Lecture 9

Random Processes

Rubem Mondaini

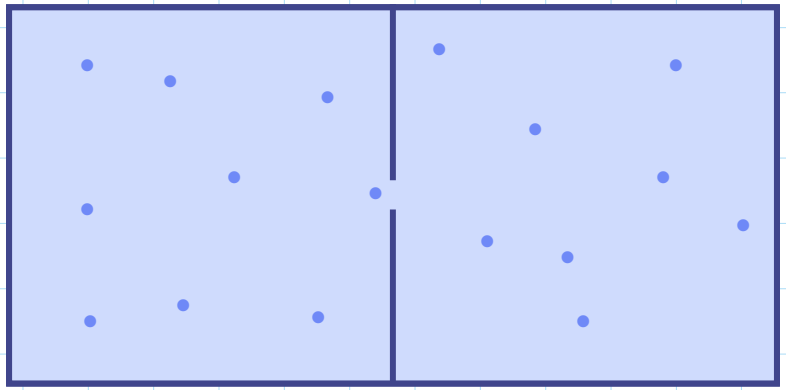
Beijing Computational Science Research Center

→ based on Hai-Qing Lin's Lectures and notes

- This PowerPoint Notes are based on the Textbook '***An Introduction to Computer Simulation Methods : Applications to Physical Systems***', 2nd Edition, Harvey Gould and Jan Tobochnik, Addison-Wesley(1996);
  - → "A First Course in Computational Physics";
  - → "Numerical Recipes";
  - → "Elementary Numerical Analysis";
  - → "Computational Methods in Physics and Engineering".

Order to Disorder

- Random process is a ***non-deterministic*** process.
- The basic assumption underlying the probabilistic model is that the motion of the particles is so complex that we can assume random behaviour.
- Example: *N* identical particles in a box.



Questions and objective

- Random processes are common in nature, how do we simulate them?
- Most computer simulations use random number generator. What is it? How to use it? What are its general features?
- Random walk and its connection to physical systems.
- Data analysis, least squares (LSQ). We already saw a bit of this...
- More on random walks.

N particles in a box

- Assume the no. of particles on the left side is n_{left}, then the probability per unit time that a particle moves from left to right is n_{left}/N.

- Thus we can simulate the motion of particles according to the following algorithm:
  - → Generate a random number to choose a particle at random
  - → Move a particle to the other side of the box
  - → Give the particle a random position on the new side of the box (visualization)
  - → Increase the "time" by unity.

Straightforward implementation of this algorithm

Python + Matploltib + Numpy

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

##########################################################
### This very simple script aims in emulating the ########
###  expansion of a gas in a random process. #############
##########################################################

##########################################################
##### Drawing setup ######################################
##########################################################
fig, axs = plt.subplots()
plt.rc('text', usetex=True)
plt.rc('font', family='serif')

axs.set_aspect('equal')
plt.axis('off')

### parameters ###########################################
L = 400  ## Box linear size
N = 300 ## Number of particles
max_time = N * 2
```

Straightforward implementation of this algorithm

```
### This routine draws the box ##############################
def draw_initial():

    linestyle = '-'
    linewidth = 2.0
    line_color = 'dodgerblue'

    axs.plot([0.0, 2*L], [0.0, 0.0], linestyle = linestyle, \
    linewidth = linewidth, color = line_color)
    axs.plot([0.0, 2*L], [L, L], linestyle = linestyle, \
    linewidth = linewidth, color = line_color)

    axs.plot([0.0, 0.0], [0, L], linestyle = linestyle, \
    linewidth = linewidth, color = line_color)
    axs.plot([2*L, 2*L], [0, L], linestyle = linestyle, \
    linewidth = linewidth, color = line_color)

    axs.plot([L, L], [0, L/2. - 0.05*L], linestyle = linestyle, \
    linewidth = linewidth, color = line_color)

