ESEIAAT - Universitat Politècnica de Catalunya

EXERCISE 2: UNSTEADY 1D CONDUCTION HEAT TRANSFER

Marc Monclús Montalvez

November 26th, 2022

Subject ..... NUMERICAL METHODS IN HEAT AND MASS TRANSFER
Course 2022-2023

Contents

1 Problem description:

1. Exercise
Write a computer program (according to the specifications given in section 4.) to solve the heat conduction equation in the situation described in section 2.

Ensure that the code is correct (code verification).

Choose a suitable mesh and time step.

A report with the following steps should be presented:
i) problem definition and description of the methodology;
ii) code structure and validation strategies;
iii) numerical studies (e.g. mesh, time step, relaxation factors, etc.);
iv) influence of the physical parameters (e.g. geometry, boundary conditions, type of materials, etc.);
v) conclusions.

Comments:
i) you must write your own code without using available software;
ii) a normal PC is enough to do this exercise;
iii) do not hesitate to contact us if you have any questions about the exercise.

2. Problem definition
We are interested in the temporal evolution of the ground temperature distribution (see Fig. 1).

$$\rho c_p \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2}$$

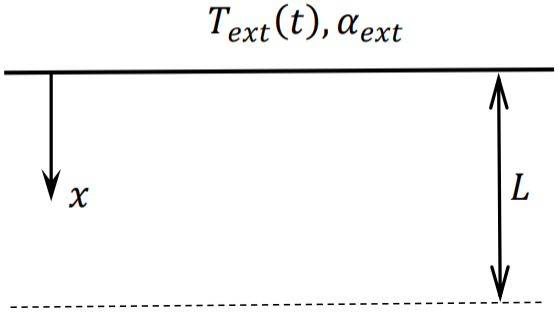

Figure 1: : Schematic representation of the proposed problem

$L\,(m)$	$\rho\left(\frac{kg}{m^3}\right)$	$c_p\left(\frac{J}{kgK}\right)$	$\lambda\left(\frac{W}{mK}\right)$	$\alpha_{ext}\left(\frac{W}{m^2K}\right)$	$T_0\,(°C)$	$A_0\,(°C)$	$A_1\,(°C)$	$A_2\,(°C)$	$T_W\,(°C)$
2.0	2300	700	1.6	8.5	19.0	19.0	5.7	9.1	19.0

Table 1: Physical parameters

A one-dimensional time-dependent heat conduction equation will be assumed to be valid to model the ground temperature (therefore, neglecting humidity changes or other aspects that may be actually relevant). Assuming constant thermal conductivity, the governing equation can be written as:

$$T_{ext}(t) = A_0 + A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t) \tag{1}$$

where time t is in seconds, and

$$\omega_1 = \frac{2\pi}{24 \times 3600}; \qquad \omega_1 = \frac{2\pi}{24 \times 365 \times 3600}; \tag{2}$$

The initial temperature of the ground, $t = 0$, is t_0. Physical parameters are given in Table 1.

2 Solving analytically:

For static solids without radiation:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_v u\ \rho\ dV = -\int \vec{\dot{q}} \cdot \vec{n}\ dS \tag{3}$$

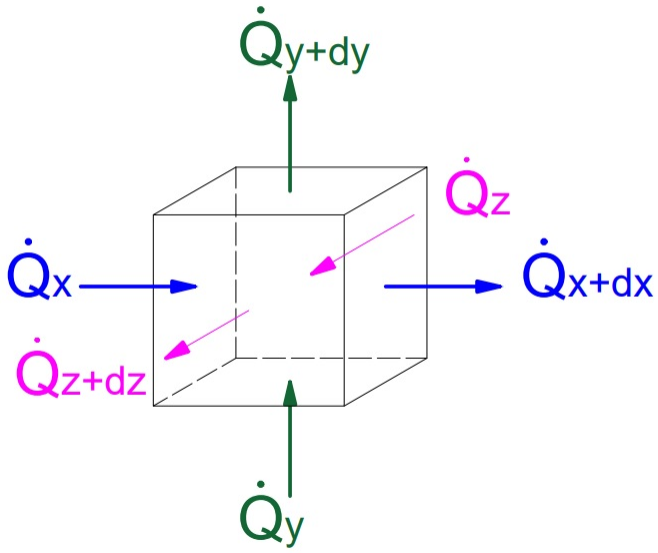

Figure 2: Diferential Control Volume

Differential form:

$$\rho = constant \tag{4}$$

Energy balance,

$$\rho \frac{\partial u}{\partial t}\, dV = \dot{Q}_x - \dot{Q}_{x+dx} + \dot{Q}_y - \dot{Q}_{y+dy} + \dot{Q}_z - \dot{Q}_{z+dz} \tag{5}$$

$$\rho \frac{\partial u}{\partial t}\, dV = \frac{\partial \dot{Q}_x}{\partial x}\, dx - \frac{\partial \dot{Q}_y}{\partial y}\, dy - \frac{\partial \dot{Q}_z}{\partial z}\, dz \; + \dot{Q}_u \tag{6}$$

$$where : du = q\; dT \tag{7}$$

$$Q_x = -\lambda \frac{\partial T}{\partial x}\, dx\; dz \tag{8}$$

$$\rho g \frac{\partial T}{\partial t} dV = \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right)\; dx\; dy\; dz + \frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right)\; dx\; dy\; dz + \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right)\; dx\; dy\; dz \tag{9}$$

Simplifying the red therms in the previous expression (the control volume) we obtain the following:

$$\rho g \frac{\partial T}{\partial t} = \nabla \left(\lambda \frac{\partial T}{\partial x} \right) + \dot{q}_u \tag{10}$$

Input Data:

Physical data:
$e,\ w,\ h,\ \lambda,\ \dot{q}_u,\ T_p,\ \alpha_A$

For 1D and steady case according to the constitutive equation: $T_{(t,x,y,z)} = T_x$

$$\rho g \frac{\partial T}{\partial t} = \nabla (\lambda \nabla T) + \dot{q}_u \tag{11}$$

$$D = \lambda \frac{d^2 T}{dx^2} + \dot{q}_u \tag{12}$$

$$T = -\frac{\dot{q}_u}{2\lambda} x^2 + C_1 x + C_2 \tag{13}$$

$$\dot{q} = -\lambda \frac{dT}{dx} = q_u x + \lambda C_1 \tag{14}$$

$$x = 0 \quad \rightarrow \quad T = T_{w1} = C_2 \tag{15}$$

$$x = e \quad \rightarrow \quad T = T_{w2} = \frac{-\dot{q}_u}{2\lambda} e^2 + C_1 e + C_2 \tag{16}$$

$$\begin{cases} C_1 = \frac{\dot{q}_u}{2\lambda} + \frac{T_{w2} - T_{w1}}{e} \\ C_2 = T_{w1} \end{cases} \tag{17}$$

Specific boundary conditions:

$$x = 0 \quad \rightarrow \quad \dot{q}_{conv} = \dot{q}_{cond} \tag{18}$$

Then, $\dot{q}_u$ will be neglected with $x = 0$,

$$\alpha_A \left(T_A - T_w \right) = \dot{q}_u x + C_1 = \frac{\dot{q}_u e}{2\lambda} + \frac{T_{w2} - T_{w1}}{e} \tag{19}$$

$$x = e \quad \rightarrow \quad \dot{q}_{cond} = \dot{q}_{conv} \tag{20}$$

Finally, after the following expression, C_1 and C_2 can be obtained:

$$\dot{q}_u e + \frac{\dot{q}_u e}{2\lambda} + \frac{T_{w2} - T_{w1}}{e} = \alpha_B \left(T_{w2} - T_{w1}\right) \tag{21}$$

For this problem the α_B coefficient will be used in the numerical resolution being the first control volume the one in contact with the atmosphere as we will see.

