Experimental biophysics Spring 2021: Brownian motion

Lab Supervisors:

Esra Yilmaz

(Email: esra.yilmaz@ftf.lth.se)

Elham Akbari

(Email: elham.akbari@ftf.lth.se)

2021-03-19

Theory: Brownian Motion

Stochastic and deterministic motions:

Brownian motion is the motion of small particles diffusing in a fluid. This motion is highly stochastic for which exact predictions are not possible. In this motion, suspended particles are being constantly hit by molecules in the surrounding fluid. These molecules hit from all possible directions and change the path of the particles instantaneously at the moment of impact. This means that in comparison to deterministic motions which are predictable, different mathematical methods are needed to characterize and analyze stochastic process.

To get familiar with this motion, figure 1 and 2 gives us an example of something that is deterministic and stochastic respectively. As we can see from figure 1, at any given time t, we can find the exact value of Y. This is called a deterministic process. On the other hand, an example of stochastic process demonstrated in figure 2. In this kind of process, we cannot specify an exact value to the dependent variable (Y) as a function of an independent variable (t). We only can specify the probability that the dependent variable has a value (ex. Yi) at a given value of t (ex. Ti). If we consider some later time t (ex. Ti+τ), we can see that Y can in principle have any value with the probability defined at the time t. For the special case when the probabilities are invariant under a shift in time, we say that the stochastic process is steady. In such cases, only differences in time are important, not the absolute time values.

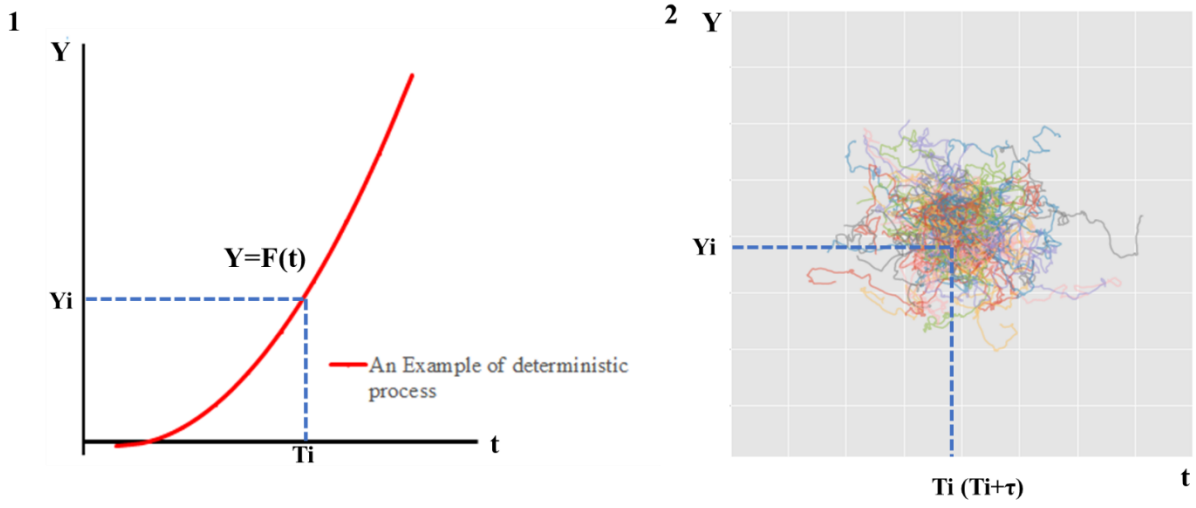


Figure 1-1) An Example of deterministic process, 1-2) An example of stochastic process

Langevin Equation

Consider a spherical particle with the radius "a" and mass "m" in a solvent fluid. Assuming the size of the particle is much larger than the size of fluid molecules, we can treat the fluid as a continuum medium with viscosity "η". "R(t)" is considered the temporal position of the particle at time "t" and "V" is its velocity. Based on Newton's equation of motion, if a body is moving relative to a fluid, it experiences a friction force (colored in blue in the figure 2), which will be proportional to the velocity with the constant of proportionality called the friction constant "ζ". For a spherical particle the friction constant is given by $\zeta=6\pi\eta a$, known as the Stokes law. In addition to the friction force, we know that there must be another type of force which gives rise to the irregular motion of the Brownian particle. We call this the random force F(t) (colored in red in figure 2), which represents the effects of the many collisions taking place between the spherical particle and the fluid molecules. Finally, by putting all of them together, we obtain an equation shown in Eq. (1). This equation is called the Langevin equation and it describes the Brownian motion of a particle diffusing in a fluid.

