Movement inside the cell

1 Assignment Description

These assignment sheets describe the modelling of two methods of transport in living cells. They provide a brief description of the topic and set problems some of which must be solved using python programs.

The assignment consists in writing an essay that contains an answer to each question and which expand on the material included in these notes. The references are there to provide you with extra source information and they are not exhaustive.

The essay must be readable on its own without reference to the assignment sheets. The essay must not necessarily follow the same order as the notes and does not need to answer the questions in the same order either. As a matter of fact using a different structure and order is a bonus. Do describe the interpretation or consequences of the results that you obtain. We also expect you to focus more on some aspects of the material presented in these notes. This could be a more detailed discussion of the derivation of the equations or algorithm described in the notes or more information about the structure of the cell to better explain the interpretation of the results. Don't hesitate to ask questions to your Biology friends if you have any. You can also solve problems which are not set and which answer some further questions that you might have.

Some part of the assignment sheet will need to be included in your essay, but do not copy these section verbatim, use your own words. The equations do not need to be changed, but you can provide more detailed explanation or derivation when appropriate.

You have to submit 3 files: 1 pdf file for the essay and 3 python programs. We advise you to typeset your essay in LaTeX but this is not compulsory.

The essay must have a maximum of 10 pages (using as a reference the provided LaTeX style file).

2 Introduction

Cells are the smallest living organisms and all large living creatures, like humans, are made of a huge number of cells. Cells are made out of a variety of elements, called organelles by biologists, each of which has very specific functions (see [2] and [1] for example). During its life cycle, the cell must transports some molecules within itself. For example proteins *manufactured* in the vicinity of the cell nucleus are needed in specific regions of the cell located elsewhere. The big question is how can large molecules like proteins or other larger complexes, be transported within the cell?

There are two main methods of transport. The first one is passive diffusion: molecules in the cell are constantly bombarded by other molecules and as a result are constantly being shaken somewhat like a ball in a pinball machine. These collisions are the result of temperature (temperature is actually a measure of the kinetic energy of free molecules) and result in a completely random displacement.

Cells can vary greatly in size, from a few micro-meters for the smallest to several hundred of micro-meters for the largest. Nerve cells can actually extend over several meters but they are an exception as they have very unusual shapes.

The other type of displacement is active transport: cells contains long molecular chains, polymers, forming what is called the cytoskeleton. These long interconnected molecules form long pathways on which some proteins, called motor proteins, can walk while pulling a cargo (see the video [4]). This type of transport still has some random component but is nevertheless more systematic.

The aim of this project is to study these two methods of transport and to determine which one is the most effective for different type of objects.

3 Diffusion

Diffusion is the results of the completely random collision between molecules. It can be described using various methods. The most analytical one is to consider a density $F(x, y, z, t)$ (which can be considered as a probability density for a single molecule or a concentration for a large number of identical molecules). It can be shown, see [3], [4], that for molecules which are randomly diffusing, $F(x, y, z, t)$ satisfies the equation

$$\frac{\partial F}{\partial t} = D\left(\frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} + \frac{\partial^2 F}{\partial z^2}\right). \tag{1}$$

To find solutions one uses the method of separation of variable to seeks solutions of the type $F(x, y, z, t) = f(x, t), g(y, t)h(z, t)$ and we find that

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

with similar equations for g and h.

A solution of (2) is given by

$$f(x, t) = \frac{1}{2\sqrt{\pi Dt}} e^{-x^2/(4Dt)} \qquad t > 0. \tag{3}$$

Notice that (1) is invariant under time and space translation meaning that if $F(x, y, z, t)$ is a solution $F(x + d_x, y + d_y, z + d_z, t + d_t)$ is also a solution for all d_x,d_y,d_z and d_t. This implies that the distribution can be centred anywhere.

Question 1:

- Check by direct substitution that (3) is a solution of (2).

