Transient temperature one-dimensional profile

Consider the problem of a large cylindrical tank of water. At its initial conditions, the whole tank is at 25 °C and there is no heat flowing throughout the liquid. However, at t > 0, the surface of the tank suddenly experiences an instantaneous temperature increase to T = 80 °C (for example, due to a strongly exothermic reaction in the interface or through the flushing of hot steam above the tank). This temperature difference will generate an immediate flux of heat transfer between the surface of the tank and its bulk.

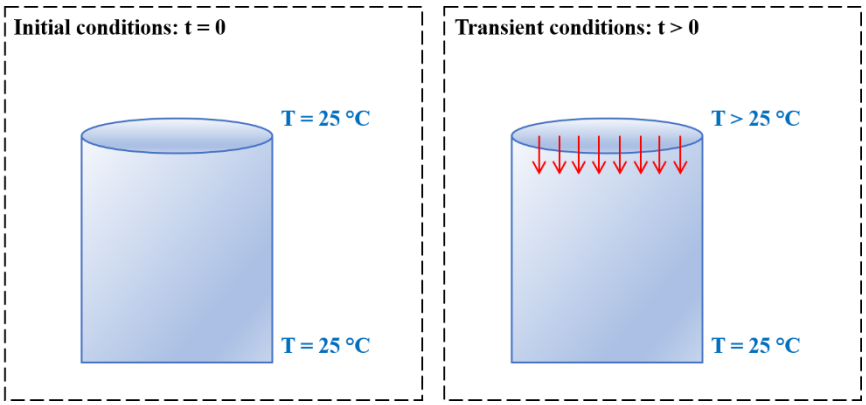

Figure 1. Schematic drawing of the water tank in its initial and in its transient conditions

Let us consider the transient energy balance that applies for an infinitesimally thin slab of liquid. For this layer, the variation of internal energy will be equal to the amount of heat entering from above minus the amount of heat leaving from the bottom.

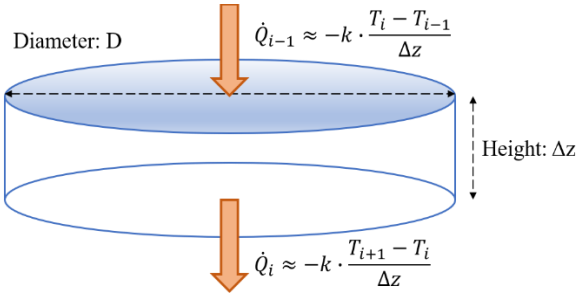

Figure 2. Control volume of a very thin slab of liquid for the transient energy balance

If E_i is the volumetric internal energy of each slab i, then the energy balance above translates into the equation below:

$$\frac{dE_i}{dt} \cdot \underbrace{\left(\frac{\pi \cdot D^2}{4} \cdot \Delta z\right)}_{control\ volume} = \dot{Q}_{i-1} \cdot \underbrace{\frac{\pi \cdot D^2}{4}}_{cross\ area} - \dot{Q}_i \cdot \underbrace{\frac{\pi \cdot D^2}{4}}_{cross\ area} \tag{1}$$

Dividing both sides of the equation above by the volume of the slab:

$$\frac{dE_i}{dt} = \frac{\dot{Q}_{i-1} - \dot{Q}_i}{\Delta z} \tag{2}$$

If the liquid has constant density and constant heat capacity, the left-hand side of the equation above can be written as:

$$\rho \cdot C_P \cdot \frac{\partial T_i}{\partial t} = \frac{\dot{Q}_{i-1} - \dot{Q}_i}{\Delta z} \tag{3}$$

Finally, substitution of Fourier's law of conductivity in the right-hand side of the equation above will generate Eq. (4). For this problem, you can neglect the effect of convective heat transfer.

$$\rho \cdot C_P \cdot \frac{\partial T_i}{\partial t} = \frac{-k \cdot \frac{\Delta T}{\Delta z}\Big|_{i-1} + k \cdot \frac{\Delta T}{\Delta z}\Big|_i}{\Delta z} \tag{4}$$