    axs.plot([L, L], [L/2. + 0.05*L, L], linestyle = linestyle, \
    linewidth = linewidth, color = line_color)
```

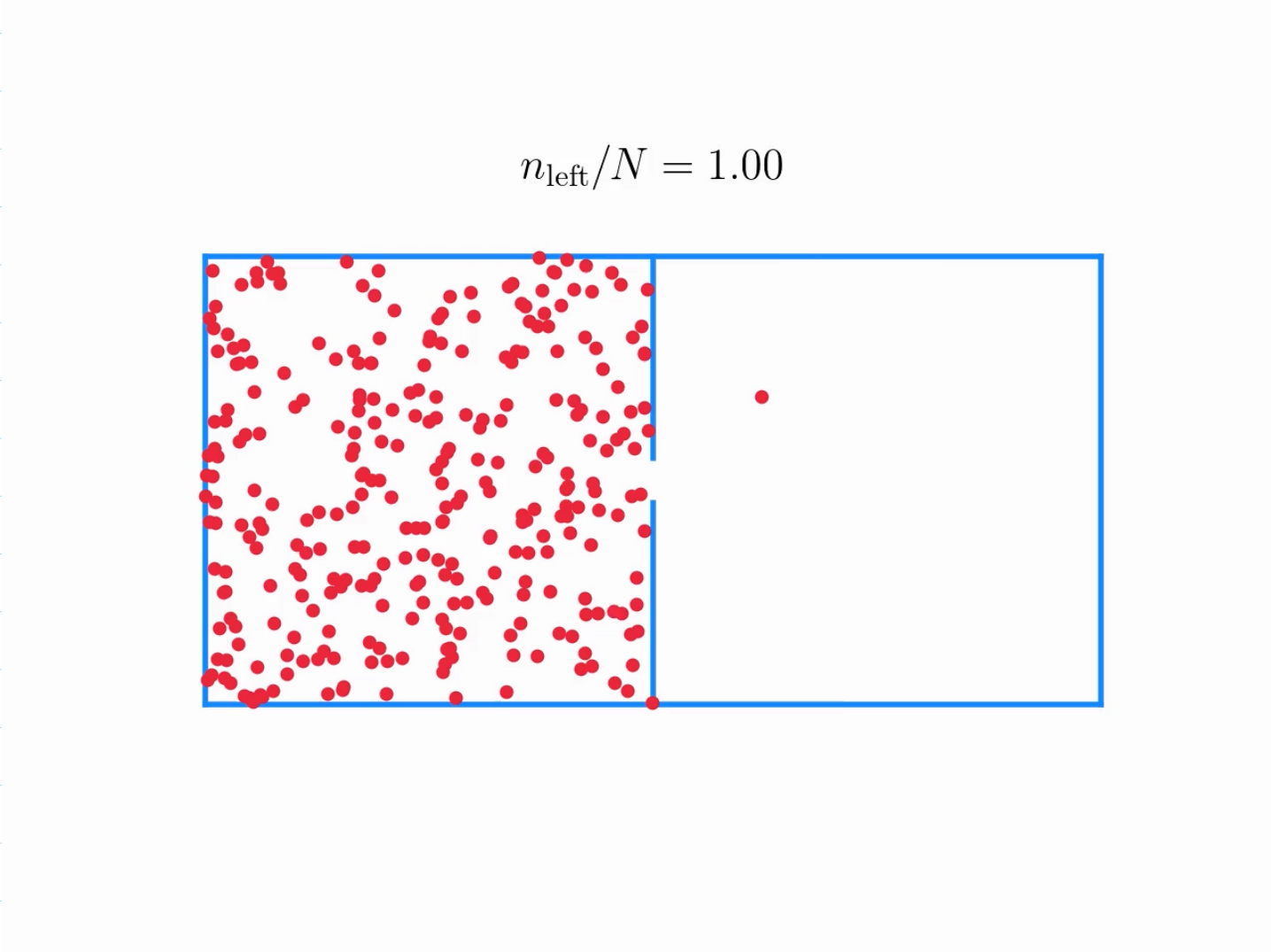
Straightforward implementation of this algorithm