Using a suitable trick, one can check that if $t > 0$,

$$N(t) = \int_{-\infty}^{\infty} f(x, t)dx = 1. \tag{4}$$

This shows that diffusion preserves the number of particles, as expected.

Question 2:

- It is obvious that (3) is singular for $t = 0$, yet, $\lim_{t\to 0} N(t) = 1$. Use this to justify that f can be used to describe the probability distribution of a single diffusing particle initially located at the origin $x = y = z = 0$.

The next question is to determine how long it takes, on average, to cover a distance x. First of all we can easily see that the average distance travelled by the particle is zero:

$$< x >= \int_{-\infty}^{\infty} xf(x, t)dx = 0. \tag{5}$$

So on average, a diffusing particle does not move, but this is only on average as it does move about. To evaluate a travelling time we must compute the average value of x^2:

$$< x^2 >= \int_{-\infty}^{\infty} x^2 f(x, t)dx \tag{6}$$

Question 3:

Show that

$$< x^2 > \quad = \quad 2Dt. \tag{7}$$

Hint define $\lambda = 1/(4DT)$ and rewrite (3) as a function of λ rather than Dt. Then use (4) to compute $dN(t)/d\lambda$.

The interpretation of (7) is that the typical distance covered during a diffusion time t is given by

$$d = \sqrt{< x^2 >} = \sqrt{2Dt}. \tag{8}$$

We thus see that the distance covered only increases like $\sqrt{t}$ and so for long displacement, diffusion is not effective at all.

To evaluate the time needed for molecules or organelles to diffuse inside de cell, we need to know how to compute D. This was derived by Einstein who used a result derived by Stokes to show that for a diffusing sphere of radius r

$$D \quad = \quad \frac{k_b T}{6\pi\eta r} \tag{9}$$

where $k_B = 1.38 \times 10^{-23} J/K$ is known as the Boltzmann constant, T is the temperature of the system in Kelvin and η the dynamic viscosity of the fluid. For water $\eta = 10^{-3} Pa\, s$ in the SI units. The inside of the cell is mostly water and the viscosity of the fluid, called cytosol, has been measured to be approximately 2 to 4 time that of water.

Question 4:

Some of the smallest objects in the cell are proteins. There are many of them and they are all relatively small. For example tubulin, the proteins of which microtubules are made, are not really spherical but can be approximated as a sphere of radius $3nm = 3 \times 10^{-9} m$. Compute D for that protein and how long it typically takes to travel a distance $\sqrt{< x^2 >} = 50\mu m = 50 \times 10^{-6} m$. (Use the temperature $T = 300K$ and η to be 2 times that of water). (Notice that in SI units, the units of D are m^2/s.)

Cells generate many tiny spheres made out of a lipid (oil) membrane inside which some signalling material (very specific proteins) are stored and shipped to various part of the cell. These vacuoles are really like parcels and are very important for cells to evolve and survive. Their size can vary and have a radius well over $100nm$.

Question 5:

How long does it take for a vacuole with a radius of $50nm = 5 \times 10^{-8} m$ to travel a distance of $50\mu m$ in the same condition as in the previous question?

Some of our neurones extends from our brain all the way to our toes.

Question 6:

How long does it take for a tubulin protein to freely diffuse from the very top of the cell in our brain to its very bottom in our toe assuming a total length of $2m$ and the same condition as in question 4? Convert the result in a familiar unit to understand how big it really is.

The diffusion equation is relatively easy to solve when D is constant and when there are no boundaries. In real situations, like in cells, the domain is bounded and D might also be a function of the position in the cell. Solving the diffusion equation analytically is then much harder and in most realistic cases impossible. Nothing is lost as one can solve it numerically.

Another weakness of the diffusion equation is that it only gives us a probability distribution of the diffusion of a particle and it does not really tell how a particle actually moves. A method one can use to study diffusion is to perform simulations and this happens to be quite simple to do. We will restrict ourselves to the diffusion in one dimension, but is it very easy to generalise it to 2 or 3 dimensions.

