Ryan Herrmann

Lab 5

BMED 430

Introduction

The purpose of this lab was to use the built in python functions that added on from the last lab to then look at the standard deviation of the output concentration when the initial conditions were changed. The lab used a variance and multiplied it by an initial condition to get the initial uncertainty and then ran the same functions as before to get the deviation.

Numerical Methods

The numerical methods were to use the governing equation of $\frac{\partial \rho_{A_f}}{\partial t} = \frac{j_{A_F} A_s N_p}{V_f}$ with the initial concentration being 0 and the time being 0. That allows us to derive using scipy and numpy to apply the governing equation along with the inputs to display mass concentration and concentration over 10 minutes with 100 different samples for each test. That will give a final value for concentration. The equations that were used with the governing equation were $j_{A_f} = k_m(\rho_{A_s} - \rho_{A_f})$ and solving for km with $k_m = \frac{2D_0}{D_{aimp}}$. D0 is found using $D_0 = \frac{k_b T}{6\pi\mu a}$. With a being the radius of the molecule solved for using $a = \left(\frac{3MW}{4\pi\rho_{drug} A_v}\right)^{\frac{1}{3}}$ in cm. The standard deviation of the final values of each of the 100 samples after 10 minutes were calculated using numpy.

Pseudo Code

- Import required packages
- Define constants and input data
    - Fluid volume
    - Number of particles
    - Particle diameter
    - Temperature
    - Fluid viscosity
    - Molecular weight of the drug
    - Drug concentration available at each particle surface
    - Initial concentration
    - Density of the drug
    - Mass of the drug
    - Avogadro's constant
    - Boltzmann's constant
    - Number of samples
    - Uncertainty
- Use the definitions to calculate the needed values of a, km, D0 and concentration (jaf)
- Loop through each time step to give a concentration over each second and for each sample and calculate final concentration, and take the final concentration for each sample and put into a list

- Plot the final concentrations for each sample
- Either through a nested loop or hard code set the initial values for fluid volume or density to 0 and redo the previous loop for each combination
    - Volf = 0
    - Rho = 0
    - Neither = 0
    - Both = 0 (there should be no variablilty)
- Display graphs and write to table
- Export table to csv

Output

The plot of different concentrations for each sample for each combination is shown in Figure 1.

