

Python In ChemE

End Term Evaluation



ARMEET LUTHRA

Assignment 1 : STOKES LAW

Problem statement :

In this assignment we were required to compare the analytical and numerically calculated terminal velocity of a sphere in a medium of given density and viscosity.

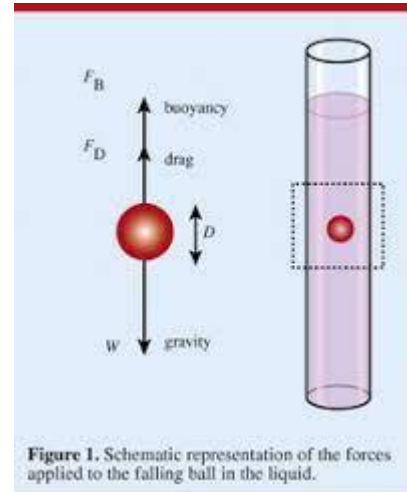
Physics and equations :

- A freely falling spherical ball experiences three forces:
- Gravitational force downwards,
- Buoyancy upwards,
- Drag force, which acts opposite to the direction of velocity of the particle

Assignment 1

- The equation of motion can be written as: $m \frac{du}{dt} = mg - \rho V g - 6\pi\eta r u$.

- m = mass of sphere
- u = velocity of sphere
- g = gravitational acceleration
- ρ = density of fluid
- V = volume of sphere
- r = radius of sphere
- η = viscosity of the fluid



Assignment 1

Algorithm

- 1) Input the necessary parameters (density of liquid, sphere, viscosity of liquid, radius of sphere, tolerance and integration time step)(initial velocity of sphere is assumed to be 0)
- 2) The equation of motion is integrated using linear approximation to integral:

$$du/dt \approx [u(t+\Delta t) - u(t)]/\Delta t$$

$$u(t+\Delta t) = g\Delta t(1 - \rho V/m) + u(t) [1 + 6\pi r \eta \Delta t/m]$$

- 3) This is implemented using a loop incrementing the Δt each time till the error between $u(t+\Delta t)$ and $u(t)$ is less than the given tolerance.
- 4) The final u will be the terminal velocity
- 5) Analytical terminal velocity is calculated by putting acceleration=0 in the equation of motion.
- 6) For a sphere it comes out to be: $v_t = (\rho_s - \rho_l) * g * r^2 * 2 / (9 * \eta)$

Python Code


```
import sys, os.path

rs=float(input("denisty of solid sphere\n"))
r=float(input("radius of solid sphere\n"))
rl=float(input("denisty of liquid\n"))
n=float(input("visocity of liquid\n"))
t=float(input("Enter tolerance\n"))
d=float(input("Enter time difference\n"))
ae=100
vt0=0
while ae>t:
    vt=vt0+d*((rs-rl)/rs*9.8-9/2*n*vt0/(r*r*rs))
    if vt!=0 :

        ae=abs(vt-vt0)
        vt0=vt

vt1=(rs-rl)*9.8*r*r*2/(9*n)
print('Terminal velocity using the numerical method:', vt)
print('Terminal velocity using the analytical method:', vt1)
print("% error using numeical method : ",abs((vt-vt1)*100/vt1))
```

Output

```
denisty of solid sphere
8050
radius of solid sphere
1
denisty of liquid
1000
visocity of liquid
1
Enter tolerance
0.0001
Enter time difference  
0.01
Terminal velocity using the numerical method: 15335.444581786996
Terminal velocity using the analytical method: 15353.333333333334
% error using numeical method : 0.11651379643728299
```

Assignment 2 : Heat Diffusion Equation

Problem Statement:

In this assignment we were required to find the TEMPERATURE PROFILE OVER TIME for a flat plate, subject to certain fixed boundary conditions and also animate the changes. This was done by solving the Heat diffusion equation.