First we chose a time step dt which is the time scale over which we want to study the diffusion (we are not interested in the detailed movement for shorter time scales). We then remember that the typical distance dx travelled during the interval dt is given by $dx = \sqrt{2D\,dt}$. The simulation then proceeds as follows:

- We set the particle at its initial position, say $x = 0$.
- We chose $dt = 0.001$ (to make the motor step much smaller than the bin size of a histogram we will create later).
- We define $dx = \sqrt{2D\,dt}$.
- We take a uniformly distributed random number, rnd, in the interval $[0, 1]$.
- If $rnd > 0.5$ the particle moves to the right by dx: $x \to x + dx$.
- If $rnd < 0.5$ the particle moves to the left by dx: $x \to x - dx$.
- we increment the time : $t \to t + dt$.

We repeat step 2 to 5 as many times as needed.

The python implementation of the algorithm is given in the file `diffusion_1d.py`

```
import math
import random

# Set the parameter of the problem.
dt=0.001  # simulation time increment
D=0.5;    # diffusion coef
tmax = 100

def diffuse(t_max):
    """ Simulate the diffusion of a particle.
        Randomly move right or left by fixed step. Stop when t=t_max
        and return the position reached.
        Uses global variables D and dt
    """
    x = 0.0
    dr = math.sqrt(2*D*dt)
    # Evolve
    t =0;
    while (t < t_max): # diffuse until tmax
        rnd = random.random() # a random number between 0 and 1
        direction = 1  # default: move right
        if(rnd < 0.5):
          direction = -1 # move left (50% chance)
        x += direction*dr
        t += dt
    return(x)

# Solution
import numpy as np
import matplotlib.pyplot as plt

Nmax = 10000
Nbins = 50

x = []
##### TO DO ############
# execute the function diffuse(tmax) Nmax time and append the results
# in the list x
########################

# Produce a histogram of the collected data in x
```

```
42 # bins: the list of x values of the center of the histogram bins
43 n, bins, patches = plt.hist(x, Nbins, normed=1, facecolor='green', alpha=0.5)
44
45 ##### TO DO ############
46 # Plot the theoretical solution d(tmax) of the diffusion equation
47 # on top of the hyitogram produced above
48 # bins is a list of x value for the center of the bins
49 ########################
50
51 plt.xlabel('X')
52 plt.ylabel('Probability')
53 plt.title(r'$\mathrm{Histogram\ of\ Diffusion:}$')
54 plt.grid(True)
55
56 plt.show()
```

The function `diffuse(t_max)` performs the simulation from $t = 0$ until `t_max` and returns the position of particle at that time. By executing this function many times, we can determine how far the particle has travelled within the chosen time and compare the histogram with the solution of the diffusion equation.

Notice that to generate a random number, we use the python function `random.random()`, from the module `random`, which return a uniformly distributed random number in the interval $[0, 1]$. The number returned will be different each time (try it) as well as each time you run your program.

Coding task 1:

Modify the file `diffusion_1d.py` to make it a fully working program and proceed as follows:

- Below line `30`, add a loop to execute the program `Nmax` times and add the result returned by `diffuse(t_max)` to the list `x`.
- Notice that line `36` produce a histogram of the value stored in the list `x`. It returns 3 values, but the only one we are interested in is the list returned in `bins` which is the `x` coordinates of the centre of the bins used for the histogram. The histogram is normalised so that it can be interpreted as a distribution.
- Below line `42`, add some code to compute the values of the theoretical solution (2) where you must take $t = t_{max}$ and use those values to generate a plot of the (2), as a red line, on the same figure as the histogram, performed in line `36` (just add the figure to plt). Use the values returned in `bins` to evaluate (2).
- Program output: a histogram of the generated data (in green) and on the same figure the theoretical curve for the distribution (in red).

Question 7:

Run the program `diffusion_1d.py` and save the figure it generates. How good is the simulation compared to the theoretical solution?