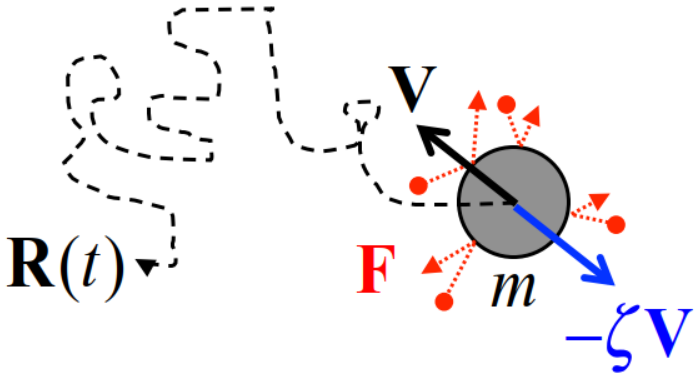


Figure 2) Schematic illustration of forces applied on a particle in a fluidic environment

$$m\frac{dV}{dt} = -\zeta V + F \qquad \text{Eq. 1.}$$

Assuming three-dimensional Cartesian coordinates, the random force has three components along the *x*, *y*, and *z* directions which are uncorrelated on the time scale of the Brownian particle. Einstein showed that the average (root mean square) distance *L*, a particle would move in time *t* (Eq. (2))

$$L = \sqrt{2Dt} \qquad \text{Eq. 2.}$$

Combining the Einstein relation obtained from the Langevin equation with Stokes law, we have a good estimate for the self-diffusion constant (Eq. (3)) of spherical particles of radius a, diffusing in a fluid of viscosity η at temperature T.

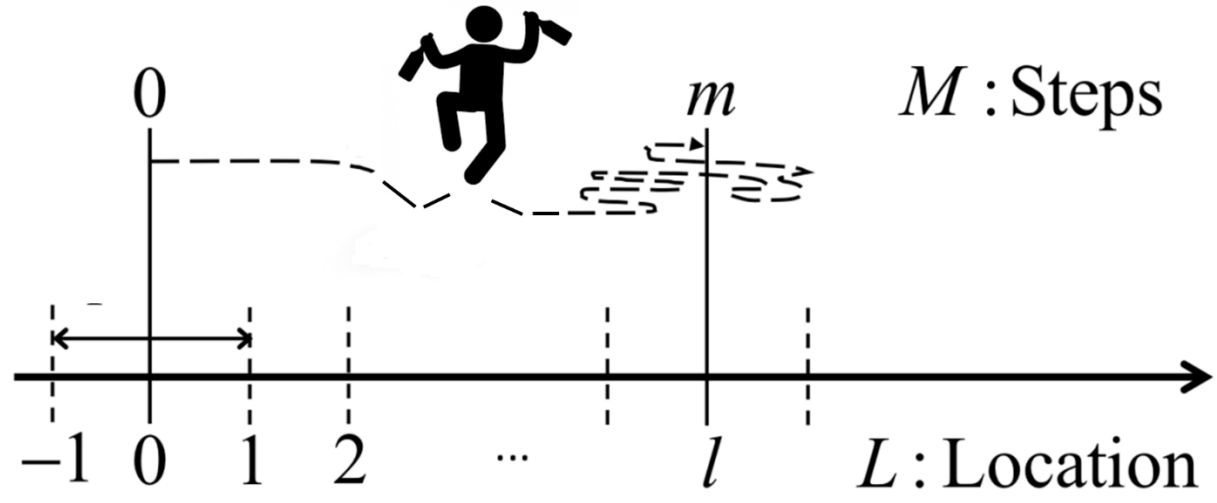
$$\boldsymbol{D = \frac{k_B T}{6\pi\eta a}} \qquad \text{Eq. 3}$$

In Eq. (3) k_B is Boltzmann constant, T is the absolute temperature, η is the viscosity and r is the radius of particles under consideration.