```
# = This routine initializes a random configuration in the left side #
def initialize(N, L):

    x = L*np.random.rand(N)     ### N random numbers in [0,L)
    y = L*np.random.rand(N)

    nleft = N

    return x, y, nleft

# === This routine changes particle positions in the box ========== #
def step(N, L):

    global nleft

    rnd_particle = np.random.randint(N)

    if (x[rnd_particle] < L):
        nleft-=1    ### move to right
        x[rnd_particle] = L*np.random.rand() + L    ### random number in
                                                    ###     [L,2L)
        y[rnd_particle] = L*np.random.rand()
    else:
        nleft+=1    ### move to left
        x[rnd_particle] = L*np.random.rand()    ### random number in [0,L)
        y[rnd_particle] = L*np.random.rand()

    return x, y, nleft
```

Straightforward implementation of this algorithm

```
### Here starts the fun ############
draw_initial()
time_range = range(0, max_time)

x, y, nleft = initialize(N, L)

particles, = axs.plot(x, y, linestyle = 'none', marker = 'o', color =
'crimson', markersize = 4)

def animate(time_range):

    x, y, nleft = step(N, L)

    particles.set_data(x, y)

    return particles, nleft_text

#### Updating animation
anim = animation.FuncAnimation(fig, animate, time_range,\
    interval=20, repeat=False, blit=False, init_func=init)

## Set up formatting for the movie files
Writer = animation.writers['ffmpeg']
writer = Writer(fps=15, metadata=dict(artist='Me'), bitrate=18000)