We have used the solution of the diffusion equation to compute the typical time it takes for a particle to cover a distance x, but what is the average time it takes a particle to travel such a distance?

Coding task 2:

To answer that question, make a copy of the program `diffusion_1d.py` and call it `diffusion_1d_avt.py` and modify it so that it computes the average time needed to travel a specified distance.

- Modify the function diffuse so that it takes `x_max` as an argument and so that the loop stops when `x > abs(x_max)`. It must return the time `t` when the particle has reached the distance `x_max`.

- The code below the definition of `diffuse` must execute the function `diffuse` 10000 times to compute the average time `<t>`.
- Include the loop that calculate the average time in another 2 loops to compute the average travel time for `D = 0.5`, `1` and `2` as well as `x_max=1.`, `2.`, `3.`, `4.` and `5.`
- The program must output the following values:

  ```
  x_max <t> D t_theory
  ```

  for each value of `D` and `x_max` (15 lines of output in total). `t_theory` is the expression which you believe to be the theoretical value for `<t>` (you can run your program without it the first time to guess what this might be).
- Program output: a title line followed by the 15 lines of data described above.

Question 8:

Create a table with the data generated by `diffusion_1d_avt.py`, from coding task 2, and explain what the expression for the average travelling time $< t >$ is.

Question 9:

When we considered the diffusion of a protein in a neurone, we assumed that it was diffusing on a infinite line when, in reality, it diffuses inside very thin cylinder, the neurone, starting at one end and ultimately reaching the other extremity. Is the diffusion in a finite domain different than in an infinite domain? Is the answer we obtained in question 6 correct or does it need to be corrected for a finite box?

4 Motor

Diffusion is not a very effective transport method for long distances. Nature has thus designed an active transport method which is faster. Watch the video of ref [5] to obtain a visual description of motors as well as some information on various biological concepts.

Motor proteins exhibit a random dynamics and are fuelled by chemical energy. Once they are attached to their cargo and their walkway (microtubules or actin filaments), they walk one step at a time but can also detach from the walkway or the cargo. The time between steps is random and distributed according to a Boltzmann (Poisson) distribution characterised by a rate which we will call $k_+(F_D)$. The detachment is also random and given by a Boltzmann distribution and rate k_d. We should also point out that the 2 events can be considered independent of each other.

4.1 A simple model

One can simulate the movement of a motor protein using a so call Monte Carlo simulation:

- One computes the time the next step will take place using

$$t_{step} = \log(rnd)/k_+(F_D) \tag{10}$$

  where rnd is a random number uniformly distributed in the interval $[0, 1]$.
- One computes the time the next detachment will take place

$$t_d = \log(rnd)/k_d \tag{11}$$

  where rnd is another random number uniformly distributed in the interval $[0, 1]$.
- If $t_{step} < t_d$, the motor moves by one step and the time variable `t` is increased by t_{step}.
- If $t_d < t_{step}$, the motor detaches and the time variable `t` is increased by t_{step}.

- One then repeats the procedure above until motor detaches.

It can be shown, but this is beyond the scope of this course, that the algorithm above simulates events described by Poisson distributions. The algorithm is implemented by the function `evolve` in the program `motors.py`.

When a motor pulls a cargo, which we will assume to be spherical, the cargo is subjected to resistance forces due to the fluid in which they are immersed. Stokes showed that the force is given by

$$F_D = 6\pi\eta r v \tag{12}$$

where one recognises, partly, the denominator of eq (9), so η is the viscosity of the cytosol (fluid inside the cell), r the radius of the cargo and v the speed at which the cargo moves in the cytosol.