Physics of problem

The heat conduction equation at steady state is given by:

$$\frac{dT}{dt} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right)$$

$$\alpha = \frac{k}{\rho C_p}, k = \text{thermal conductivity}, \rho$$

$= \text{density of material}, C_p = \text{specific heat capacity}$

Assignment 2

Algorithm

- 1) The flat plate is divided into square cells
- 2) We use the approximation:

$$\frac{\partial^2 T}{\partial x^2} = \frac{T(x + \Delta x, y) + T(x - \Delta x, y) - 2 * T(x, y)}{\Delta x^2}$$

- 3) Using this approximation, the heat conduction equation becomes:

$$\begin{aligned} T(x, y, t + \Delta t) &= \gamma (T(x + \Delta x, y, t) + T(x - \Delta x, y, t) + T(x, y - \Delta y, t) \\ &\quad + T(x, y + \Delta y, t) - 4 * T(x, y, t)) + T(x, y, t) \end{aligned}$$

$$\gamma = \frac{\alpha \Delta t}{\Delta x^2}, \text{ assume } \Delta x = \Delta y, \text{ for numerical stability: } \Delta t \leq \frac{\Delta x^2}{4\alpha}$$

- 4) We solve this using 3 loops where we apply the above equation for each cell and save the temperature value for different time stamps in a 3d array.
- 5) Using FuncAnimation we animate the temperature profile.

Python Code

```
%matplotlib notebook
import sys, os.path
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.animation import FuncAnimation

arr=np.zeros((51,51,501))
arr[0,:,:]=50
# decomment the next line for the second part of assignment
# arr[:,50,:]=50
gamm=2*0.1

for k in range(1,501):
    for i in range(1,50):
        for j in range(1,50):
            arr[i,j,k]=gamm*(arr[i-1,j,k-1]+arr[i,j-1,k-1]+arr[i+1,j,k-1]+arr[i,j+1,k-1]-4*arr[i,j,k-1])+arr[i,j,k-1]

fig = plt.figure()

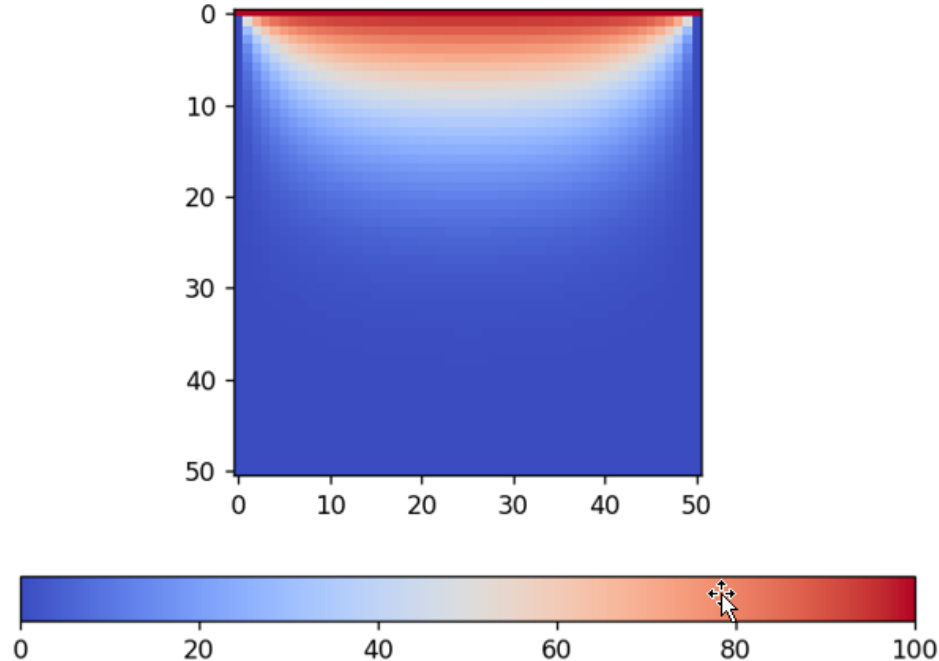
def animate(i):
    global image
    image = plt.imshow(arr[:, :, i], cmap='coolwarm')

    return image,

anim = FuncAnimation(fig, animate, frames = 500, interval = 10, repeat=False, blit = True)
image=plt.imshow(arr[:, :, 500], cmap='coolwarm')
plt.colorbar(image,orientation='horizontal')
plt.show()
```