anim.save('N%dL%d.mp4' %(N, L), fps=15, dpi=200)
```

Resulting animation



Questions to be Answered

- How long does it take for the system to reach the equilibrium?
- How does this time depend on particles number?
- What is the magnitude of the ***fluctuations***?

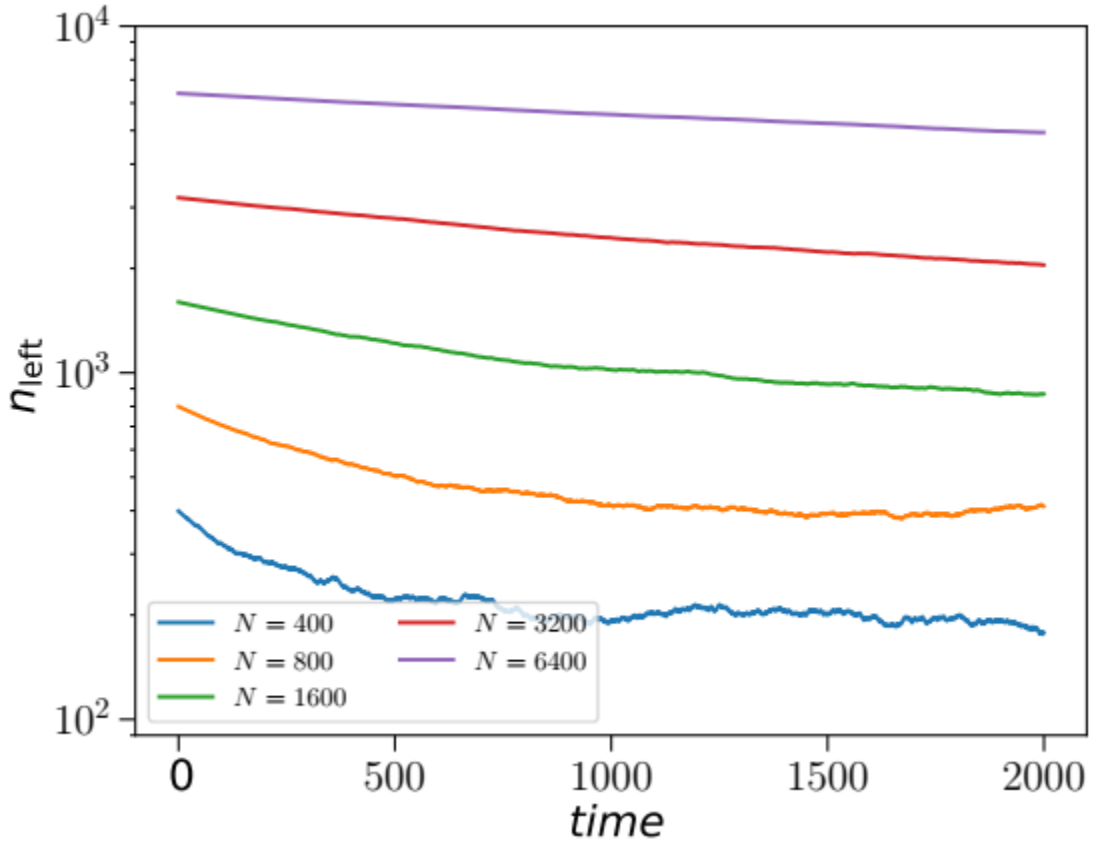
If there are $n(t)$ particles on the left side, then the change in $n(t)$ in time interval Δt is:

$$\frac{\Delta n}{\Delta t} = -\frac{n(t)}{N} + \frac{N - n(t)}{N} \quad \Rightarrow \quad \frac{dn}{dt} = 1 - \frac{2n(t)}{N}$$

→ $n(t) = N/2\,[1 + \exp(-2t/N)]$

where the initial condition is $n(t = 0) = N$.

Indeed!



Question: relaxation time?

$$n(t) - N/2 = 1/e \rightarrow t = ?$$

Monte Carlo

- Such probabilistic method for simulating the approach to equilibrium is an example of **MONTE CARLO** methods.

→ the random sampling of the most probable outcomes.
An alternative method is to use exact enumeration and determine the possibilities at each time interval.

- Monte Carlo methods are especially useful for large *N*, why?

- Due to its random nature, we need to do Monte Carlo simulation **many times** and average over the results to obtain meaningful averages.

Introduction to Random Walk

- So far we have considered random processes that do not care about the positions of the particles in either side of the box
- All we needed was the number of particles in each side. What if we do care now?

Drunken sailor problem:

- A drunkard begins at a lamp post and takes *N* steps of equal length in random directions, how far will the drunkard be from the lamp post?
- Such motion is an example of **random walk**.
- Physically, this is similar to the trajectory of a particle that can move in any direction with equal probability.

One-dimensional Random Walk

- Suppose the walker begins at $x = 0$ and that each step is of equal length l.
- At each time interval, the walker has a probability p of a step to the right and a probability $q = 1 - p$ of a step to the left.
- The direction of each step is **independent** of the preceding one.
- Then the displacement and its square of a walker after *N* steps are

$$x(N) = \sum_{i=1}^{N} s_i \text{ and } x^2(N) = \left(\sum_{i=1}^{N} s_i \right)^2 \text{ where } s_i = \pm l$$

One-dimensional Random Walk

$$\langle x(N)\rangle = \sum_{i=1}^{N} (pl - ql) = (p - q)\, lN \text{ and}$$

$$\langle x^2(N)\rangle = \sum_{i=1}^{N} \langle s_i^2 \rangle + \sum_{i \neq j=1}^{N} \langle s_i s_j \rangle = Nl^2 + N(N-1)(p-q)^2 l^2$$

$$\langle \Delta x^2(N)\rangle \equiv \left\langle \left(x(N) - \langle x(N)\rangle\right)^2 \right\rangle$$

Mean square net displacement

$$= \langle x^2(N)\rangle - \langle x(N)\rangle^2$$

$$= Nl^2 + N(N-1)(p-q)^2 l^2 - N^2 (p-q)^2 l^2$$

$$= \left[1 - (p-q)^2\right] Nl^2$$

$p + q = 1 \Rightarrow p^2 + q^2 + 2pq = 1$

$$= 4pqNl^2$$

What are we interested?

- **Asymptotic scaling law:**

  for sufficiently large *N*, one has

  $$\langle \Delta x^2(N) \rangle \sim N^{2\nu}$$

  For 1D, $\nu = 1/2$.

  For 2D and 3D, ν =?

- **Correlation among walkers:**

  < x(i) x(j) >, etc.

- **Simulation technique**

Performing Monte Carlo simulation of the random walk is simple, the difficult part is associated with the bookkeeping.

→ Use of array is one of the useful tricks.