As one would expect the rate $k_+(F_D)$ decreases when F_D increases and the detachment rate increases with F_D. This was studied in detail by Kunwar et al. [6] who found

$$k_+(F) = \frac{v}{d}\left(1 - \left(\frac{F}{F_{stal}}\right)^{1/2}\right) \tag{13}$$

and

$$k_d = \{ \begin{array}{ll} k_{d0}e^{F/F_d} & (F \leq F_{stall}) \\ \frac{k_{d0}}{0.254(1-e^{-F/\lambda_F})} & (F \geq F_{stall}) \end{array} \tag{14}$$

where

- $k_{d0} \approx 1/s$: detachment rate under zero load
- $F_{stall} = 1.25pN = 1.25 \times 10^{-12}N$: stall force for dynein
- $F_d \approx 0.87pN = 8.7 \times 10^{-13}N$: detachment force
- $\lambda_F \approx 1.97pN = 1.97 \times 10^{-12}N$
- $d = 8nm = 8 \times 10^{-9}m$: motor step
- $v_M = 0.85\mu m/s = 8.5 \times 10^{-7}m/s$: average motor speed (dynein)
- $k_BT = 4.14^{-23}J$: at a temperature of $300K$.

Coding task 3:

The program `motors.py` simulates the movement of a motor, but a few parts must be completed:

- Complete the definition of the function `F_load(self,R)` so that it corresponds to the Stokes force (12), using the parameter already defined in the function `__init__` of the class. Use the value v_M in the list above for v.
- Complete the definition of the function `kp` so that it corresponds to $k_+(F)$ in eq (13).
- Complete the definition of the function `kd` so that it corresponds to k_d in eq (14).
- Complete the definition of the function `average_attached_time`
- In the code below the class definition, use the function `average_attached_time` to compute the average detachment time for a cargo of radius $1 \times 10^{-7}m$.
- Add some code to generate the following 2 figures
  - The average detachment time t as a function of the cargo radius r.
  - The average displacement, x, before detachment as a function of the cargo radius r.

Scan the radius from $0.1\mu m = 10^{-7}m$ to $60\mu = 6 \times 10^{-5}m$ using circa 200 data points.

Program output: 3 figures one after another:

- figure 1 : a typical trajectory of a motor (lasting at least until $1 = 1$)
- figure 2 : Average detachment time as a function of cargo radius.
- figure 3 : Average displacement as a function of cargo radius.

Question 10:

Run the program `motors.py` and save the 3 figures it generates:

- The first figure represents the typical trajectory of a diffusing particle or radius $100nm = 1 \times 10^{-7}m$ and is created by the function call `mot1.evolve(r,0.01,True)`. Run it a few times and chose one when the motor remains attached for at least 1s.
- The other two figures are the one you have to encode yourself for coding task 3.

Question 11:

When a motor detaches from the microtubule or the actin filament, it usually re-attach itself rapidly or another motor pulls the cargo along the same track. If we assume that the re-attachment is instantaneous after detachment, what is the critical distance d_{crit} for which the average speed of motor transport is the same as the average speed due to diffusion for a vacuole of radius $50nm = 5 \times 10^{-8}m$?. Should vacuoles of $50nm$ radius be diffusing or should they rather be transported by motors?

4 References

[1] T.D. Pollard, W.C. Earnshaw, J. Lippincott-Schwartz, *Cell Biology*, ISBN : 9781416022558

[2] http://www.biologymad.com/resources/Ch%201%20-%20Cells.pdf

[3] http://web.mit.edu/biophysics/sbio/PDFs/L15_notes.pdf

[4] Some derivation of the diffusion equation: http://mathbench.umd.edu/modules/cell-processes_diffusion/page09.htm, https://en.wikipedia.org/wiki/Fick's_laws_of_diffusion

[5] A movie about transport in the cell: http://www.ibiology.org/ibioseminars/cell-biology/ron-vale-part-1.html

[6] A. Kunwar *et al. Mechanical stochastic tug-of-war models cannot explain bidirectional lipid-droplet transport.* PNAS (2011) 108, 18960-18965 doi:10.1073/pnas.1107841108

5 To submit:

- One pdf file for the essay, including the figures for questions 7 and 10.
- Python code 1: `diffusion_1d.py`
- Python code 2: `diffusion_1d_avt.py`
- Python code 3: `motors.py`

