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 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 and solving for km with . D0 is found using . With a being the radius of the molecule solved for using 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.

A graph with a red line

Description automatically generated

**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.

A graph with a red line

Description automatically generated

**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.

A graph with a red line

Description automatically generated

**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

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")