LAB 1: DIFFUSION SIMULATION WITH PYTHON

Lab Goals

- Install and run a Jupyter notebook
- Learn Python commands for lists, plots, and importing data
- Simulate a simple diffusion process

This is a 1 week lab. There are no partners for this lab.
Upload your Jupyter notebook to BrightSpace as a PDF by the end of your lab day.

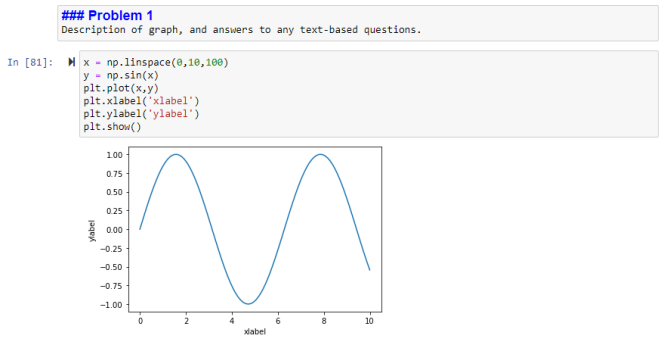
AS YOU LEARN PYTHON, YOU WILL GET MANY ERRORS AND HAVE TO DEBUG YOUR CODE.
TALK TO EACH OTHER AND ASK FOR HELP!!!

PRELAB

Instead of a prelab quiz, follow the instructions in "Jupyter installation.pdf" in BrightSpace to install Jupyter and upload a notebook to BrightSpace. This is also due at the end of your lab day. Having Jupyter on your own computer will allow you to finish the labs outside of class. Jupyter notebook will also be used for some of your homework assignments if you are also taking CHM374 lecture.

JUPYTER AS YOU LAB NOTEBOOK

You will be using the Jupyter notebook as your lab notebook. Items that require a graph or text answer are in bold in the lab and labeled by a problem number. In your Jupyter notebook, change a cell from a code cell to a text cell by changing it to "Mark Down." Label each problem a text cell containing the header "### Problem Number", for example



SUBMITTING YOUR JUPYTER LAB NOTEBOOK AS A PDF

Select File/ Print Preview. The notebook will open the preview in a separate tab as an html document. Print the preview and select to Save as PDF. Upload the PDF to BrightSpace.

The lab must be submitted to BrightSpace by the end of the day. You are welcome to send the Jupyter notebook to yourself and finish at home.

JUPYTER AND PYTHON TUTORIAL

This lab is an introduction to both Python and Jupyter notebooks and assumes no previous coding experience. Python is a free and open-source programming language, which is widely used in research and industry. The Jupyter notebook is a web-based notebook that can run Python code. This semester, we will use Python code within a Jupyter notebooks for plotting, fitting, and data analysis. The Jupyter notebook runs Python code, but will also serve as your lab notebook.

Open a new Jupyter browser tab by opening Anaconda Navigator and selecting Jupyter notebook. Find the "Python Tutorial.ipynb" notebook on the Desktop in your section's folder and open it to see the tutorial. The lab will refer to sections in the tutorial. If you are new to Python, it might be useful to read through it once to get familiar with the key concepts.

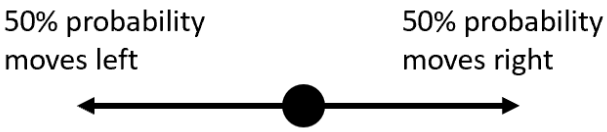
LAB: DIFFUSION AND THE RANDOM WALK MODEL

DIFFUSION INTRODUCTION

Diffusion

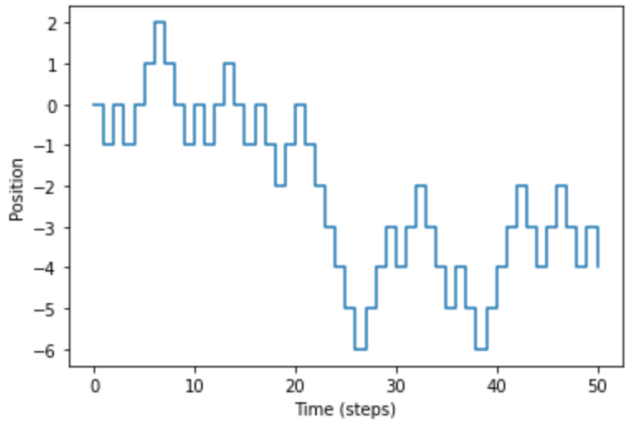
Diffusion is the movement of anything, such as an atom or molecule, from a region of higher concentration to a region of lower concentration. In this lab, we are going to explore a simple model of diffusion called a random walk and simulate the spread of the diffusing particles as a function of time.