```
PROGRAM random_walk
! Simulation of a random walk in one dimension

DIM xcum(64), x2cum(64)

CALL initial (p,N,xcum(),x2cum(),ntrial)
FOR itrial = 1 to ntrial
        CALL walk(p,N,xcum(),x2cum())
NEXT itrial

CALL output(N,xcum(),x2cum(),ntrial)

END
```

```
SUB initial(p,N,xcum(),x2cum(),ntrial)
      RANDOMIZE (initialize random number sequence)
      IPUT prompt " maximum number of steps N " : N
         LET p = 0.5
      LET ntrial = 1000              !Number of trials
      FOR istep = 1 to N
            LET xcum(istep) = 0  ! Not necessary in True Basic
            LET x2cum(istep) = 0
      NEXT istep
END SUB

SUB walk(p,N,xcum(),x2cum())
       LET x = 0                          ! Initial position of walker for each
   trial
       FOR istep = 1 to N
             IF rnd <= p then
                LET x = x +1
             ELSE
                LET x = x - 1
             END IF
             ! collect data after every step
             CALL data(x,xcum(),x2cum(),istep)
        NEXT istep
END SUB
```

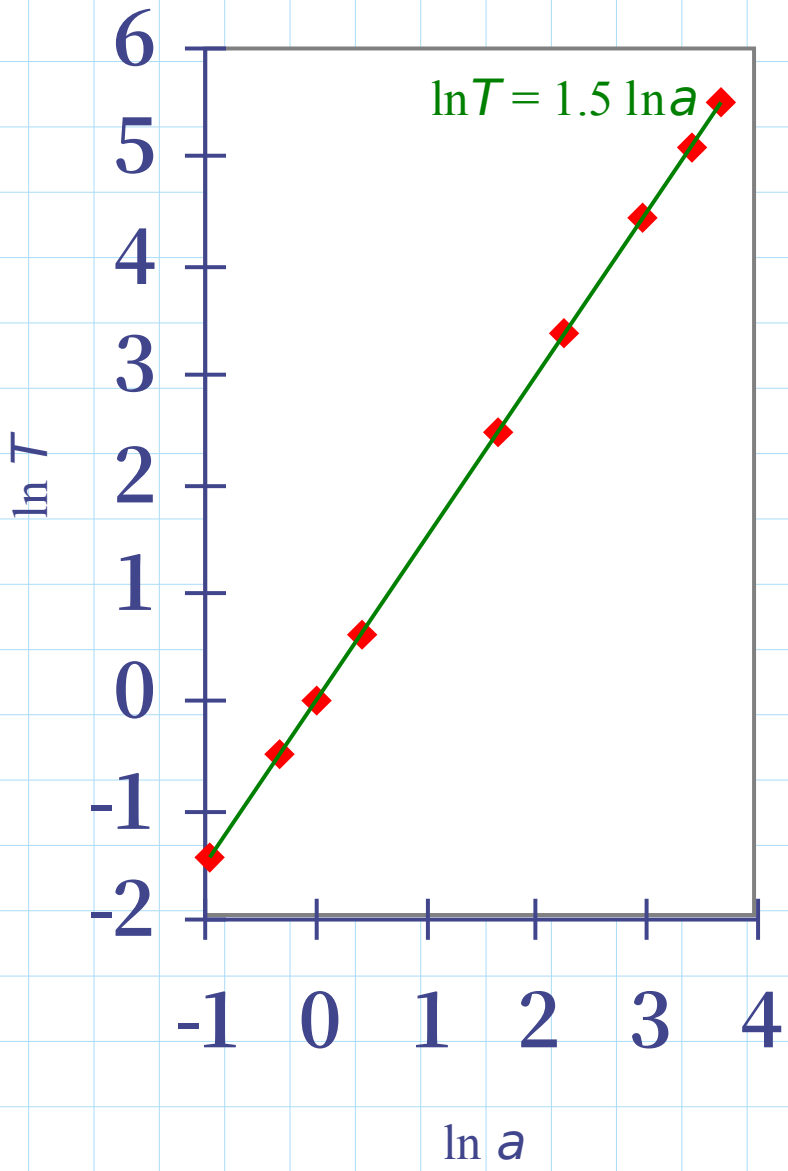
```
SUB data(x,xcum(),x2cum(),istep)
      LET xcum(istep) = xcum(istep) + x
      LET x2cum(istep) = x2cum(istep) + x*x
END SUB