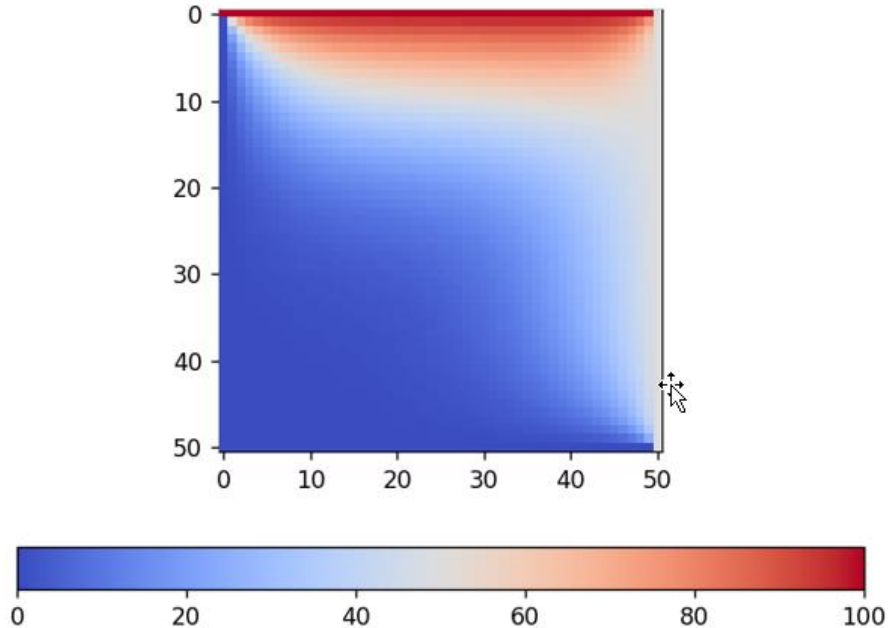
Output

1. Initial boundary condition: Top edge at 100° , rest plate at 0° . Final temperature profile for this case is :

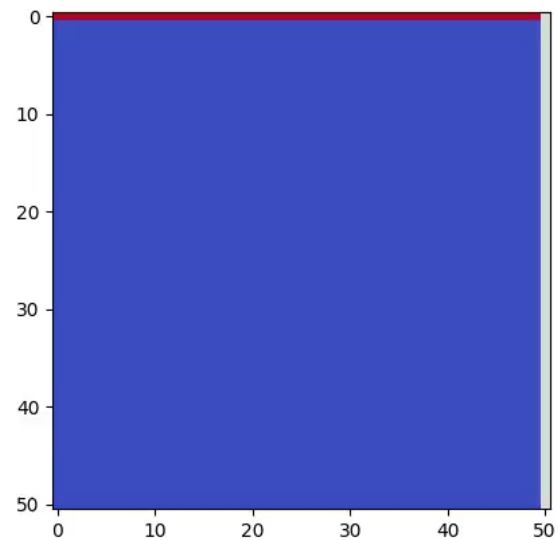
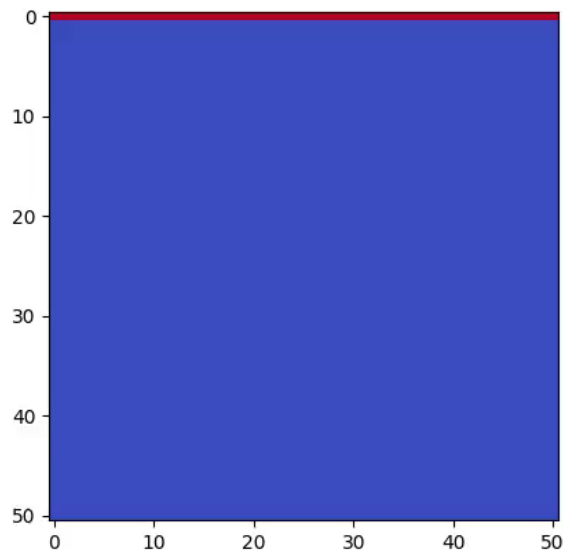