$k_B = 1.38064852 \times 10^{-23}\ m^2\ kg\ s^{-2}\ K^{-1}$

1D Random Walk

Imagine that a drunken woman is wandering around town after a heavy night of drinking. She is completely disoriented with no clue as to how to get home. At each step, she loses memory of which direction she was going, and chooses a new random direction in which to move. For simplicity, we will consider the 1D version of this random walk. Thus, at each step, the drunken woman has two options. She can either take a step to the left (assigned to -1) or to the right (assigned to +1). At each step she is basically flipping a coin to determine in which direction to move. After certain steps, the location of this woman is equal to the sum of all -1 and +1 from 0 to the last step.



In table 1, diffusion constants D of various particles and their corresponding diffusion time τ to spread over a typical microfluidic distance 100 µm can be seen.

Table 1. Diffusion constants of various particles and their corresponding diffusion time

Particle	Typical size (µm)	D ($\mu m^2.s^{-1}$)	τ (s)
Solute ion	10^{-4}	2×10^3	2.5
Small protein	5×10^{-3}	40	125
Virus	10^{-1}	2	2500
Bacterium	1	0.2	25000
Mammalian cell	10	0.02	250000

Questions:

- How long does it take hemoglobin to diffuse 1 cm in water? ($D=10^{-7}$ cm^2 s^{-1})
- How long to diffuse 10 µm?

1D Random Walk Modeling in Python

Step 1 (Defining libraries):

We can use *numpy* to generate random steps. The cumulative value of a series of random steps is a random walk. First we import *numpy* in order to use the random function and matplotlib.pyplot to do the plotting.

```
import numpy as np
import matplotlib.pyplot as plt
```

Step 2 (Defining random walk function):

Now we define a function to generate our random walks. The *numpy* function random.choice() generates a random sample from a 1D array. We can define an array [-1,0,1] for our steps and then chose randomly from these to generate an array of random steps of chosen length. Summing then gives us the walk. We can do the summing using the np.concatenate(start array,step array).cumsum() to start each walk from value in start array (normally zero) and then add the cumulative effect of each step along the array.

```
def randwalks(num_walks=1, num_steps=100):
    """Generate random walks.
        Defaults are 1 walk and 100 steps.
        Parameters
        ----------
```

```
        num_walks : float, optional
            The number of walks to generate (default is 1)

        num_steps : float, optional
            The number of steps in each walk (default is 100)
        """
    # possible steps
    step_set = [-1, 0, 1]
    # define starting point = 0
    origin = np.zeros((1,num_walks))
    # assign step matrix
    step_shape = (num_steps,num_walks)
    # generate step matrix
    steps = np.random.choice(a=step_set, size=step_shape)
    # generate walks
    path = np.concatenate([origin, steps]).cumsum(0)
   return path
```

Step 3 (Generating some walks with the function):

```
walks = randwalks(10000,10000)
```

Step 4 (Plot walks):

```
# plot 1D walks
for i in range(np.size(walks[0,:])):
    _ = plt.plot(range(np.size(walks[:,i])),walks[:,i])


# Add title and axis names
plt.title('1D random walks')
plt.xlabel('Number of steps taken')
plt.ylabel('Distance from start in 1D [steps]')
#plt.savefig('plots/multiple_random_walks_1d_1.png',dpi=600);
```

Step 5 (2D random walk):