3 Numerical solution:

This problem is a transient case due to the temperature variation on the face of the ground in contact with the atmosphere, being this temporal variation of the temperature defined by equation (1).

In this case, the problem is solved as if it were a steady one-dimensional case with a constant heat transfer coefficient, but for each time-step.

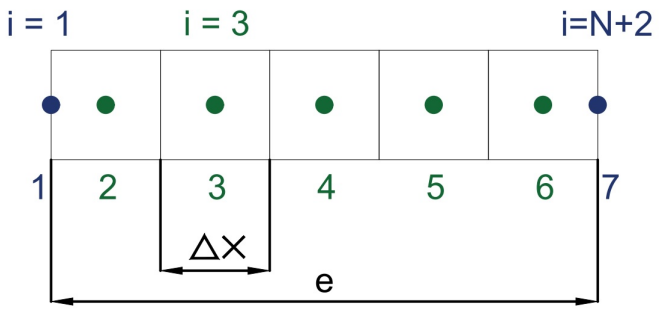

Figure 3: Discretization of the 1D domain

In previous figure, we can see an example of discretization of the domain along the x axis where N are the input that defines the number of control and the corresponding internal nodes.

We can see these points in green, but there are two other points on the boundaries, which can be seen in blue in the previous image. In this blue points is where the boundary conditions are introduced.

3.1 Algorithm

3.1.1 Grid generation:

The following expressions define the boundary conditions of both ends of the domain: Nodes: N+2 Faces: N+1

$X_{w[i]} = (i-1)\triangle x \qquad (i = 1 \;\; to \;\; i = N+1)$

$X_{p[1]} = 0$

$X_{p[N+2]} = e$

$X_{p[i]} = \frac{X_{w[i]} + X_{w[i]-1}}{2} \qquad (i = 2 \;\; to \;\; i = N+1)$

Here we defined

3.1.2 Discretization equations:

For the internal nodes ($i = 2 \;\; to \;\; i = N+1$):

$$\left(\frac{\lambda_w S_w}{d_{pw}} + \frac{\lambda_e S_e}{d_{pe}}\right) T_p = \frac{\lambda_w S_w}{d_{pw}} T_w + \frac{\lambda_e S_e}{d_{pe}} T_e + \dot{q}_{up} V_p$$

where the coefficients are obtained and the the temperatures inside the Gauss-Seidel iterative solver,
$T_{[0]} = A_0 + A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t)$

$a_p = \left(\frac{\lambda_w S_w}{d_{pw}} + \frac{\lambda_e S_e}{d_{pe}}\right),$

$a_w = \frac{\lambda_w S_w}{d_{pw}},$

$a_e = \frac{\lambda_e S_e}{d_{pe}},$

$b_p = \dot{q}_{up} V_p$

$Tp[i] = \frac{a_{w[i]} \;\; T_{[i-1]} + a_{e[i]} \;\; T_{[i+1]} + b_{p[i]}}{a_{p[i]}}$

$Assuming, \;\; \lambda = constant \quad \rightarrow \quad \lambda_w = \lambda_e = \lambda,$

$\alpha_A (T_A - T_P) = -\lambda \frac{dT}{dx} \mid_e$

$\alpha_A (T_A - T_P) \approx -\lambda_e \frac{T_E - T_P}{d_{pe}}$

$\left(\frac{\lambda_e}{d_{pe}} + \alpha_A\right) T_P = \frac{\lambda_e}{d_{pe}} T_E + \alpha_A T_P$

Where the coefficients at the boundaries are the following ones:

$a_p T_p = a_e T_e + b_p$

$a_{p[i]} T_{p[i]} = a_{e[i]} T_{[i+1]} + b_{p[i]}$

$a_{p[1]} T_{p[1]} = a_{e[1]} T_{[2]} + b_{p[1]}$

$$T_{[N+2]} = \frac{a_{w[N+2]}T_{[N+1]} + a_{e[N+2]}T_{[N+3]}{}^{*} + b_{p[N+2]}}{a_{P[N+2]}}$$

where the terms in red are neglected.

