e^{-\frac{x^2}{2\sigma(t)^2}} \tag{1}$$

In general, the expected value of a measurable function of $x$, $g(x)$, given that x has a probability density function $f(x)$, is given by the inner product of f and g.

$$\langle g(x) \rangle = \int_{-\infty}^{\infty} g(x) f(x)\, \mathrm{d}x$$

In this case, we substitute $g(x)$, and$f(x)$ with $x(t)^2$ and $\rho(x,t)$ respectively. For every instant, t can be regarded as a constant in the integrand, hence the expected value for $x^2$ is a function of t.

$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 \frac{1}{\sqrt{2\pi\sigma(t)^2}} e^{-\frac{x^2}{2\sigma(t)^2}}\, \mathrm{d}x = \sigma(t)^2$$

### 2.2.2 Part(b)

In this subsection, a program is written to solve the 1D diffusion equation using the finite difference form with a diffusion constant D = 2, with an initial density profile that is sharply peaked around x = 0, but extends over a few grid sites (box profile). To avoid the determination of the number of particles that are involved in this diffusion system, we use the notion of probability to study such system. Fig.3 illustrates the initial probability density.

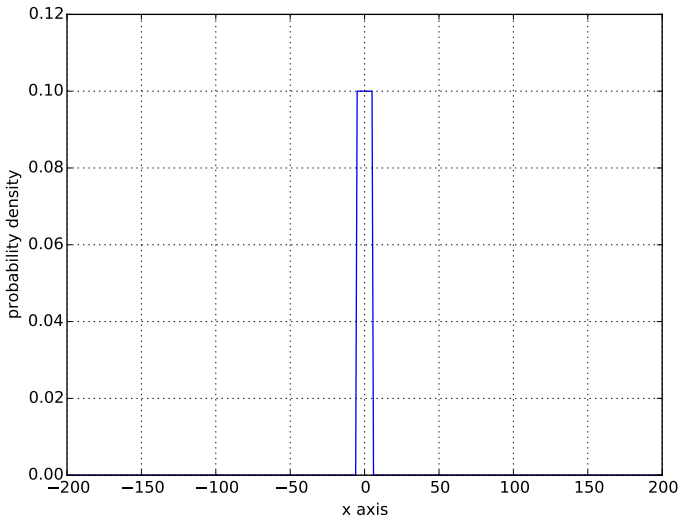


Figure 3: **Initial probability density that is sharply peaked around x = 0, but extends over a few grid sites (box profile)**

Then we apply the finite difference form of the diffusion equation to solve the problem numerically.

$$\rho(t+dt,x) = \rho(t,x) + D\frac{\Delta t}{(\Delta x)^2}[\rho(t,x+dx) + \rho(t,x-dx) - 2\rho(t,x)]$$

After that, we fit the calculated densities at later time equal to 50s, 100s, 300s, 500s and 900s corresponding to a Normal Distribution with $\sigma(t) = \sqrt{2Dt}$.

## 2.3 Results

Fig.4 shows the results. Over these 5 different time snapshots, significant changes of the distribution are visible. The peak of the density of probability spreads out with time. And

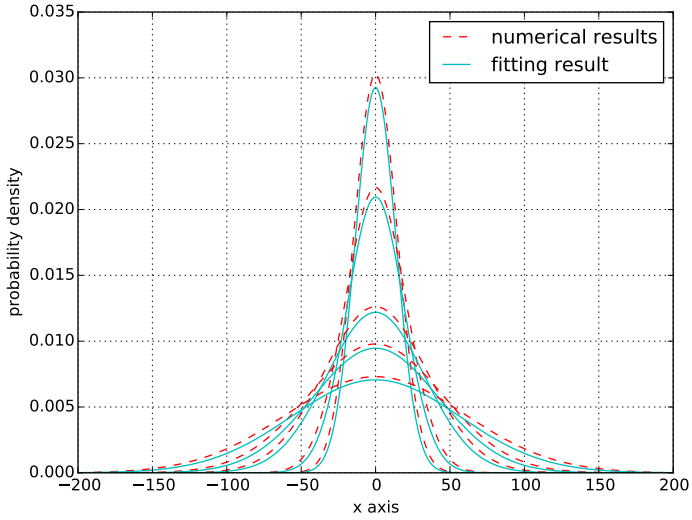


Figure 4: **Numerically calculated probability densities with the fitting line at different times**

it can be seen from raw eyes that all the numerically calculated densities overlaps well with the the normal distribution with $\sigma(t) = \sqrt{2Dt}$.

To be more precise, we could also compare the fitting parameter with the idea Normal Distribution with $\sigma(t) = \sqrt{2Dt}$. Table.1 contains these values. By comparing the analytical and fit results we can verified that the numerically calculated density profile corresponds to a Normal Distribution with $\sigma(t) = \sqrt{2Dt}$ at later times.