Output

2. Initial boundary condition: Top edge at 100° , Right edge at 50° , rest plate at 0° . Final temperature profile for this case is:



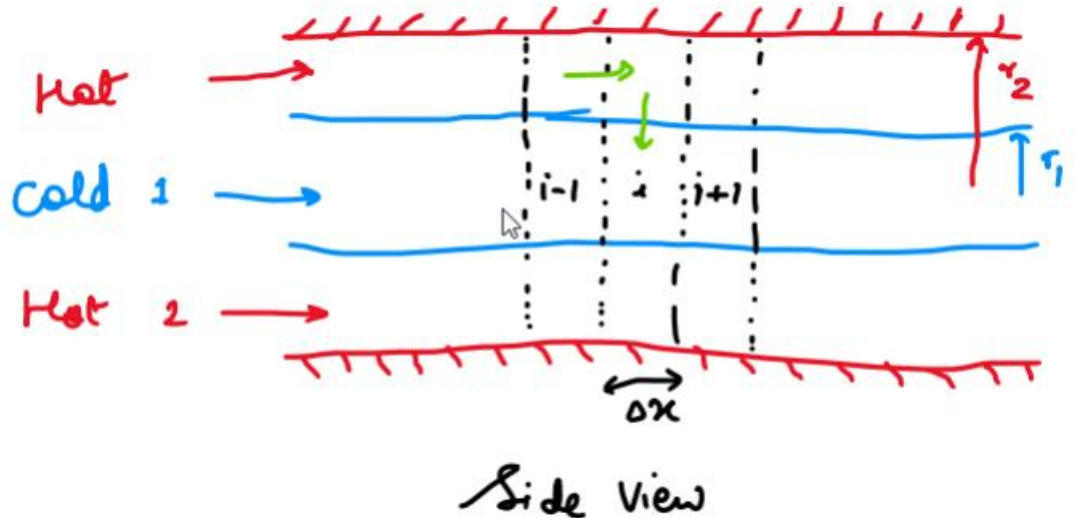
Animation



Assignment 4 : Double pipe heat exchanger

Problem statement :

In this assignment we were required to solve and obtain the transient response of Temperature with time for the given concentric cylinder double pipe heat exchanger for co-current flow



Assignment 4

Physics and equations :

Using energy balance:

$$\text{Accumulation} = I_{\text{in}} - Q_{\text{out}} + \text{generation}$$


The heat balance equation for concentric heat exchanging cylinders is:

Assignment 4

$$\Rightarrow \frac{dT_1}{dt} = \frac{\dot{m}_1 C_{p1} (T_1(i-1) - T_1(i)) + U \cdot 2\pi r_1 \Delta x (T_2(i) - T_1(i))}{\rho_1 C_{p1} A_{c1} \Delta x}$$

$$\frac{dT_2}{dt} = \frac{\dot{m}_2 C_{p2} (T_2(i-1) - T_2(i)) - U \cdot 2\pi r_1 \Delta x (T_2(i) - T_1(i))}{\rho_2 C_{p2} A_{c2} \Delta x}$$

Here T_1 : temperature of inner cylinder, T_2 : temperature of outer cylinder

Assignment 4 : Algorithm

Algorithm

1) The length along the cylinder is divided into several small pieces.