For an infinitesimally small Δz, the equation above will reduce to the differential equation shown below:

$$\rho \cdot C_P \cdot \frac{\partial T}{\partial t} = k \cdot \frac{\partial^2 T}{\partial x^2} \tag{5}$$

$$\frac{\partial T}{\partial t} = \frac{k}{\rho \cdot C_P} \cdot \frac{\partial^2 T}{\partial x^2} \tag{6}$$

The equation above is a partial differential equation whose analytical solution can be obtained through a bit of mathematical ingenuity and some labor. However, we will try to come up with a numerical solution for this scenario. This will illustrate how useful and practical it is to have knowledge of a programming language plus some numerical methods.

Let us start with the forward Euler's method for integrating differential equations. Roughly speaking, Euler's method can be written as below:

$$T_i|_{k\cdot\Delta t} = T_i|_{(k-1)\cdot\Delta t} + \frac{\partial T_i}{\partial t}\Big|_{(k-1)\cdot\Delta t} \cdot \Delta t \tag{7}$$

That is to say, the temperature of each layer i at each time t = k·Δt can be approximated by numerically adding the temperature of that same layer in a previous instant to the temporal derivative of T (calculated at that previous instant) times the time step Δt. This is extremely convenient for us, since we already know the temperature of all layers at t = 0 (they were T = 25 °C as defined in the first paragraph of this exercise!). Euler's method will often be a practical alternative when integrating a time derivative, for time only moves forward. If you are interested, you can read more about Euler's method in Wikipedia – https://en.wikipedia.org/wiki/Euler_method

As for the numerical values for the temporal derivatives of T, they can be calculated for each slab i as per Eq. (4) shown before, that is:

$$\frac{\partial T_i}{\partial t} = -\frac{k}{\rho \cdot C_P} \cdot \frac{\Delta T}{\Delta z^2}\bigg|_{i-1} + \frac{k}{\rho \cdot C_P} \cdot \frac{\Delta T}{\Delta z^2}\bigg|_{i} \tag{8}$$

$$\frac{\partial T_i}{\partial t} \approx -\frac{k}{\rho \cdot C_P} \cdot \frac{T_i - T_{i-1}}{\Delta z^2} + \frac{k}{\rho \cdot C_P} \cdot \frac{T_{i+1} - T_i}{\Delta z^2} \tag{9}$$

$$\frac{\partial T_i}{\partial t} \approx \frac{k}{\rho \cdot C_P} \cdot \left(\frac{T_{i+1} - T_{i-1}}{\Delta z^2}\right) \tag{10}$$

This numerical method is called *finite difference*, of which there are several variants. In the expression above, the variant of choice is *second-order central*. You can read more about finite differences in Wikipedia – https://en.wikipedia.org/wiki/Finite_difference

That means that, for each time k·Δt and slab i, the time derivative can be numerically approximated. Of course, solving this derivative also demands that one knows the temperatures at the two extremities of the space we are integrating in at all times. This is quite simple. In the upper extremity, where z = 0, the temperature of the liquid surface is defined by the problem as 80 °C. At the bottom, if we allow for a space sufficiently great z = H, the temperature will be the same as it was before the heating began, i.e. the bulk temperature will be 25 °C.

