ENGN2020 – HOMEWORK8

Problem 1

Part (a)

$$0 = -v\frac{dC}{dz} + ka(C^{sat} - C)$$

The initial condition is that at z = 0, where is exactly the outlet of O_2, C = 0. That`s $C(z=0)=0$.
Let:

$\theta \equiv \frac{C-C^{sat}}{-C^{sat}}$, then $\frac{d\theta}{dz} = -\frac{1}{C^{sat}}\frac{dC}{dz}$

The original equation can be written as:

$$v\frac{d\theta}{dz} = -ka\theta$$

Let:

$l \equiv \frac{z}{v/ka} = \frac{kaz}{v}$, then $\frac{dz}{dl} = \frac{v}{ka}$,

The equation above can be written as:

$$\frac{d\theta}{dl} = -\theta$$

Where:

$$\theta = \frac{C^{sat} - C}{C^{sat}}$$

$$l = \frac{z}{v/ka} = \frac{kaz}{v}$$

The initial condition becomes:

$$\theta(l=0) = 1$$

Part (b)

$$R\frac{dq}{dt} + \frac{1}{C}q = V$$

The equation can be written as:

$$R\frac{dq}{dt} = \frac{1}{C}(VC - q)$$

Let:

$\theta \equiv \frac{q-VC}{-VC}$, then $\frac{d\theta}{dt} = -\frac{1}{VC}\frac{dq}{dt}$,

The equation can be written as:

$$RC\frac{d\theta}{dt} = -\theta$$

Let:

$\tau \equiv \frac{t}{RC}$, then $\frac{dt}{d\tau} = RC$,

The equation can be written as:

$$\frac{d\theta}{d\tau} = -\theta$$

Where:

$$\theta = \frac{q - VC}{-VC}$$

$$\tau = \frac{t}{RC}$$

The initial condition becomes:

$$\theta(\tau = 0) = 1$$

Problem 2

$$mC_P \frac{dT}{dt} = -hA(T - T^{\infty}) + a(1 + sinbt)$$

Where:

$$m = 1kg$$
$$a = 200W$$
$$b = 0.1\ s^{-1}$$
$$A = 0.02\ m^2$$
$$T_0 = 25\ °C$$

The function can be rewritten as:

$$\frac{dT}{dt} = -\frac{hA}{mC_P}(T - T^{\infty}) + \frac{a}{mC_P}(1 + sinbt)$$

In python, the function can be defined as:

```
import math
from scipy.integrate import odeint
'''
* @name: f_fit
* @description: the function to be fitted by the experimental points
* @param t: time, in form of array
* @param cp: heat capacity
* @param h: surface heat-transfer coefficient
* @return: the corresponding temperature of given t
'''
def f_fit(t,cp,h):
    #copy the input parameter of cp and h
    alpha = cp
    beta = h
    #the pre-defined parameter
    m = 1
    a = 200
    b = 0.1
    A = 0.02
    T_inf = 25

    #define the function to be solved by numerical method in lambda format
    f = lambda y,t: -beta*A*(y-T_inf)/m/alpha + a * (1+ math.sin(b*t))/m/alpha

    #set the initial guess by T0
    y0 = 25

    #solve for temperature of given time
    results = odeint(f, y0, t)

    #resize the temperature as 1*141
    results.resize((141))
    return results
```

The experiment data can be loaded by following code:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

#load the experimental data
data = np.load('thermal-block.npz')
#get time
t = data['times']
#get temperature
```

```
realTemperature = data['temperatures']
#resize the temperature as the same size with time
realTemperature.resize((141))
```

Use scipy.optimize.curve_fit to fit for the experimental results to calculate cp and h. After calculation,

$$C_P = 1395.71665065\mathrm{J/kg/K}$$
$$h = 871.58696023\mathrm{W/m^2/K}$$

```
#curve fit to calculate cp and h
popt, pcov = curve_fit(f_fit, t, realTemperature)
#calculate the temperature based on fitting result
fitTemperature = f_fit(t,popt[0],    popt[1])

fig, axs = plt.subplots(1, 1)
axs.plot(t,fitTemperature,'r')
axs.plot(t,realTemperature,'bx')
axs.set_xlabel('time')
axs.set_ylabel('CA')