3.1.3 Time discretization:

The calculations made up to now for a single instant of time are repeated throughout the total time of 100x24x3600 seconds for 500 time-steps (generating more or less time-steps we obtain the result similar to increasing or reducing the number of control volumes) , all within a for loop where inside the Gauss-Seidel, the outside temperature varies as a function of equation (1).

4 Analysis of the code and conclusions:

For the resolution of the discretization equations, the Gauss-Seidel method has been used, but inside a for loop for the time evolution.

Number of nodes: 70
Number of time steps: 500
Total number of iterations: 1214
External Temp in kelvin (L=0) at final time: 21
Final temperature in Kelvin in x=L/2 at final time: 51
Final temperature in Kelvin in x=L at final time: 292
Final time(s): 8640000.0

See also figures 10 and 11 for this result.

The code has been tested plotting the temperatures along "L" for different times to analyze the behavior of the code with the variation of temperature of the external ambient. The following explains different graphs extracted from the program for different times.

At the final time (100*24*3600) we obtain the following graph where the minimum temperature is at x=0 where the ambient is cold and the maximum temperature is near to x=1.30 and at x=L=2m the temperature will remain at 292K always. The color map in the left side of the page is an illustration of the temperature in one-dimensions of the case where the x-axis i the number of control volumes. In this simulation the number of control volumes is 30 and the number of time steps is 150.

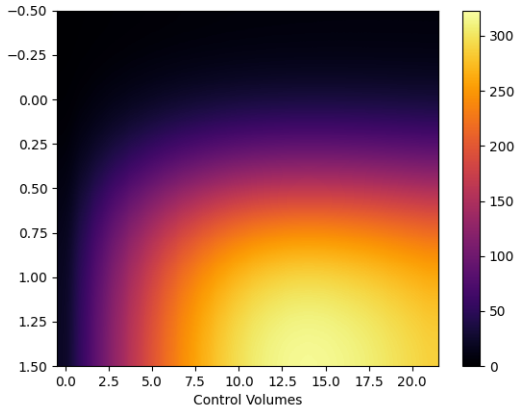

Figure 4: Temperature of the ground along the control volumes where the color-bar is the temperature in Kelvin at the final instant

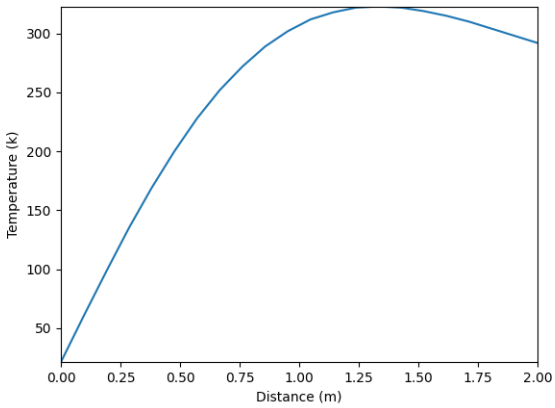

Figure 5: Temperature of the ground along the 2 meters at the final instant

The following two figures are for the same simulation but at the instant t=0, where the temperature distribution along the deep of the ground is 292 kelvin because there has not yet been heat exchange with the outside.

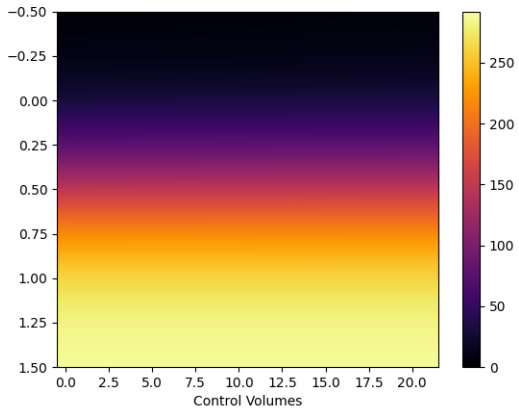

Figure 6: Temperature of the ground along the control volumes where the color-bar is the temperature in Kelvin at the first time-step