Table 1: Analytical and fit parameters for density distribution

| Time[s] | $\mu$(analytical) | $\mu$(numerical) | $\sigma$(analytical) | $\sigma$(numerical) |
|---|---|---|---|---|
| 500 | 0 | $-1.74\times10^{-8}$ | 14.14 | 13.63 |
| 1000 | 0 | $-1.21\times10^{-8}$ | 20.00 | 19.04 |
| 3000 | 0 | $-2.01\times10^{-8}$ | 34.64 | 32.71 |
| 5000 | 0 | $1.19\times10^{-8}$ | 44.72 | 42.15 |
| 9000 | 0 | $-2.74\times10^{-8}$ | 60.00 | 56.49 |

# 3 Mixing of two Gases

In the Group Assignment 1, we are asked to use random walks to simulate the mixing of two gases in 2D in a rectangular enclosure. Also, we are asked to understand the variation of linear population densities with select time-intervals. The questions posed in the assignment are as follows.

## 3.1 Introduction

Use the techniques and insights gained previously to write a program to simulate the mixing of two gases in 2D in a rectangular enclosure:

(a) Set up a 2D grid in xy space with dimensions $600 \times 400$. Fully populate the left third of the grid with a gas of species "A" and the right third of the grid with a gas of species "B". The center third of the grid remains empty. Pick a random location on the grid, and have the gas particle move at random one position up/down/left/right. If the selected position is occupied, reject the move and pick another particle (you may optimize the algorithm by only picking from occupied sites). Repeat for a large number of iterations.

(b) Plot the linear population densities $n_A(x)$ and $n_B(x)$ after select time-intervals. Also plot a few sample configurations of the grid at various time-steps.

(c) Average the densities over 100 trials for added accuracy and replot the densities.

## 3.2 Methods

We generated a grid of size 600 x 400 by populating the left third of the grid with gas species "A" and the right third with a gas species "B", leaving the centre of the grid empty. To pick a random location, that is an occupied site, on the grid we used the "random" function from the 'random' library.

We considered three different cases to move the randomly selected particle:

- $Case1$ - If the particle is at the four corners of the grid, then it has only two degrees of freedom.
- $Case2$ - If the particle is along the border of the grid, excluding the edges, then it has three degrees of freedom.
- $Case3$ - If the particle is situated within the interior of the gird, then it has all four degree of freedom.

In each case, the particle has an equal probability of moving in it's permissible directions. The random motion of the particles is repeated over a large number of iterations.

Also, the question asked us to plot the linear population densities after select time-intervals. The population densities are calculated by counting the number of gas molecules of types "A" and "B" independently along the x-axis.

For the last part of the question, we average the densities over 100 trials using a "for" loop and replot the densities at various time-steps.

### 3.3 Results

## 4 Modified method for Gas mixing problem

### 4.1 Method

The normal way to do this question can be done by the following ways:

- Find the position of the gas particle randomly
- move the gas particle randomly.

As the definition of our movement, if the positions around the particle has no empty states, then this particle would not be move. This is an waste iteration in the algorithm. According to this, we can add one acceleration method to this problem. That is only selecting the gas particle which its around have at least one empty states randomly.

On the other hand, let's we consider more about this question. The gas particle only move to the empty direction randomly. When this move happened, the coordinate of the gas states and the empty states exchanged. So this question is basic equal to the empty states move randomly.

If we think like this, we can solve this question by the following ways:

- find the position of the empty states randomly
- move the empty states randomly.

The same as the acceleration of gas particle movement, we can still cancel the waste iteration by just choose the empty states randomly for those which its neighbor have at least one gas states.

According to this, our code is like below (the choice function and the move function):

The choice function give you the coordinates of the randomly choice empty states positions, and then move functions according to this coordinates to change the matrix of all gas states and empty states.

As in the choice and the move, we would encounter the problem about the boundary (the movement of the corner or the axis). In this question, we still permit the code to

```
def choice5(nx,ny,origin):
    c=0
    while True:
        x=randrange(0,nx)
        y=randrange(0,ny)
        if origin[x,y] == 0 :
            if x < nx-1:
                if origin[x+1,y]!=0:
                    c=1
            if x >0:
                if origin[x-1,y]!=0:
                    c=1
            if y< ny-1:
                if origin[x,y+1]!=0:
                    c=1
            if y>0:
                if origin[x,y-1]!=0:
                    c=1
            if c==1:
                c=0
                break
    return x,y
```

```
def move2(nx,ny,i,j,origin):
    #corresponding to the move of choice 2 and choice 3,
    #after that,return the charge density
    c=random()
    if 0.0  <= c <= 0.25:
        if i<nx-1:
            if origin[i+1,j]!= 0:
                origin[i,j]=origin[i+1,j]
                origin[i+1,j]= 0
    elif 0.25 < c <= 0.5:
        if i>0:
            if origin[i-1,j] != 0:
                origin[i,j] = origin[i-1,j]
                origin[i-1,j] = 0
    elif 0.5 < c <= 0.75:
        if j<ny-1:
            if origin[i,j+1]!= 0:
                origin[i,j]=origin[i,j+1]
                origin[i,j+1] = 0
    else:
        if j>0:
            if origin[i,j-1] != 0:
                origin[i,j] = origin[i,j-1]
                origin[i,j-1] = 0
    return origin
```

