Ryan Herrmann

Lab 4

BMED 430

Introduction

The purpose of this lab was to use python to apply Euler's method for an initial value problem. The goal of the labs was to predict the concentration of a drug in a solution after being suspended in a spherical delivery system.

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. 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 $k_m = \frac{2D_0}{Daim_p}$. 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.

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
- Use the definitions to calculate the needed values of a, km, D0 and concentration (jaf)
- Loop through each time step to give a concentration and take values for a table every thirty seconds
- Plot the final concentration and mass values
- Calculate theoretical values
- Plot theoretical values
- Display graphs and write to table
- Export table to csv

Output

The graph for the concentration versus time is shown in Figure 1.

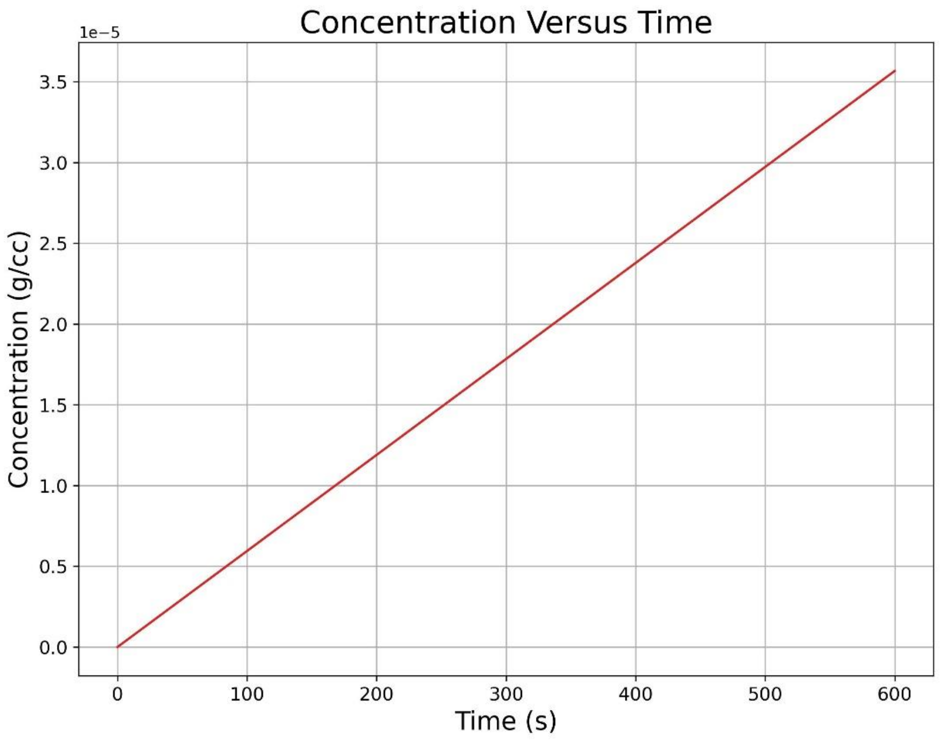

Figure 1: Concentration of the Drug in the Solution for 10 Minutes: the graph of plotting the concentration from the iterative loop versus time.

Figure 2 shows the values of the concentration but times the volume to get the mass concentration versus time.