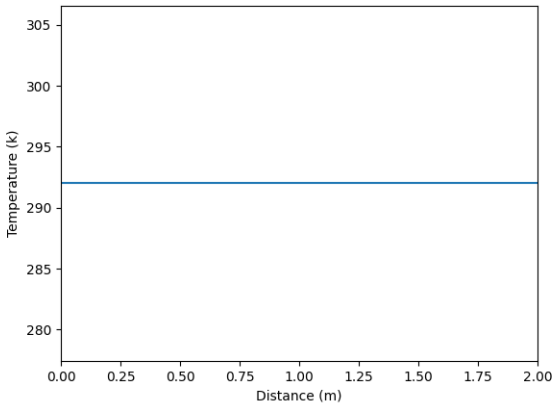

Figure 7: Temperature of the ground along the 2 meters at the first time-step

The next two figures are for the 70% of the total time.

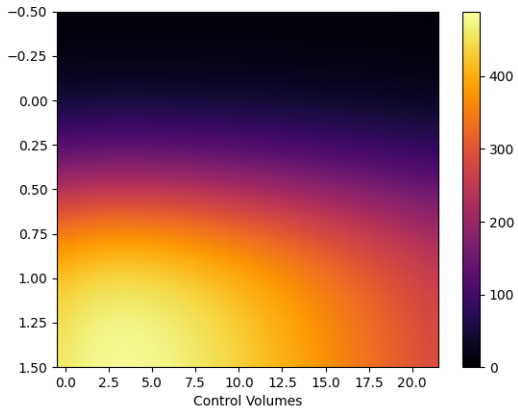

Figure 8: Temperature of the ground along the control volumes where the color-bar is the temperature in Kelvin at the first time-step

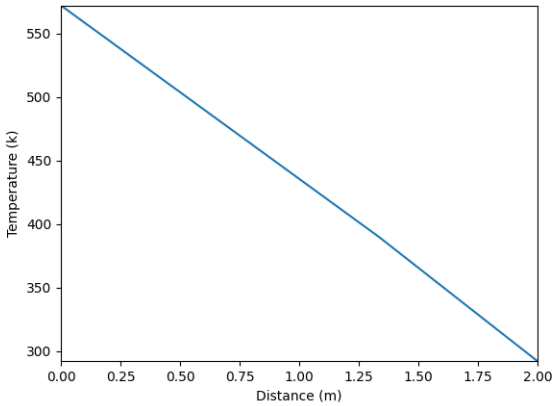

Figure 9: Temperature of the ground along the 2 meters at the first time-step

The following two figures show the temperature distribution for the final time when the number of control volumes is equal to 70.

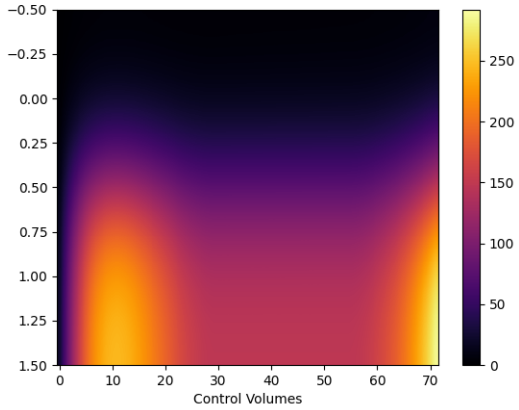

Figure 10: Temperature of the ground along the control volumes where the color-bar is the temperature in Kelvin at the first time-step (70 control volumes

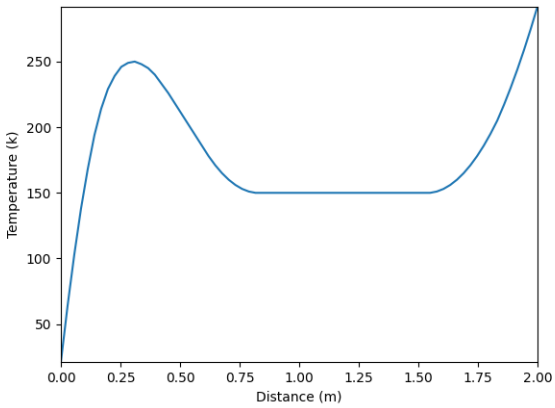

Figure 11: Temperature of the ground along the 2 meters at the first time-step (70 control volumes)

The following two figures show the temperature distribution for the final time when the number of control volumes is equal to 130.