2) Using Euler approximation, the heat balance equation becomes:

$$T1[x, t + \Delta t] = \gamma1 \\ * (m1 * cp1 * (T1[x - \Delta x, t] - T1[x, t]) + U * 2 * \pi * r1 \\ * \Delta x * (T2[x, t] - T1[x, t])) + T1[x, t]$$

$$T2[x, t + \Delta t] = \gamma2 \\ * (m2 * cp2 * (T2[x - \Delta x, t] - T2[x, t]) - U * 2 * \pi * r2 \\ * \Delta x * (T2[x, t] - T1[x, t])) + T2[x, t]$$

$$\gamma1 = \frac{\Delta t}{\rho1 * cp1 * \pi * r1 * r1 * \Delta x}$$

$$\gamma2 = \frac{\Delta t}{\rho2 * cp2 * \pi * (r2 * r2 - r1 * r1) * \Delta x}$$

Assignment 4 : Algorithm

We solve this using 2 loops, where we apply the above equation for each piece length and save the temperature value for different time stamps in a 2d array.

- 3) Using FuncAnimation, we animate the colormap of temperature profile for different time stamps.
- 4) Using plt.pause(0.005), we plot the Temperature vs distance for both inner and outer cylinders for different time stamps.

Python Code

```
arr1[:,0]=T0
arr2[:,0]=T0
arr1[0,:]=T1i
arr2[0,:]=T2i
x=np.linspace(delx/2,L-delix/2,n)

for k in range(1,1001):
    for i in range(1,100):
        arr1[i,k]= gm1*( m1*cp1*(arr1[i-1,k-1]-arr1[i,k-1])  + U*2*pi*r1*delx*(arr2[i,k-1]-arr1[i,k-1]) )+arr1[i,k-1]
        arr2[i,k]= gm2 *(m2*cp2*(arr2[i-1,k-1]-arr2[i,k-1])  - U*2*pi*r2*delx*(arr2[i,k-1]-arr1[i,k-1]))+arr2[i,k-1]

array=np.zeros((150,n,1001))
for i in range(50):
    array[i,:,:]=arr2[:,:,:]
for i in range(51,150):
    array[i,:,:]=arr1[:,:,:]

plt.figure(1)

plt.xlabel("distance")
plt.ylabel("temperature")
for i in range(1000):
    plt.plot(x,arr1[:,i],x,arr2[:,i])

    plt.pause(0.005)

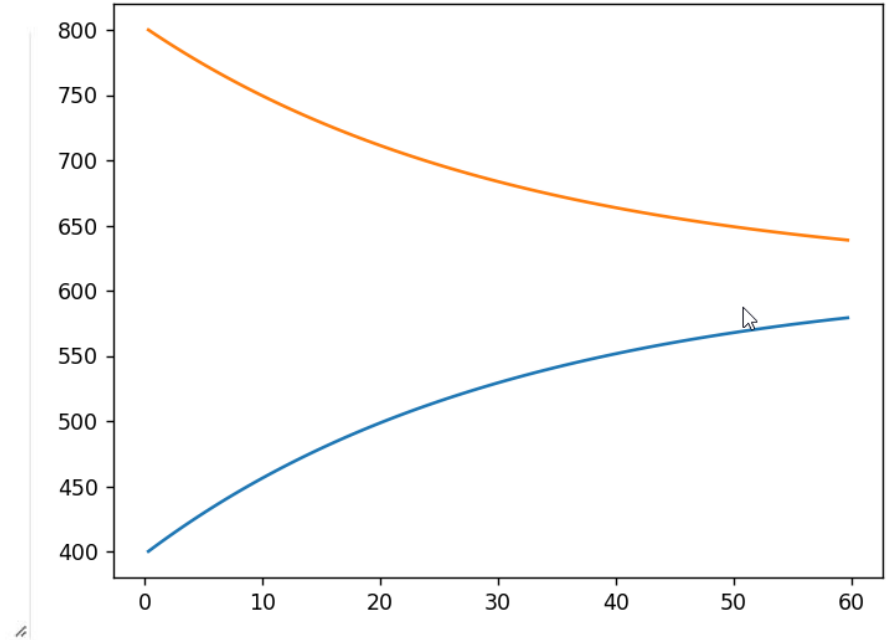
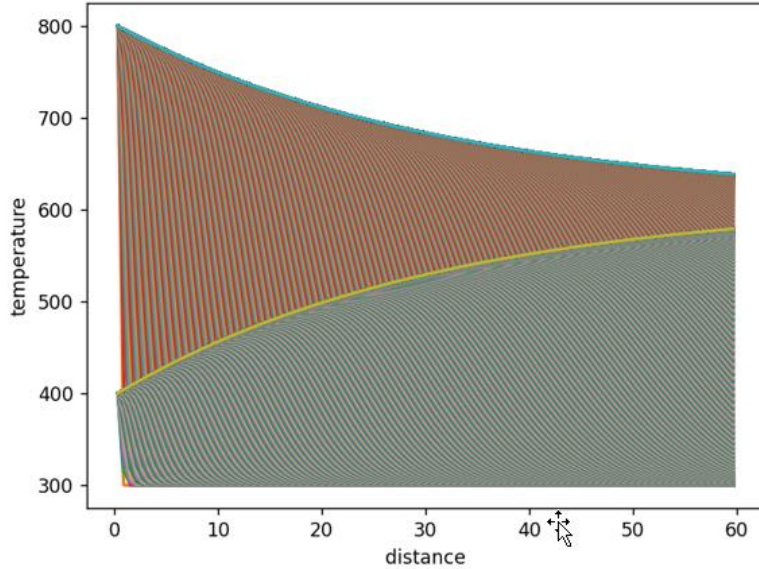
fig=plt.figure()
def animate(i):
    global image
    image = plt.imshow(array[:, :, i], cmap='coolwarm')
    plt.xlabel("distance(1unit=0.6m)")
    plt.ylabel("along radius(in cm)")

    return image,

anim = FuncAnimation(fig, animate,frames = 1000, interval = 10,repeat=False, blit = True)
image = plt.imshow(array[:, :, 1000], cmap='coolwarm')
plt.colorbar(image,orientation='horizontal')
plt.show()
```