plt.show()
```

The plot containing the best-fit model solution and the experimental points is shown in Fig.1.

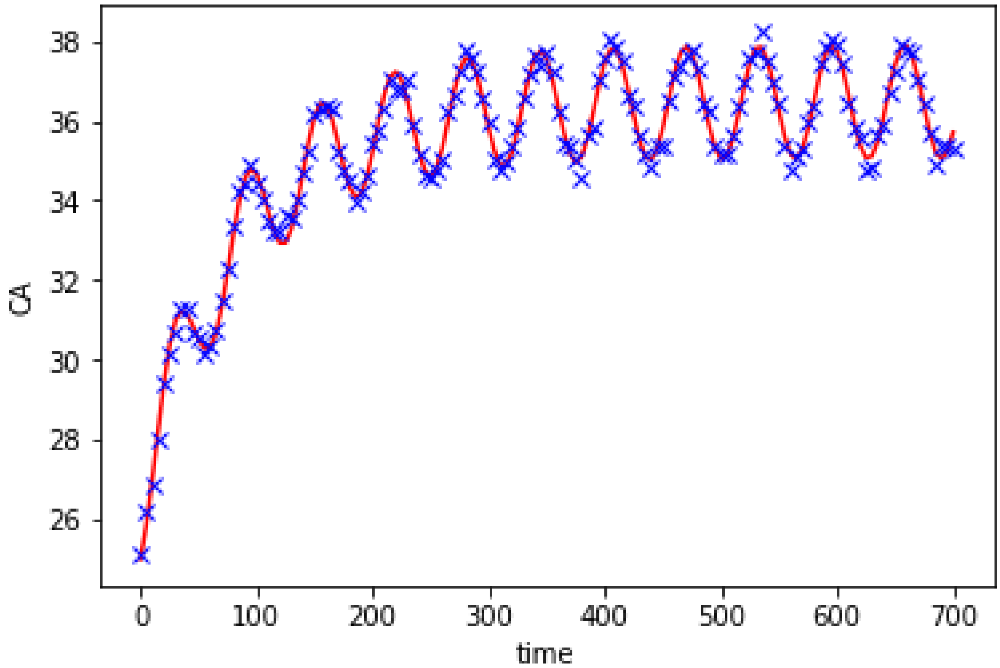


Fig 1. The plot containing the best-fit model solution and the experimental points

Problem 3

Part (a)

According to the problem:

$$\rho C_p \frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2} \cdots\cdots\cdots\cdots\cdots ①$$

At x = 0:
$$-kA\frac{\partial T}{\partial x} = -hA(T - T^{\infty}) \cdots\cdots\cdots ②$$

At x = L:
$$-kA\frac{\partial T}{\partial x} = 0 \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ③$$

Let $\theta \equiv \frac{T-T^{\infty}}{T_0-T^{\infty}}, then\ \frac{\partial\theta}{\partial x} = \frac{1}{T_0-T^{\infty}}\frac{\partial T}{\partial x}, \frac{\partial\theta}{\partial t} = \frac{1}{T_0-T^{\infty}}\frac{\partial T}{\partial t}, \frac{\partial^2\theta}{\partial x^2} = \frac{1}{(T_0-T^{\infty})^2}\frac{\partial^2 T}{\partial x^2}$

Then, equation 3 becomes:

$$-kA(T_0 - T^{\infty})\frac{d\theta}{dx} = 0$$

Let $z \equiv \frac{x}{L}$, $then\ \frac{\partial\theta}{\partial x} = \frac{1}{L}\frac{\partial\theta}{\partial z}$, $\frac{\partial^2\theta}{\partial x^2} = \frac{1}{L^2}\frac{\partial^2\theta}{\partial z^2}$

Then, equation 3 becomes:

$$\frac{\partial\theta}{\partial z} = 0, \qquad at\ z = 1$$

Then equation 2 becomes:

$$\frac{\partial\theta}{\partial z} = \frac{hL}{k}\theta, \qquad at\ z = 0$$

Let $\eta \equiv \frac{hL}{k}$, then equation 2 becomes:

$$\frac{\partial\theta}{\partial z} = \eta\theta, \qquad at\ z = 0$$

From above $\frac{\partial^2\theta}{\partial x^2} = \frac{1}{(T_0 - T^{\infty})^2}\frac{\partial^2 T}{\partial x^2}$ and $\frac{\partial^2\theta}{\partial x^2} = \frac{1}{L^2}\frac{\partial^2\theta}{\partial z^2}$, then:

$$\frac{\partial^2 T}{\partial x^2} = \frac{(T_0 - T^{\infty})^2}{L^2}\frac{\partial^2\theta}{\partial z^2}$$

Then equation 1 becomes:

$$\rho C_p(T_0 - T^{\infty})\frac{\partial\theta}{\partial t} = \frac{k(T_0 - T^{\infty})^2}{L^2}\frac{\partial^2\theta}{\partial z^2}$$

Then:

$$\frac{L^2\rho C_p}{k(T_0 - T^{\infty})}\frac{\partial\theta}{\partial t} = \frac{\partial^2\theta}{\partial z^2}$$

Let $\tau = \frac{t}{\frac{L^2\rho C_p}{k(T_0 - T^{\infty})}}$, then:

$$\frac{\partial\theta}{\partial\tau} = \frac{\partial^2\theta}{\partial z^2}$$

In conclusion:

$$\begin{cases} \dfrac{\partial\theta}{\partial\tau} = \dfrac{\partial^2\theta}{\partial z^2} \\ \dfrac{\partial\theta}{\partial z} = \eta\theta, at\ z = 0 \\ \dfrac{\partial\theta}{\partial z} = 0, at\ z = 1 \end{cases}$$

Where:

$$\begin{cases} \theta = \dfrac{T - T^{\infty}}{T_0 - T^{\infty}} \\ z = \dfrac{x}{L} \\ \eta = \dfrac{hL}{k} \\ \tau = \dfrac{k(T_0 - T^{\infty})}{L^2\rho C_p}t \end{cases}$$

Part (b)

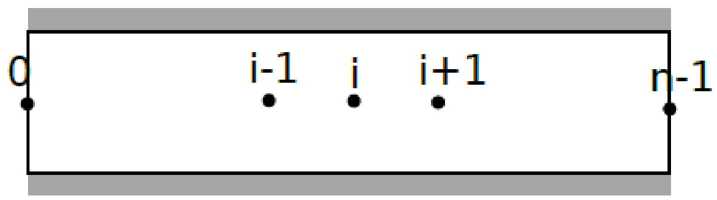


Fig 2. The finite element model

To discretizing these equations:

$$\frac{\partial^2\theta}{\partial z^2} = \frac{\theta[i+1] - 2\theta[i] + \theta[i-1]}{\Delta z^2}$$

$$\frac{\partial\theta}{\partial z} = \frac{\theta[i+1] - \theta[i-1]}{2\Delta z}$$

The first boundary condition becomes:

$$At\ z = 0, \qquad \frac{\partial\theta}{\partial z}|_{i=0} = \frac{\theta[1] - \theta[-1]}{2\Delta z} = \eta\theta[0], \text{then}\ \theta[-1] = \theta[1] - 2\Delta z\eta\theta[0]$$

Therefore:

$$\frac{\partial\theta}{\partial\tau}|_{i=0} = \frac{\partial^2\theta}{\partial z^2}|_{i=0} = \frac{\theta[1] - 2\theta[0] + \theta[-1]}{\Delta z^2} = \frac{2\theta[1] - 2\theta[0] - 2\Delta z\eta\theta[0]}{\Delta z^2}$$

The second boundary condition becomes:

$$At\ z = 1, \qquad \frac{\partial\theta}{\partial z}|_{i=n-1} = \frac{\theta[n] - \theta[n-2]}{2\Delta z} = 0, then\ \theta[n] = \theta[n-2]$$

Therefore:

$$\frac{\partial\theta}{\partial\tau}|_{i=n-1} = \frac{\partial^2\theta}{\partial z^2}|_{i=n-1} = \frac{\theta[n] - 2\theta[n-1] + \theta[n-2]}{\Delta z^2} = \frac{2\theta[n-2] - 2\theta[n-1]}{\Delta z^2}$$

The governing function becomes:

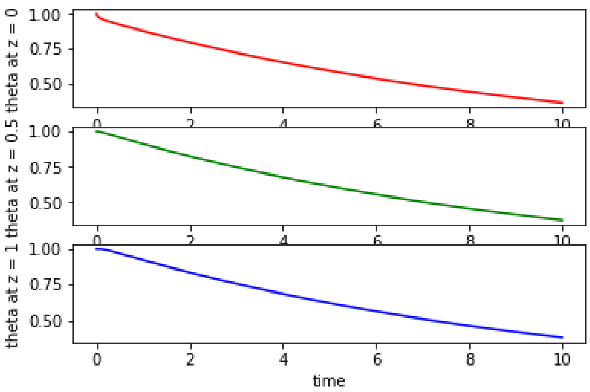
$$\frac{\partial\theta}{\partial\tau} = \frac{\theta[i+1] - 2\theta[i] + \theta[i-1]}{\Delta z^2}$$

The code for class Derivatives is shown as below:

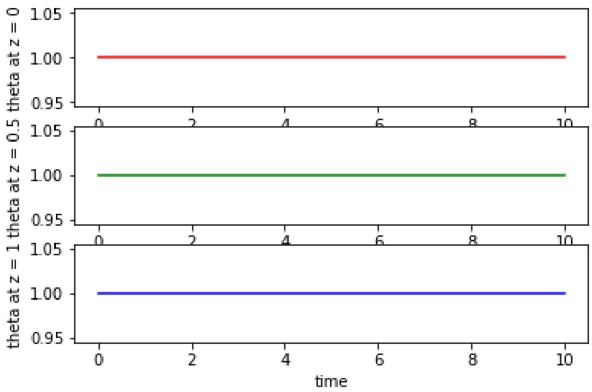
```
class Derivatives:
    '''
    * @name: __init__
    * @description: the constructor of class
    * @param n: parameter of number of elements
    * @param eta: parameter of eta
    '''
    def __init__(self,n,eta):
        self.n = n
        self.step = 1/n
        self.eta = eta
    '''
    * @name: __call__
    * @description: the call function