Because the probabilities for a step in any direction (+-x, +-y, +-z) are independent, multidimensional random walks can be generated as combinations of 1D walks. We can add the start and stop positions to the plot too because it looks nice.

```
# Here is an example of a 2D walk
path2D = randwalks(2,10000) #generate an array with two 1D walks
# define the start and stop positions for plotting
start2D = path2D[:1]
stop2D = path2D[-1:]

# Plot the path
fig = plt.figure(figsize=(8,8),dpi=600)
ax = fig.add_subplot(111)
ax.scatter(path2D[:,0], path2D[:,1],c='blue',alpha=0.25,s=0.05);
ax.plot(path2D[:,0], path2D[:,1],c='green',alpha=0.5,lw=0.25,ls='-');
ax.plot(start2D[:,0], start2D[:,1],c='red', marker='+')
ax.plot(stop2D[:,0], stop2D[:,1],c='black', marker='o')
# Label everything
plt.xlabel('Distance from start in x direction [steps]')
plt.ylabel('Distance from start in y direction [steps]')
plt.title('2D Random Walk')
plt.tight_layout(pad=0)
#save the figure
#plt.savefig('plots/random_walk_2d.png',dpi=250);
```

Step 6 (Plotting all 2D random walks in a same figure):

```
fig = plt.figure(figsize=(8,8),dpi=600)
ax = fig.add_subplot(111)
for i in range(100):
    # Here is an example of a 2D walk
    path2D = randwalks(2,1000) #generate an array with two 1D walks
    # define the start and stop positions for plotting
    start2D = path2D[:1]
    stop2D = path2D[-1:]
    # Plot the path
    ax.scatter(path2D[:,0], path2D[:,1],c='blue',alpha=0.25,s=0.05);
```

```
    ax.plot(path2D[:,0], path2D[:,1],c='green',alpha=0.5,lw=0.25,ls='-');
    ax.plot(start2D[:,0], start2D[:,1],c='red', marker='+')
    ax.plot(stop2D[:,0], stop2D[:,1],c='black', marker='o')
# Label everything
plt.xlabel('Distance from start in x direction [steps]')
plt.ylabel('Distance from start in y direction [steps]')
plt.title('2D Random Walk')
plt.tight_layout(pad=0)
#save the figure
#plt.savefig('plots/random_walk_2d.png',dpi=250);
```

Step 7 (Analyzing Results):

Homework:

Run the code below on your python, explain the purpose of this simulation and its final results.

(Hint: Consider the equation 2, 3 and this equation: $D = \frac{1}{3T^2}\int_0^T dt\,\langle[R(t) - R(0)]^2\rangle$ which explains the relationship between mean square displacement and diffusion constant.)

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
plt.style.use('ggplot')
dim = 3 # system dimension (x,y,z)
nump = 1000 # number of independent Brownian particles to simulate
nums = 1024 # number of simulation steps
dt = 0.05 # set time increment, \Delta t
zeta = 1.0 # set friction constant, \zeta
m = 1.0 # set particle mass, m
kBT = 1.0 # set temperatute, k_B T
std = np.sqrt(2*kBT*zeta*dt) # calculate std for \Delta W via Eq.(F11)
np.random.seed(0) # initialize random number generator with a seed=0
R = np.zeros([nump,dim]) # array to store current positions and set initial condition Eq.
V = np.zeros([nump,dim]) # array to store current velocities and set initial condition Eq
W = np.zeros([nump,dim]) # array to store current random forces
Rs = np.zeros([nums,nump,dim]) # array to store positions at all steps
Vs = np.zeros([nums,nump,dim]) # array to store velocities at all steps
Ws = np.zeros([nums,nump,dim]) # array to store random forces at all steps
time = np.zeros([nums]) # an array to store time at all steps
```

```
for i in range(nums): # repeat the following operations from i=0 to nums-1
W = std*np.random.randn(nump,dim) # generate an array of random forces according to E
R, V = R + V*dt, V*(1-zeta/m*dt)+W/m # update R & V via Eqs.(F5)&(F9)
Rs[i]=R # accumulate particle positions at each step in an array Rs
Vs[i]=V # accumulate particle velocitys at each step in an array Vs
Ws[i]=W # accumulate random forces at each step in an array Ws
time[i]=i*dt # store time in each step in an array time
```