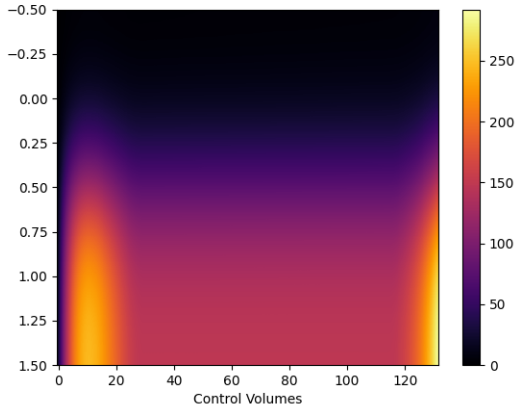

Figure 12: Temperature of the ground along the control volumes where the color-bar is the temperature in Kelvin at the first time-step (130 control volumes)

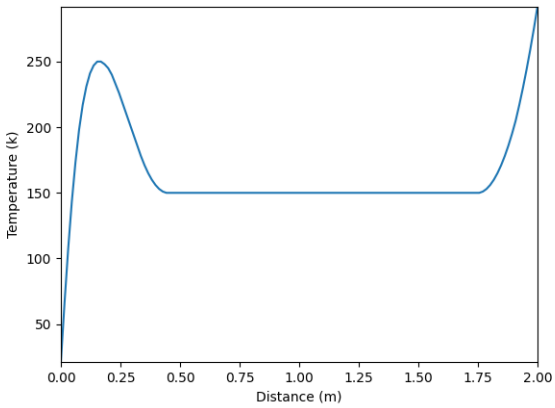

Figure 13: Temperature of the ground along the 2 meters at the first time-step (130 control volumes)

5 Python Code

The code is on the following Github repository owned by the author of this work, where updates can be obtained:

https://github.com/SpaceSubmarine/ConductionHT/tree/main

```
import numpy as np
import matplotlib.pyplot as plt
import math

# Input ========================================
N = 70  # Number of nodes
L = 2  # m
S = 1  # arbitrary surface in m2
delta_x = L / N  # differential length
Vp = S * delta_x  # m3
rho = 2300  # kg/m3
cp = 700  # J/kgK
T_a = 19 + 273  # Kelvin
T_b = 19 + 273  # Kelvin
alpha_ext = 8.5  # W/m2K
lambda_1 = 1.6  # W/mK
max_error = (10 ** (-11))  # Max error Gauss-Seidel
max_iter = 500000  # Maximum number of iterations
T_w = 19 + 273  # Initial Temperature of the cooper in Kelvin
t_step = 500  # Number of time divisions 150

total_time = 100*24*3600  # the total time purposed at the exercise
time = np.linspace(0, total_time, t_step)  # total_time + 1

# Here we define the distance in meters for the walls in a vector
x_wall = np.linspace(0, L, N + 1)  # Value of x at the walls
for i in range(1, N):
    x_wall[i] = i * delta_x

x_wall[0] = 0
x_wall[-1] = L

# Coefficients and temperatures initialization ====================
domain = np.linspace(0, L, N + 2)
T_dist = np.zeros(len(domain)).astype(int)
T_dist[:] = T_w  # initial temperature of the ground
T_dist[0] = T_a  # initial external temperature
T_dist[-1] = T_b
```

```
ae = np.zeros(len(domain))
ae[:] = lambda_1 * S / delta_x
aw = ae.copy()
ap = np.zeros(len(domain))
bp = np.zeros(len(domain))
q_flux = np.zeros(len(domain))

aw[0] = 0
ae[0] = lambda_1 * S / delta_x
ap[0] = (ae[0] * T_dist[1] + bp[0]) / T_dist[0]
ae[N + 1] = 0
aw[N + 1] = lambda_1 * S / delta_x
ap[N + 1] = (aw[N + 1] * T_dist[N] + bp[N + 1]) / T_dist[N]
q_flux[0] = alpha_ext * (T_dist[1] - T_dist[0])
bp[0] = q_flux[0] * rho * Vp

A_0 = 19 + 273
A_1 = 5.7 + 273
A_2 = 9.1 + 273
omega_1 = (2 * math.pi / (24 * 3600))
omega_2 = (2 * math.pi / (365 * 24 * 3600))

name = 1
total_iter = 0
stored_diff = []
# ======================================================================
for t in time:
    plt.ion()
    print("Time step: ", t)

    # Gauss-Seidel=====================================================
    diff = 100000  # Arbitrary but different from zero
    Iter = 1  # Initializing the number of iterations
    while diff > max_error and Iter < max_iter:
        T_dist_old = T_dist.copy()

        for i in range(1, N + 1):
            # =================================================
            T_dist[0] = A_0 + A_1 * math.sin(math.radians(omega_1 * t)) +
            ↪  A_2 * math.sin(
                math.radians(omega_2 * t))  # External temperature in time
            # =================================================
            ap[i] = float(lambda_1 * S / delta_x) + (lambda_1 * S /
            ↪  delta_x)
            aw[i] = float(lambda_1 * S / delta_x)
            ae[i] = float(lambda_1 * S / delta_x)
```

```
            bp[i] = -lambda_1 * ((T_dist[i] - T_dist[i - 1]) / delta_x) *
            ↪  Vp
            # ========================================================
            T_dist[i] = float((aw[i] * T_dist[i - 1] + ae[i] * T_dist[i +
            ↪  1] + bp[i]) / ap[i])

        diff = max(T_dist - T_dist_old)
        stored_diff.append(diff)
        Iter += 1
        total_iter += 1
        print(Iter)

    plt.clf()
    # animated plot
    plt.figure(1)
    plt.plot(domain, T_dist)
    plt.xlabel('Distance (m)')
    plt.axis([0, max(domain), min(T_dist), max(T_dist)])
    plt.ylabel('Temperature (k)')
    plt.savefig("./heat_plot/" + str(name) + ".png")
    plt.ion()


    plt.figure(2)
    plt.imshow((domain, T_dist), aspect='auto', interpolation='gaussian',
    ↪  cmap='inferno')
    plt.colorbar()
    plt.xlabel('Control Volumes')
    plt.autoscale()
    plt.show()
    plt.savefig("./heat_field/" + str(name) + ".png")
    plt.ion()
    plt.pause(0.0001)
    plt.clf()
    name += 1
plt.ioff()  # Plot evolution end

# Final Info print
# ============================================================
print("============================================================")
print("Number of nodes: ", N)
print("Number of time steps: ", t_step)
print("Total number of iterations: ", total_iter)
print("External Temp in kelvin (L=0) at final time: ", T_dist[0])
print("Final temperature in Kelvin in x=L/2 at final time: ",
↪  T_dist[int(len(T_dist)/2)])
print("Final temperature in Kelvin in x=L at final time: ",
↪  T_dist[len(domain)-1])
```

```
print("Final time(s): ", t)

# Plot maintain
# ================================================================
plt.figure(2)
plt.imshow((domain, T_dist), aspect='auto', interpolation='gaussian',
↪  cmap='inferno')
plt.autoscale()
plt.colorbar()
plt.xlabel('Control Volumes')
plt.show()

plt.figure(1)
plt.plot(domain, T_dist)
plt.xlabel('Distance (m)')
plt.axis([0, max(domain), min(T_dist), max(T_dist)])
plt.ylabel('Temperature (k)')
plt.show()
```