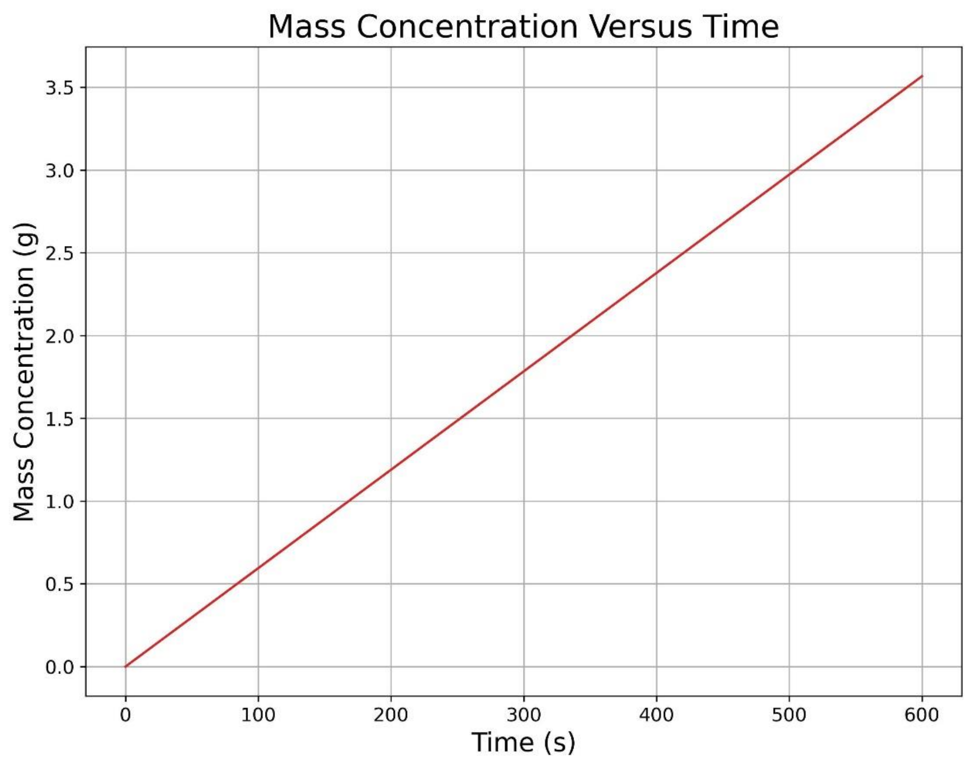

Figure 2: Mass concentration of the drug versus time: The mass concentration is in grams time is over 10 minutes.

The theoretical values for the mass concentration were calculated using the linear equation and shown in figure 3.

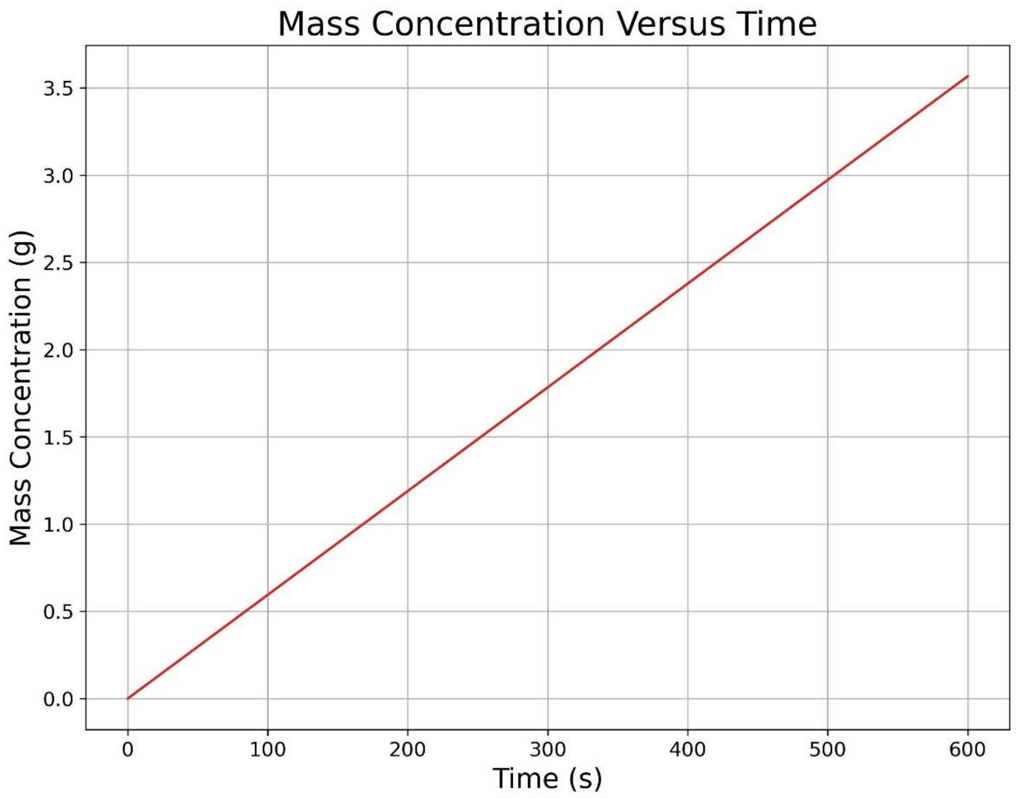

Figure 3: The theoretical values for mass concentration over time: The concentration was calculated and then graphed.

Tables of the values of concentration are shown below in Table 1 with the final values at the bottom.

Table 1: Drug concentration and mass concentration versus time: The final concentraion values are higfhlighted in orange at the final time section of 600 seconds or 10 minutes.