In the random walk model for diffusion, a particle moves in a random direction at every time increment. For example, a 1D particle can move either 1 step left or right at each time increment.



1 Step in Random Walk

Over time, the particle will drift away from its starting position. Here is a 1D random walk simulation with 50 steps.



Prelab = 5 points for installing Jupyter.

TOTAL IS 50 POINTS
SINCE 1 WEEK LAB

Over time, the particle will diffuse away from the starting position. In this lab, we will simulate many random walk processes. We will try to answer: How does spread of the particle vary with the total number of steps?

2 points

PROBLEM 1: INSERT A TEXT CELL

a) **Read Section 2 of the Tutorial.**
b) **Open a new Jupyter notebook in your section's folder on the desktop and call it "LastName_Lab1"**
c) **Click on first cell, change it to MarkDown, and type a problem header. Try running the cell with Ctrl-Enter to display the formatted text.**

```
### Problem 1
Description of graph, and answers to any text-based questions.
```

d) **In the text cell, make a hypothesis on how far the particles will move on average from the center as a function of the number of steps. Some options: no average movement, linear, quadratic, square root, exponential.**

2 points

PROBLEM 2: CALCULATIONS AND PLOTTING

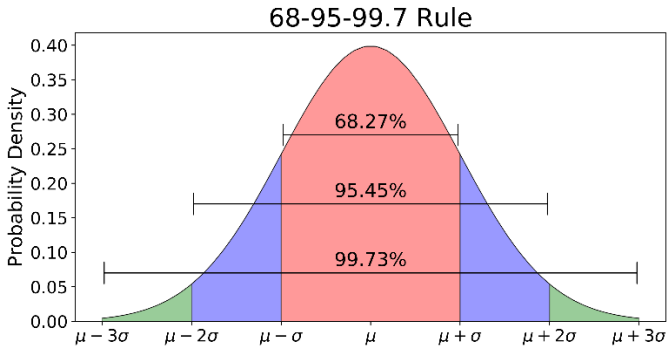
a) **Read Section 3 and 4 of the tutorial about how to perform math calculations. Make a text cell with "# Problem 2", then start a new cell by double clicking below the first cell. Calculate and print $\exp(-1/2)$. Calculate and print $1/\sqrt{\pi}$.**

b) **Read Section 5 on lists. Start a new cell. Define a variable x which is a linearly spaced array from -4 to 4 with 100 points and print the list.**

c) **Read Section 6 on plotting. In the same cell and using the list x from the last part, plot a quadratic curve, $y = x^2$. Label your axes.**

PROBLEM 3: PLOTTING A NORMAL DISTRIBUTION

THE NORMAL DISTRIBUTION

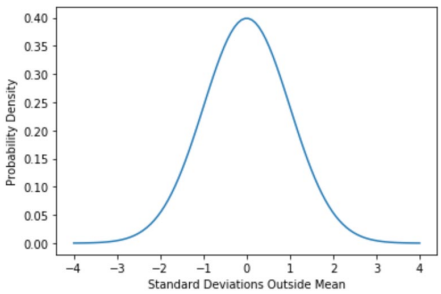


Before simulating the random walk model, we will review the normal distribution. The normal distribution is ubiquitous in the natural sciences. In fact, in probabilty theory the central limit theorem states that the average of many samples taken from any probability distribution is itself a random variable whose distribution converges to a normal distrubtion for a large number of samples. In the random walk model, the particle can at every step move either left or right. After many steps, we can therefore expect the distribution of the particle to follow a normal distribution.

A normal distribution has a mean μ and the width is given by the standard deviation σ. The probability of getting a value within 1 standard deviation from the mean is 68%. The functional form for a normal distribution is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

a) Start a new problem with labeled with a text cell, and make a new cell for code. Plot the normal distribution f(x) from above with a mean $\mu = 0$ and standard deviation $\sigma = 1$. Label both axes. Note that since this is a complicated function, you will likely get an error the first time. Check your parentheses! It should look like:



We are going to use a random number generator to make a list of values from this normal distribution and plot them in a histogram.

b) Use the function x = np.random.normal(mu, sigma, N) **to make a list of 50 random numbers from the normal distribution with mean 0 and $\sigma = 1$ and print the results.**
The root mean square (RMS) is a way to estimate the spread of values. For a list x, the RMS is given by:

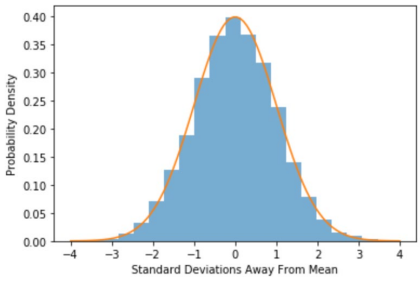
$$RMS = \sqrt{\frac{1}{n}\sum_i x_i^2}$$

In Python, this is given by xrms = *np.sqrt(np.mean(x**2))*

c) In the same cell, calculate the RMS of the random numbers. How does it compare to the standard deviation? For a normal distribution, the RMS should equal the standard deviation.

d) We are now going to make a histogram of these random numbers to see how they are distributed. Start a new cell. Make a list of N=5000 random normal numbers (do no print the list because it would be huge!). Plot the results in a histogram. There is an example of a histogram and random number generator in Section 6 of the tutorial. It should look like:

10 points total.
(5 for normal distribution, 5 points for histogram)



PROBLEM 4: SIMULATING A RANDOM WALK

We will now simulate a random walk for a particle in 1D. At each time step, the particles randomly moves either +1 or -1. We will see how far on average the particle travels from the starting position after many steps. Here we will define a function random_walk(N) that generates a random walk with N steps. Read Section 7 of the tutorial on how to define and call a function.

a) Start a new problem text cell, and start a code cell. Type the following function for a random walk into your notebook and run it for N=20 steps. Print the results.

```
def random_walk(N):

    # make a list of N random values of 1,-1
    random_steps = np.random.choice([1,-1], N) # discrete steps

    # add the random steps one-by-one to get a random walk
    positions = np.cumsum(random_steps)  # absolute position
    positions = np.insert(positions, 0, 0.0) # insert the initial position

    # Print the final position of the random walk
    print( positions[-1] ) # print the final position

    # Return the entire random walk list
    return positions

# Call the function
N=100
steps = random_walk(N) # prints the final position and sets steps equal to the list of all positions
print(steps)
```

b) Start a new cell. Note that even though you defined random_walk() in the last cell, it is still defined for the new cell. Run another simulation with N=50 steps and plot the results. Label your axes. Hint: if you call plt.plot(steps) with only one argument, it will use x = [0,1,2,3,…, 49] as the x-axis. Or you can make x = [0,1,2,3,…, 49] using x = np.arange(0,50).

10 points for correct plot.

c) Plot 3 different simulations on the same plot. Note that you can just copy and paste your code 3 times. Or you can also use a for loop, described in Section 10, to run the code many times (try 100).

5 points

d) Start a new cell. Now you will make a histogram of the final positions of at least 20 experiments with N=50 steps. Calculate the RMS value of the list to estimate how much the particles have spread out.

Hint: When you run the simulation steps=random_walk(N), it also prints the final position. You can also get the final position as a variable with final_position = steps[-1]. You can run the simulation 50 times manually and put the final positions in an array with final_position_list = np.array([-10, 5, …]). If you are comfortable with for loops, you can quickly make the list with a for loop and run 10,000 simulations (see second example in Section 10 of the tutorial).

In the last part, you calculated the RMS spread after N=50 steps. You now could run this simulation for many different values of total steps N, and plot the RMS spread vs N. To save time, I have simulated it for you and have put the results in a csv file on your desktop called *random_walk.csv*. Read Section 8 in the tutorial on how to import data from a csv file. You should have a file "random_walk.csv" from BrightSpace or your sections folder on the lab desktop.

e) Import CSV Data from "random_walk.csv," and plot the data. Label the x-axis 'Steps Taken' and y-axis 'RMS Position'. Hint: you will need to change "x" and "y" to the csv file column names.

10 points for plot

f) The root mean square (RMS) of a normal distribution is the standard devation σ. The square of the standard deviation is called variance $V = \sigma^2$. Plot the variance as a function of steps taken. What functional form does this look like? This slope is called the Diffusion Coefficient.

POSTLAB

PROBLEM 5: POSTLAB QUESTIONS

Start a new text cell with a header "# Postlab Questions" and answer the following questions.

A fundamental property of many diffusion processes is that the variance scales linearly with time, V $V = D\,t$.

2 points

a) Was your prediction from Question 1 correct? How does the width or standard deviation of the particles scale with time?

For many diffusion processes, the diffusion constant can be estimated by $D = \frac{h^2}{\Delta t}$, where h is the step size and Δt is the time between steps.

2 points

b) How well does the diffusion estimate work for the random walk simulation we have performed?

2 points

c) One type of diffusion process is a small particle randomly moving around in a liquid. Do you think the random walk model is a good model for this process? Why or why not? This type of random motion is called Brownian motion. Albert Einstein's first important paper modeled this motion and used it to help prove the existence of atoms.