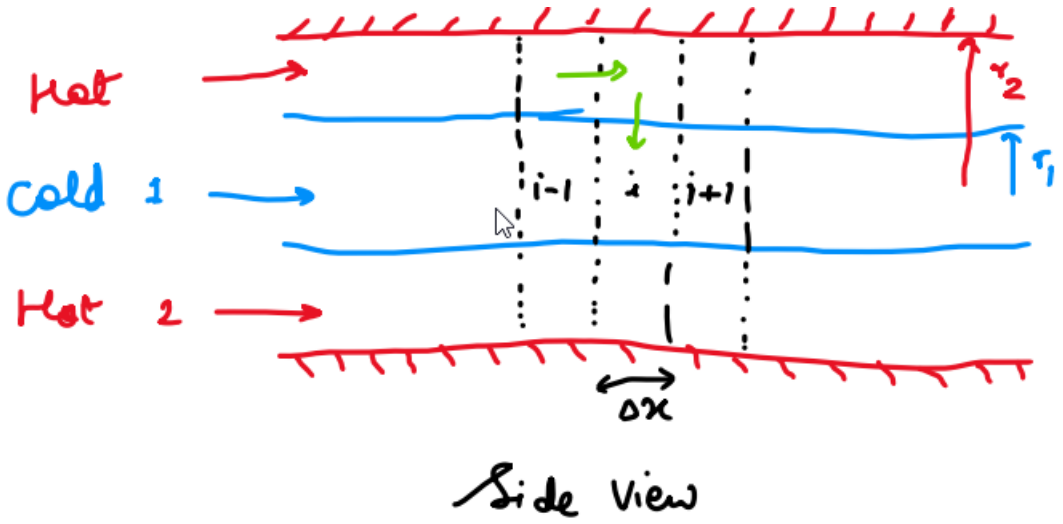
REPORT

ASSIGNMENT 3, Python in ChemE

ARMEET LUTHRA

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.



Attempted solution

Physics of problem

Using energy balance:

Accumulation = In - Out + Generation

The heat balance equation for concentric heat exchanging cylinders is:

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

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

Here $T1$: *temperature of inner cylinder*, $T2$: *temperature of outer cylinder*

Algorithm

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

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

$$T1[x, t+\Delta t] = \gamma 1 * (\ m1 * cp1 * (T1[x-\Delta x, t] - T1[x,t]) + U * 2 * \pi * r1 * \Delta x * (T2[x,t] - T1[x,t])\ ) + T1[x,t]$$

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

$$\gamma 1 = \frac{\Delta t}{\rho 1 * cp1 * \pi * r1 * r1 * \Delta x}$$

$$\gamma 2 = \frac{\Delta t}{\rho 2 * cp2 * \pi * (r2 * r2 - r1 * r1) * \Delta x}$$

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.

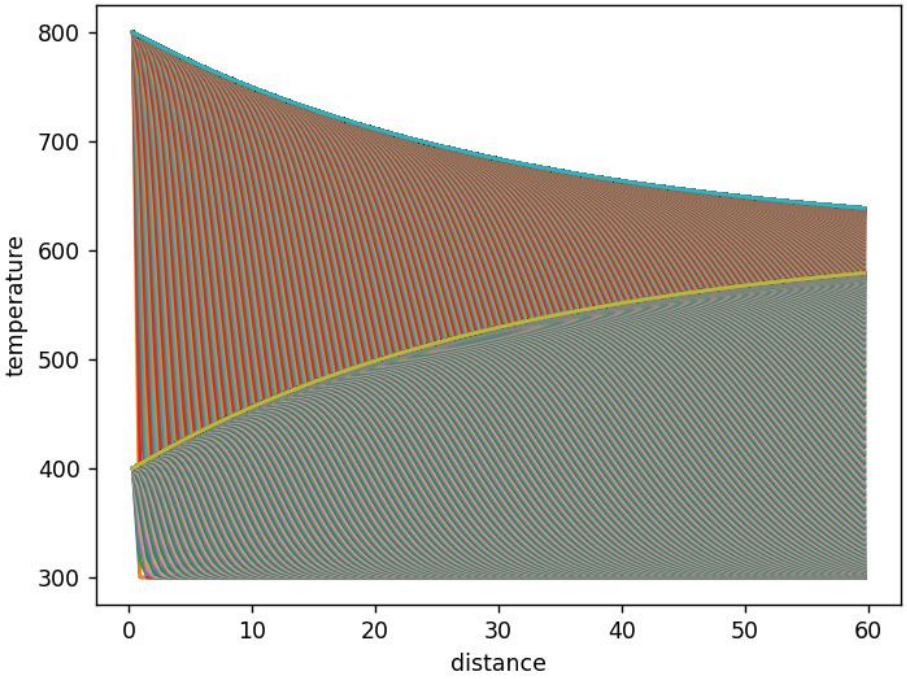
Main Code

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

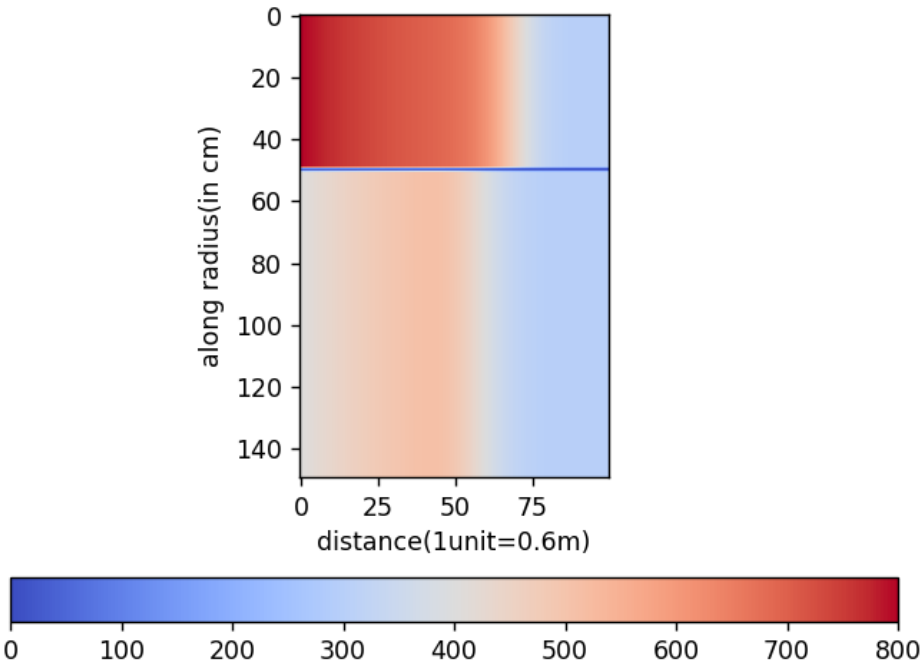
```
anim = FuncAnimation(fig, animate,frames = 1000, interval = 10,repeat=False, blit = True)
image = plt.imshow(array[:,:,1000], cmap='coolwarm')
plt.colorbar(image,orientation='horizontal')
plt.show()
```

Output

1. At t=1000sec, Temperature vs distance plot



2) Temperature profile color map at some instant


0
20
40
60
80
100
120
140
along radius(in cm)
0
25
50
75
distance(1unit=0.6m)
0
100
200
300
400
500
600
700
800