Figure 5: The choice function and the move function of the empty states.

randomly choose that direction to move, but we don't change the matrix at all. You can see that is another waste iteration. By this, we do not need to specify the corner atoms or axis atoms specifically. But just change the matrix if the gas/empty states can move.

The logic is like below:

- if we randomly choose the +y direction to move, we only change the gas/empty states which its y corrdinate is lower than the boundry(ny-1 in the code).
  - The reason for that is if we randomly choosing y coordinate is in the +y boundary, and the randomly choice direction to move is +y, then this move cannot happen, we don't change anything.
- if we randomly choose the -y direction to move, we only change the gas/empty states which its y corrdinate is higher than the boundry(0 in the code).
- if we randomly choose the +x direction to move, we only change the gas/empty states which its x corrdinate is lower than the boundry(0 in the code).
- if we randomly choose the -x direction to move, we only change the gas/empty states which its x corrdinate is higher than the boundry(0 in the code).

The whole solution code is like below:

```
if method==3:
    #random choose empty whose neighbor have gas, then move it
    for i in range (0,niter):
        x,y = choice5(nx,ny,origin_old)
        origin_new = move2(nx,ny,x,y,origin_old)
        origin_old = np.copy(origin_new)
return origin_new
```

Figure 6: solution iteration of the code.

### 4.2 Gas Mixing-problem (c) based on modified method

For the Mixing-problem, when we use the acceleration method, we can get the mixing states more quickly, we use 60×40 parts as a test case, For this case, we only need $10^6$ iterations (the normal method needs $10^7$ iterations to get this states) to get the whole mixing states like below:

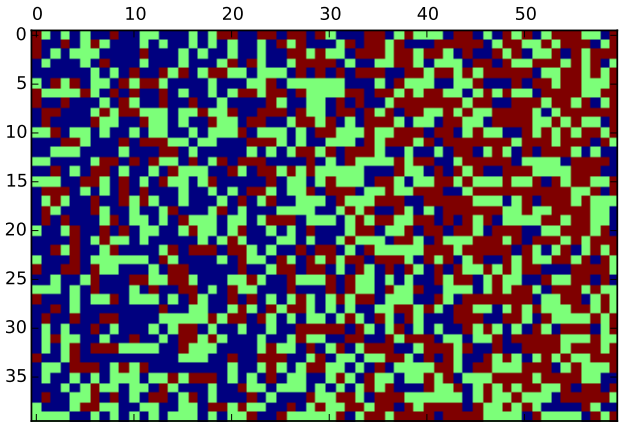


Figure 7: The gas mixing of 60×40 dimension grids, blue stands for gas A, red stands for gas B, and green stands for the empty.

We difine the density like below, for each x grid point (x coordinate), accumulate all the y coordinate, get the number of gas A or gas B numbers. Then use this number divided by the whole gas A or gas B numbers. The density is show like below:

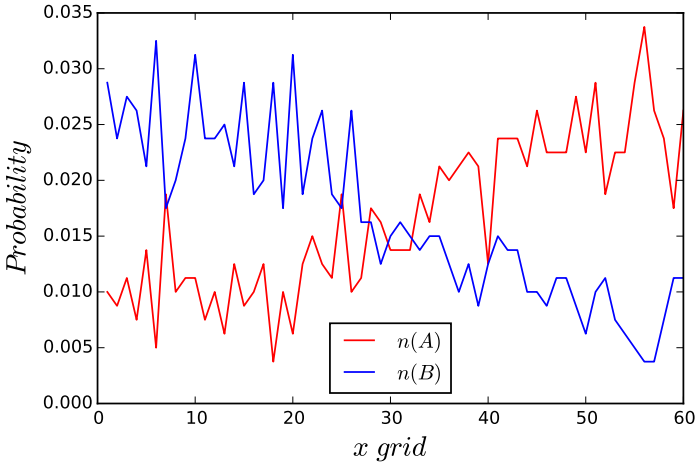


Figure 8: The density of 60×40 dimension grids when it close to the whole mixing states.