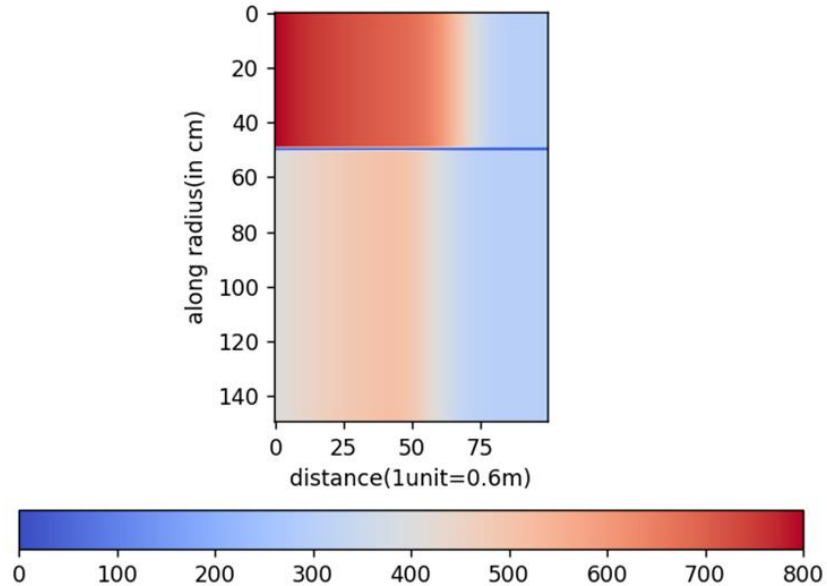
Output

1. At $t=1000\text{sec}$, Temperature vs distance plot



Output

2) Temperature profile color map at some instant



Animation