It is incredibly easy to write this algorithm in Python. As a matter of fact, we have written it for you. It is as below:

```
#-------------------------------------------------------------------------------
import numpy
import matplotlib.pyplot as plt
import math
#-------------------------------------------------------------------------------

H = 0.025               # depth of penetration for simulation, m
n = 25                  # number of nodes for finite differences
T0 = 298.15             # temperature at t = 0, K
T_surface = 353.15      # temperature at the surface of the liquid, K
T_bulk = 298.15         # temperature at the bulk of the liquid, K
dz = H/n                # delta z for finite differences
k = 0.6                 # thermal conductivity of water, W/(m*K)
rho = 1000              # density of water, kg/m3
Cp = 4200               # heat capacity of water, J/(kg*K)
t_final = 120           # total time of simulation, s
dt = 0.5                # delta t for Euler method

z = numpy.linspace(dz/2,H-dz/2,n)
t = numpy.arange(0,t_final+dt,dt)
T = numpy.ones(n)*T0
dTdt = numpy.empty(n)
Q = numpy.empty(n)

for j in range(1,len(t)):
        plt.clf()
        for i in range(1,n-1):
                dTdt[i] = -(k/(rho*Cp))*((T[i]-T[i-1])/dz**2)+(k/(rho*Cp))*((T[i+1]-
T[i])/dz**2)
        dTdt[0] = -(k/(rho*Cp))*((T[0]-T_surface)/dz**2)+(k/(rho*Cp))*((T[1]-
T[0])/dz**2)
        dTdt[n-1] = -(k/(rho*Cp))*((T[n-1]-T[n-2])/dz**2)+(k/(rho*Cp))*((T_bulk-T[n-
1])/dz**2)
        T = T+dTdt*dt

        plt.figure(1)
        plt.plot(z,T)
```

```
plt.axis([0, H, 273.15, 373.15])
plt.xlabel('Depth / m')
plt.ylabel('Temperature / K')
plt.text(0.015,350,'t = ' + str(t[j]) + ' s')
plt.show()
plt.pause(0.01)
```

You are invited to copy and paste this code in your compiler of choice. If you do so, you will see the development of a temperature profile between the liquid surface and the liquid bulk. For this example, the temperature at the surface was fixed at 80 °C. The number of nodes for the finite difference method was fixed at n = 25, the timestep Δt was fixed at 0.5 s, and the profile develops until t = 120 seconds.

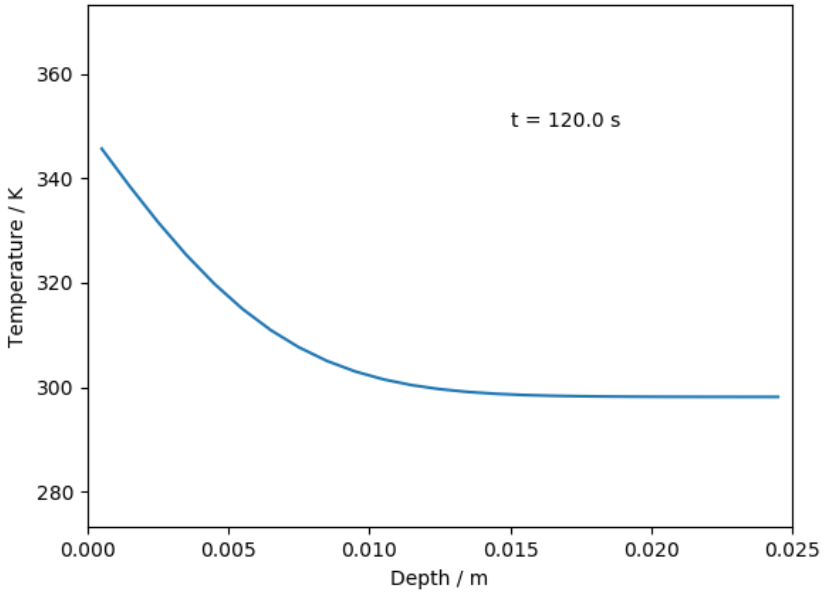

Figure 3. Temperature profile in the liquid after 120 seconds

You might be asking yourself: what is the influence of H in this algorithm? Ideally, none. If H is big enough, it does not matter if we set H = 2.5 cm, 3.0 cm or 5.0 cm. Higher values of H will however mean that we have less calculations being made close to the gas-liquid interface, which is where they really matter (since it is there that we have the largest temperature variations). It is not a good idea to simply fix a very large H. On the other hand, a very small H will mean that we risk not giving the penetration profile enough space to develop, resulting in an unrealistic boundary condition.