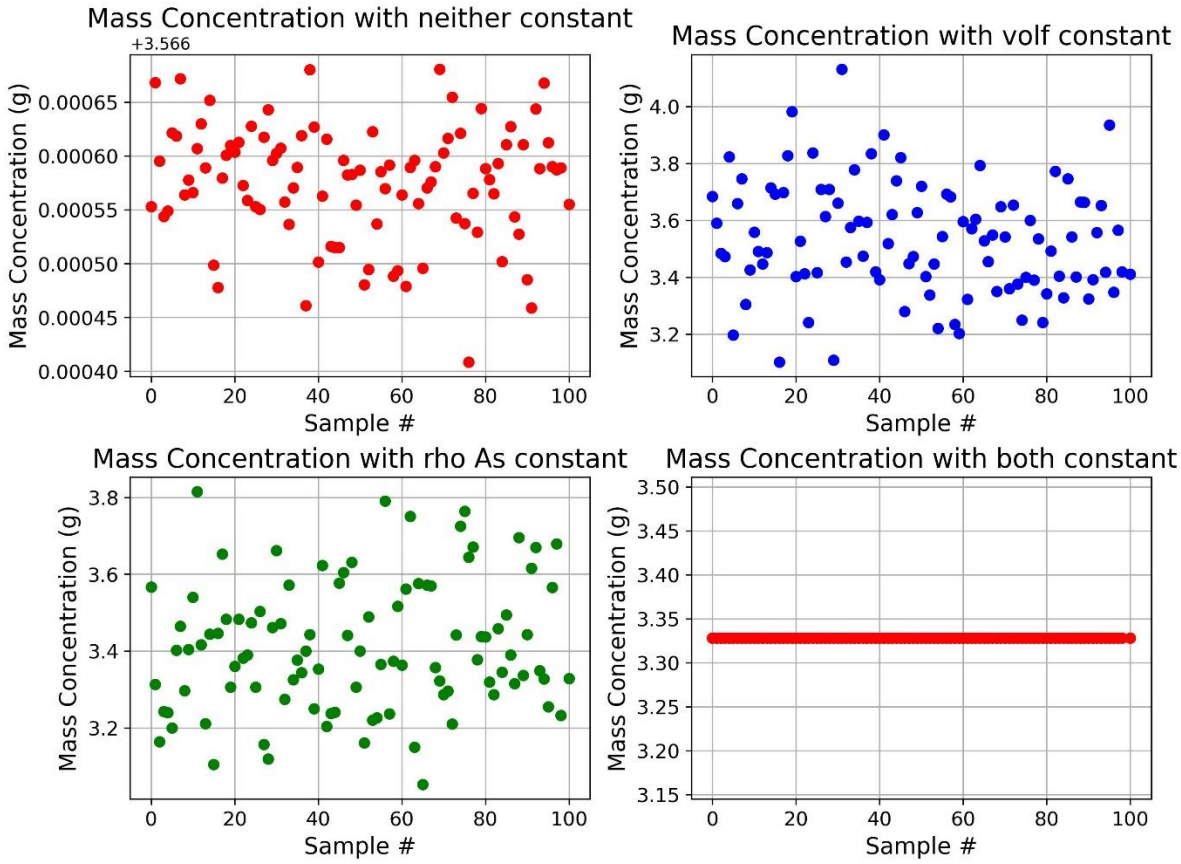

Figure 1: Final mass concentrations for each combination of initial conditions: Noted that the variability is not very understandable except for in a table but also that when both are left constant = 0, there is no variability as expected.

In Table 1, the combinations and the final concentration as well as standard deviations are shown.

Table 1: Combinations of initial conditions, the uncertainty and standard deviation: The combinations are shown with what was held constant. The results were also the mass concentrations.

Held Constant	Samples	Rho As (+/-)	Volf (+/-)	Result (g)	Stdev (+/-)
N/A	100	0	5000	3.567	5.28E-05
Volf	100	0.0025	0	3.533	0.1945
Rho As	100	0.002391	4961	3.411	0.1673
Both	100	0	0	3.328	0

Discussion

Python, although slowed down when 100 samples were run for 10 minutes was able to calculate the values as well as the standard deviation. From the results, when held constant, there is no variability in the concentration, because just like lab 1, if the are ran over and over with the exact same conditions, the answer will be the same. For the other, when neither was held constant the variability in concentration was also very low, and in doing so, it means that for a real life test varying both would yield a more accurate result than just holding one constant. Between if you had to hold one constant, the variability is about the same.