```
# mean square displacement vs time
msd = np.zeros([nums])
for i in range(nums): # loop over time steps
    for n in range(nump): # loop over particles
        msd[i]=msd[i]+np.linalg.norm(Rs[i,n,:])**2 # (R(t) - R(0))^2 = R(t)^2, since R(0)
    msd[i] = msd[i]/nump # average over particles
dmsd = np.trapz(msd, dx=dt)/ (3*(nums*dt)**2) # integrate using trapezoidal rule
print('D =',kBT/zeta,'(Theoretical)')
print('D =',dmsd,'(Simulation via MSD)')
fig, ax = plt.subplots(figsize=(7.5,7.5))
ax.set_xlabel(r"$t$", fontsize=20)
ax.set_ylabel(r"mean square displacement", fontsize=16)
ax.plot(time,6*kBT/zeta*time,'r',lw=6, label=r'$6D\,t={6k_BT \,t}/{\zeta}$'
ax.plot(time,msd,'b',lw=4, label=r'$\langle R^2(t)\rangle$')
ax.legend(fontsize=16,loc=4)
plt.show()
```

Experimental Set up

This experiment shows the diffusion of molecules in gelatin at different temperature and viscosity. The purpose of this experiment is to show the relationship between diffusion distance, diffusion time and diffusion constant.

Materials

- Gelatin Jello (with light background color)
- Food dye
- Sugar
- Petri dishes with covers
  (Other small flat containers can be used instead, e.g., lids, dishes or shallow bowls)
- Pasteur pipette
  (Round wooden toothpicks can be used instead to make a hole in the gel and add the dye to the gel)
- Timer
- Disposable gloves, Goggles

Preparation of Materials

1. Prepare the gelatin according to the instructions.

2. Take the petri dish of gelatin that has been sitting at room temperature for 5-10 minutes. Using a cut off pipette tip or a glass Pasteur pipette, make a small hole in the gel for dye. You can also use a round wooden toothpick to make a hole, but the hole will be smaller. The position of the hole in the plate is not important, but it should be far enough from the edge of the plate so diffusion of the spot can be easily observed.

3. Carefully pipette a small drop (2-10 µl) of dye solution into the hole using a Pipetman or Pasteur pipette. Try not to overfill the hole or you will be observing flow of the dye over the gel surface, rather than diffusion of the dye in the gel. You can also add the dye to the hole by soaking a round wooden toothpick in dye for a few minutes, then inserting the toothpick into a hole made with another toothpick. Make sure the dye has transferred into the gel before removing the toothpick.

4. Then cover the petri dish and measure the diameter of the dye spot at different times after adding the dye, e.g., 0, 5, 10, 30, 60, 150 minutes. The increase in size of the dye spot will be

easy to see even after 5 or 10 minutes. After measuring the diameter of the dye spot, divide by two to find the radius – the radius of the dye spot at each time is the distance the dye molecules have diffused from their starting position. Measure just to the edge of the dark dye spot which contains most of the dye molecules.

5. Photos can be taken at each time point using a cell phone. By including a ruler in the photo, you can accurately determine the diffusion distance from the photos – this is done by measuring the diameter of the dye spot in each photo and dividing by the length of one cm of the ruler in the photo to obtain the actual diameter of the dye spot; then divide the actual diameter by two to find the radius of the dye spot at each time point.

6. Warm the petri dish of gelatin by holding it between your hands for 3-4 minutes and repeat the dye diffusion experiment. Keep the petri dish warm between time points by holding it in your hands, keeping it horizontal.

7. Cool the petri dish of gelatin by placing it in a refrigerator or on ice and repeat the dye diffusion experiment. Keep the petri dish cold between time points and make sure that it is horizontal.

8. Measure diffusion time and diffusion distance for all the experiments and compare and discuss their value for:

- Warm and cold petri dishes

And Calculate diffusion coefficient for petri dish by plotting appropriate data and using Stokes-Einstein equation and discuss the effect of temperature.