Time (s)	Drug Concentration (g/cc)	Mass Concentration (g)
0	0	0
30	1.78E-06	0.1784
60	3.57E-06	0.3568
90	5.35E-06	0.5351
120	7.14E-06	0.7135
150	8.92E-06	0.8919
180	1.07E-05	1.07
210	1.25E-05	1.249
240	1.43E-05	1.427
270	1.61E-05	1.605
300	1.78E-05	1.784
330	1.96E-05	1.962
360	2.14E-05	2.14

390	2.32E-05	2.319
420	2.50E-05	2.497
450	2.68E-05	2.675
480	2.85E-05	2.853
510	3.03E-05	3.032
540	3.21E-05	3.21
570	3.39E-05	3.388
600	3.57E-05	3.567

Discussion

Python was able to use the functions and looping iterations to graph and tabulate the values of the concentration of the drug after ten minutes. This can be used in the future for similar drugs and show that python could possbly do something more complex like a complex sherwood number and be able to graph the proper values.

Appendix

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

#Import table
vol = 100 #liters
vol = vol*1000 #switch to mL
sigfigs = 4
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
rhoA_s = 0.05 #g/cm^3
rho_drug = 1 #g/cc
rhoA_f = 0 #initial concentration
massA_f = rhoA_f*vol #g

#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
```

```
snp_time = 30 #look at every thirty seconds

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

#conversion factors
#convert cp to pa*s
cP_Pas = 1e-3

vis = vis*cP_Pas

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)

#calculate for jaf
Jaf = Jaf_funct(Km,rhoA_f,rhoA_s)

#print(As, " ", D0, " ", Km, " ", Jaf) #print check

#initial table formatting
L_time = [timei]
```

```
L_timet = ['%.*g' % (sigfigs,timei)]
L_conc = [rhoA_f]
L_conct = ['%.*g' % (sigfigs,rhoA_f)]
L_mass = [massA_f]
L_masst = ['%.*g' % (sigfigs,massA_f)]

#set up loop
icount = 0
jcount = 0 #snapshot in time

while timei<timef:
    J_af = Jaf_funct(Km,rhoA_f,rhoA_s)
    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
    jcount += 1
    timei = icount*dt
    stime = jcount*dt

    #snapshot
    if stime == snp_time:
        L_time.append(timei)
        L_timet.append('%.*g' % (sigfigs,timei))
        L_conc.append(rhoA_f)
        L_conct.append('%.*g' % (sigfigs,rhoA_f))
        L_mass.append(massA_f)
        L_masst.append('%.*g' % (sigfigs,massA_f))
        jcount = 0

print(f'concentration after ten minutes: {rhoA_f:.4g} g/cc')
print(f'concentration after ten minutes: {massA_f:.4g} g')

#create dictionary
results = {"Time (s)":L_time,"Drug Concentration (g/cc)": L_conct,"Mass 
Concentration (g)":L_masst}
df1 = pandas.DataFrame(results)
df1.set_index("Time (s)",inplace = True)
print(df1)
print("")

#make the theory graph
```

```python
phi = Km*p_surf*Num_part/vol
xt = np.linspace(0,600,30)
y_massconc = rhoA_s*(1-np.exp(-phi*xt))*vol

fig = plt.figure(figsize = (10,8))
plt.plot(L_time,L_conc, color = "tab:red")
plt.title("Concentration Versus Time", fontsize = 20)
plt.xlabel("Time (s)",fontsize = 16.5)
plt.ylabel("Concentration (g/cc)", fontsize = 16.5)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.grid(True)
plt.show()
fig.savefig('Euler_Lab/Concentration.jpeg',dpi = 300,bbox_inches = 'tight')

fig = plt.figure(figsize = (10,8))
plt.plot(L_time,L_mass, color = "tab:red")
plt.title("Mass Concentration Versus Time", fontsize = 20)
plt.xlabel("Time (s)",fontsize = 16.5)
plt.ylabel("Mass Concentration (g)", fontsize = 16.5)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.grid(True)
plt.show()
fig.savefig('Euler_Lab/MassConcentration.jpeg',dpi = 300,bbox_inches = 'tight')

fig = plt.figure(figsize = (10,8))
plt.plot(xt,y_massconc, color = "tab:red")
plt.title("Mass Concentration Versus Time", fontsize = 20)
plt.xlabel("Time (s)",fontsize = 16.5)
plt.ylabel("Mass Concentration (g)", fontsize = 16.5)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.grid(True)
plt.show()
fig.savefig('Euler_Lab/MassConcentrationTheory.jpeg',dpi = 300,bbox_inches =
'tight')

L_dfs = [df1]
with open('Euler_Lab/ConcentrationTable.csv','w',newline='') as f:
    for df in L_dfs:
        df.to_csv(f)
        f.write("\n")
```