Appendix

```
#Import for python
import pandas
import numpy as np
import scipy.constants
import math
import matplotlib.pyplot as plt

#Basic Formatting
sigfigs = 4
N_samples = 100

#Import table
vol = 100 #liters
Num_part = 22736
p_diam = 0.2 #cm
p_rad = p_diam/2
temp = 300 #K
vis = 0.852 #cP
molec_weight = 600 #g/mol
rho_drug = 1 #g/cc
rhoA_f = 0 #initial concentration
massA_f = rhoA_f*vol #g
```

```
unc = 0.05

#Timing for the loop for the interations and methods etc
timei = 0 #initial time
timef = 10 #min
timef = timef*60 #seconds
dt = 0.05 #seconds

#Scipy constants
Kb = scipy.constants.Boltzmann #Boltzman constant
Av = scipy.constants.Avogadro

#Separate Constants

#conversion factors
#convert cp to pa*s
cP_Pas = 1e-3
vis = vis*cP_Pas
#print(rhoAs_a)

p_surf = 4*np.pi*p_rad**2

#Definitions of variables given the input
def a_sol (MW):
    return ((3*MW)/(4*scipy.constants.pi*Av))**(1/3)

def Jaf_funct (K_m,Paf,Pas):
    return K_m*(Pas - Paf)

def Km_funct(D0, Diamp):
    return 2*D0/Diamp

def D0_funct(TempD,viscos,a):
    x = (Kb*TempD)/(6*scipy.constants.pi*viscos*a)
    return x

#using the function to make as
As = a_sol(molec_weight)/100

#moving that to D0
D0 = D0_funct(temp, vis, As)
D0 = D0*100**2

#Km calculation
Km = Km_funct(D0,p_diam)
```

```
vol = vol*1000 #switch to mL
rhoA_s = 0.05 #g/cm^3

#lists for results
stDev_List = []
stDev_List_M= []
concMean_list = []
concMean_listM = []

stDev_ListF = []
stDev_List_MF= []
concMean_listF = []
concMean_listMF = []

#lists for number of samples
num_samples = []

#lists of +/- rho and vol these will be formatted
rhoA_uncertainty = []
volf_uncertainty = []

#all values for the plot with number of samples as the first column
whole_list = []
plot_samples = np.arange(0,N_samples, dtype=int)
whole_list.append(plot_samples)

#hardcoded list
end_Status = 4
j=0

#poor use of a while loop when a for loop should have been used but it works the
same.
#this is to make the differnt starts. Graphs will be just based on samples
while j<end_Status:

    num_samples.append(N_samples)

    unc_volf = unc*vol # +/- g/cc
    unc_rhoA_S = unc*rhoA_s # +/- g/cc

    #change via cases
    #holds rhoAs constant
    if j==0:
        unc_rhoA_S = 0
```

```
    #holds volf constant
    elif j==1:
        unc_volf = 0
    #holds both constant
    elif j==3:
        unc_volf = 0
        unc_rhoA_S = 0

    rhoA_uncertainty.append('%.*g' % (sigfigs,unc_rhoA_S))
    volf_uncertainty.append('%.*g' % (sigfigs,unc_volf))

    volf_a = np.random.normal(vol, unc_volf, N_samples)
    rhoAs_a = np.random.normal(rhoA_s, unc_rhoA_S, N_samples)

    L_conc = []
    L_conct = []
    L_mass = []
    L_masst = []

    for i in range(N_samples):
        rhoA_f = 0 #initial concentration
        massA_f = rhoA_f*vol #g
        #set up loop
        icount = 0
        time = timei

        while time<timef:

            rhoA_s = rhoAs_a[i]
            J_af = Jaf_funct(Km,rhoA_f,rhoA_s)
            vol = volf_a[i]

            drhoA = J_af*p_surf*Num_part/vol
            rhoA_n = rhoA_f + drhoA*dt

            rhoA_f = rhoA_n
            massA_f = rhoA_f * vol

            icount += 1
            time = icount*dt
        # print(f'concentration after ten minutes: {rhoA_f:.4g} g/cc')
        # print(f'concentration after ten minutes: {massA_f:.4g} g')
        L_conc.append(rhoA_f)
        L_conct.append('%.*g' % (sigfigs,rhoA_f))
        L_mass.append(massA_f)
```

```
        L_masst.append('%.*g' % (sigfigs,massA_f))

    whole_list.append(L_mass)

    #print(L_masst)
    mean_conc = np.mean(L_conc)
    stdev_conc = np.std(L_conc)
    # print(mean_conc)
    # print(stdev_conc)

    stDev_List.append(stdev_conc)
    concMean_list.append(mean_conc)

    #formatted lists for the table
    stDev_ListF.append('%.*g' % (sigfigs,stdev_conc))
    concMean_listF.append('%.*g' % (sigfigs,mean_conc))

    mean_mass_conc = np.mean(L_mass)
    stdev_mass_conc = np.std(L_mass)

    stDev_List_M.append(stdev_mass_conc)
    concMean_listM.append(mean_mass_conc)

    #formatted list for table
    stDev_List_MF.append('%.*g' % (sigfigs,stdev_mass_conc))
    concMean_listMF.append('%.*g' % (sigfigs,mean_mass_conc))

    j+=1

# print(stDev_List_MF)
# print(concMean_listMF)
# print(volf_uncertainty)
# print(rhoA_uncertainty)
# print(num_samples)

#finally hard print a list to lable
listNames = ["N/A","Volf","Rho As","Both"]

#create dictionary
results = {"Held Constant":listNames,"Samples":num_samples,"Rho As (+/-
)":rhoA_uncertainty,"Volf (+/-)":volf_uncertainty,"Result
(g)":concMean_listMF,"Stdev (+/-)":stDev_List_MF}

#create df
df1 = pandas.DataFrame(results)
```

```
df1.set_index("Held Constant",inplace = True)
print(df1)
print("")
#print(whole_list)

fig, axes = plt.subplots(2, 2, figsize=(10, 8))

axes[0, 0].plot(whole_list[0], whole_list[1], 'ro')
axes[0, 0].set_title("Mass Concentration with neither constant", fontsize=16)
axes[0, 0].set_xlabel("Sample #", fontsize=14)
axes[0, 0].set_ylabel("Mass Concentration (g)", fontsize=14)
axes[0, 0].tick_params(labelsize=12)
axes[0, 0].grid(True)

axes[0, 1].plot(whole_list[0], whole_list[2], 'bo')
axes[0, 1].set_title("Mass Concentration with volf constant", fontsize=16)
axes[0, 1].set_xlabel("Sample #", fontsize=14)
axes[0, 1].set_ylabel("Mass Concentration (g)", fontsize=14)
axes[0, 1].tick_params(labelsize=12)
axes[0, 1].grid(True)

axes[1, 0].plot(whole_list[0], whole_list[3], 'go')
axes[1, 0].set_title("Mass Concentration with rho As constant", fontsize=16)
axes[1, 0].set_xlabel("Sample #", fontsize=14)
axes[1, 0].set_ylabel("Mass Concentration (g)", fontsize=14)
axes[1, 0].tick_params(labelsize=12)
axes[1, 0].grid(True)

axes[1, 1].plot(whole_list[0], whole_list[4], 'ro')
axes[1, 1].set_title("Mass Concentration with both constant", fontsize=16)
axes[1, 1].set_xlabel("Sample #", fontsize=14)
axes[1, 1].set_ylabel("Mass Concentration (g)", fontsize=14)
axes[1, 1].tick_params(labelsize=12)
axes[1, 1].grid(True)

plt.tight_layout()
plt.show()

fig.savefig('Monte_Carlo/MonteCarloGraphsNotIn.jpeg',dpi = 300,bbox_inches =
'tight')

L_dfs = [df1]
with open('Monte_Carlo/MCResultsTableNotIn.csv','w',newline='') as f:
    for df in L_dfs:
        df.to_csv(f)
```

```
        f.write("\n")
```