SUB output(N,xcum(),x2cum(),ntrial)
       PRINT "# steps","<x>","<x^2>","<x^2>-<x>^2"
       PRINT
       FOR istep 1 to N
              LET xbar = xcum(istep)/ntrial
              LET x2bar = x2cum(istep)/ntrial
              LET variance = x2bar -  xbar*xbar
              PRINT istep, xbar, x2bar, variance
        NEXT istep
END SUB
```

Data Analysis

- After simulation, with a collection of data (*x*(i), etc.), how do we get useful information?
- This is a necessary step in all experiments, e.g., coffee cooling constant, Kepler's law.
- Graphic is an intuitive approach. However, it could be biased. Is there any analytical approach?

Plot of ln T vs ln a

Planet	T (earth years)	a (AU)
Mercury	0.241	0.387
Venus	0.615	0.723
Earth	1.00	1.00
Mars	1.88	1.523
Jupiter	11.86	5.202
Saturn	29.5	9.539
Uranus	84	19.18
Neptune	165	30.06
Pluto	248	39.44



Why 1.5, not 1.53 or 1.588???

Method of Least Squares

- A common method for finding the best straight line fit to a series or measured points (linear regression).
- Suppose that we have n pairs of measurements (x_1,y_1), (x_2,y_2), $\cdots$, (x_n,y_n) and that the errors are entirely in the values of y.
- We want to obtain the best fit to the linear function

$$y = mx + b, \qquad y = \sum_{k=0}^{p} a_k x^k .$$

Covered, see Data_Modeling.pdf

Random Walks & the Diffusion Equation

- One reason random walks are very useful in simulating many physical processes and modelling many differential equations of physical interest is that their behaviour is closely related to the solution of the ***diffusion equation***.
- Consider the one-dimensional diffusion equation

$$\frac{\partial P(x,t)}{\partial t} = D \frac{\partial^2 P(x,t)}{\partial x^2},$$

where D is the self-diffusion coefficient and $P(x,t)dx$ is the probability of a particle being in the interval between x and $x+dx$ at time t.

Random Walks & the Diffusion Equation

At infinity position, *P* is *zero*, $P(x = \pm\infty, t) = 0$, and all the spatial derivative of *P* are *zero*,

$$\left.\frac{\partial P(x,t)}{\partial x}\right|_{x=\pm\infty} = 0.$$

The average of any function of x can be written as

$$\langle f(x,t)\rangle = \int_{\infty}^{\infty} dx f(x)\, P(x,t).$$

Multiply both sides of the diffusion equation by x and performing integration:

$$\int_{\infty}^{\infty} dx\, x \frac{\partial P(x,t)}{\partial t} = D\int_{\infty}^{\infty} dx x \frac{\partial^2 P(x,t)}{\partial x^2}.$$

Random Walks & the Diffusion Equation

The left-hand side is

$$\int_{-\infty}^{\infty} dx x \frac{\partial P(x,t)}{\partial t} = \frac{\partial}{\partial t} \int_{-\infty}^{\infty} dx x P(x,t) = \frac{\partial}{\partial t} \langle x \rangle .$$

We do integration by parts to the right hand side

$$D \int_{-\infty}^{\infty} dx x \frac{\partial^2 P(x,t)}{\partial x^2} = Dx \frac{\partial P(x,t)}{\partial x} \Bigg|_{x=-\infty}^{x=\infty} - D \int_{-\infty}^{\infty} dx \frac{\partial P(x,t)}{\partial x}.$$

Both terms are zero and we find that

$$\frac{\partial}{\partial t} \langle x \rangle = 0 \quad \Rightarrow \quad \langle x \rangle = 0 \quad \text{for all } t.$$

Random Walks & the Diffusion Equation

- Similarly, we would get (two integrations by parts)

$$\frac{\partial}{\partial t}\left\langle x^2 \right\rangle = 2D \quad \Rightarrow \quad \left\langle x^2(t) \right\rangle = 2Dt$$

- The random walk and diffusion equation have the same time dependence if we identify t with $N\Delta t$ and $2D$ with $l^2/\Delta t$.

  Recall that $\langle \Delta x^2(N) \rangle = 4pqNl^2 \Rightarrow \langle \Delta x^2(N) \rangle = Nl^2$

- In three dimensions the second derivative $\partial^2/\partial x^2$ is replaced by the Laplacian operator ∇^2. $2D$ is replace by $2dD$.

Other Topics

- The Poisson Distribution & Nuclear Decay
- Problems in Probability
  (*see references and random number*)
- Method of Least Squares
  (*see LSQ*)