    * @param y: current parameter for 'theta'
    * @param t: input value of 'tao'
    * @return: dy, the value of dydt
    '''
    def __call__(self, y, t):
        dy = np.zeros((self.n))
        #the boundary condition at z=0
        dy[0] = (2*y[1]-2*y[0]-2*self.step*self.eta*y[0])/self.step/self.step
        #the governing function
        for i in range(1,self.n-1):
            dy[i] = (y[i+1]- 2*y[i]+y[i-1])/self.step/self.step
        #the boundary condition at z=n-1
        dy[self.n-1] = (2*y[self.n-2]-2*y[self.n-1])/self.step/self.step
        #return dy
        return dy
```

Part (c)I

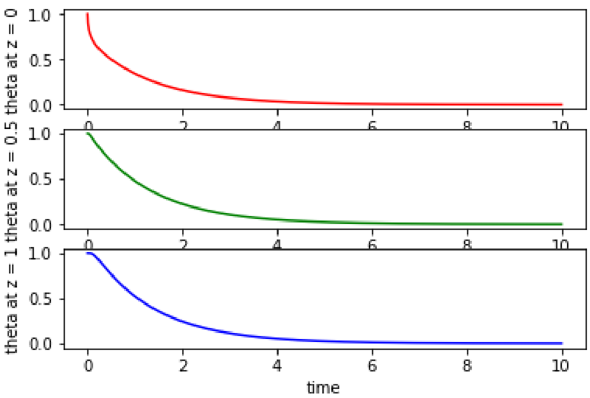
In order to show the effect of η, the plots of θ versus t at different positions, namely z = 0, z = 0.5, z = 1 are shown in Fig.3.



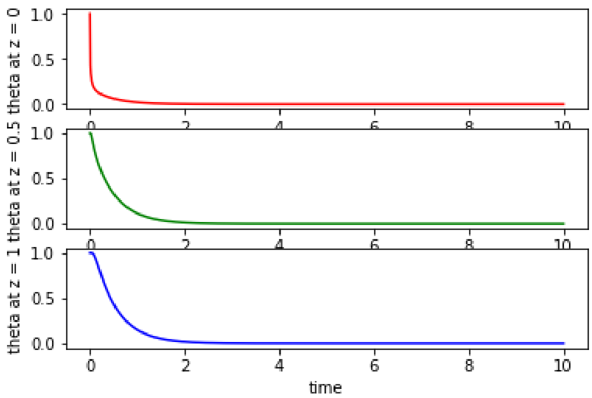
(a) η = 0.1



(b) η = 0



(c) η = 1



(d) η = 10

Fig 3. plots of θ versus t at different positions, namely z = 0, z = 0.5, z = 1

From Fig.3, it can be found that, when η = 0, the left end is also perfectly insulated, there will be no heat exchange between block and environment, the block keeps its original temperature.

When $\eta \neq 0$, as η increases, the system takes less time to reach the stable state. So η has effect on the "speed" of transferring heat.

To solve this problem numerically, the following code was used. In this piece of code, scipy.integrate.odeint is used.

```
'''
* @name: solve
* @description: the function to get solve the finite element problem
* @param n: the number of finite elements in the block
* @param eta: parameter of transferring heat
'''
def solve(n,eta):
    #define a object based on input n and eta
    dydt = Derivatives(n, eta)
    #get the initial guess of y0
    y0 = np.ones((n))
    #get t
    t = np.linspace(0,10,1000)
    #solve for y
    y = odeint(dydt, y0, t)
    #show y versus t at different positions, z = 0, z=0.5, z=1
    fig, axs = plt.subplots(3, 1)
    axs[0].plot(t,y[:,0],'r')
    axs[0].set_xlabel('time')
    axs[0].set_ylabel('theta at z = 0')

    axs[1].plot(t,y[:,n//2],'g')
    axs[1].set_xlabel('time')
    axs[1].set_ylabel('theta at z = 0.5')

    axs[2].plot(t,y[:,n-1],'b')
    axs[2].set_xlabel('time')
    axs[2].set_ylabel('theta at z = 1')

    plt.show()
```