UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Aerospace Technology Engineering

EXERCISE 4

Numerical solution of the heat equation (2D)

APMC++ to Thermal Engineering Problems

Abril Bertran Garrigos

May 2025

Index

1 Case of study

For this exercise, it is wanted to analyze the following case by means of numerical solution of the heat equation (2D):

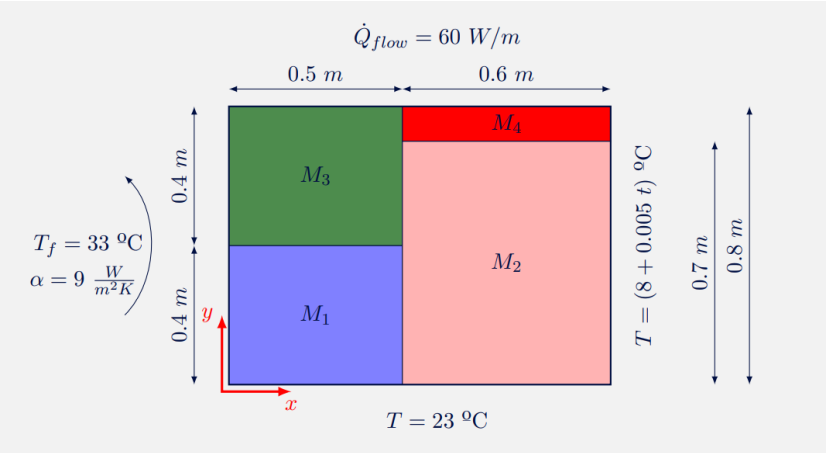

Figure 1: Case of study

It consists of a rod of 4 different materials (M1 to M4) with the following properties:

Material	$\rho\ [kg/m^3]$	$c_p\ [J/(KgK)]$	$\lambda\ [W/(mK)]$
M_1	1500	750	170
M_2	1600	770	140
M_3	1900	810	200
M_4	2500	930	140

Figure 2: Material Properties

The boundary conditions are:

Cavity wall	Boundary condition
Bottom	Isotherm at $T = 23$ ºC
Top	Uniform $\dot{Q}_{flow} = 60$ W/m length
Left	In contact with a fluid at $T_f = 33$ ºC and $\alpha = 9\ W/m^2K$
Right	Uniform temperature $T = 8 + 0.005t$ ºC (where t is the time in seconds)

Figure 3: Boundary Conditions

The initial temperature is stated to be $T(t = 0) = 8°C$ and by using and implicit method it is asked to deliver the following:

- Deliver a file containing 10000 rows and 3 columns. The first column must contain the time in seconds from 0 to 10000 s, the second column the temperature T1 of point (0.65 0.56) at that time, and the third column the temperature T2 of point (0.74 0.72).
- Plot a temperature map for different time instants, or you could generate a video of the temperature map evolution over time.

2 Algorithm for solving 2D heat equation

2.1 Numerical inputs

To solve the problem in 2D it is needed to generate a N_y x N_x mesh where $N_x = 55$ and $N_y = 40$. This mesh has its origin in the bottom left corner, so in MATLAB this will correspond to the box (j=Ny,i=0).In addition, it is selected to use centered nodes to create the mesh, so the corresponding coordinates will follow a logic of the following type: $x = 0.5dx + (i-1)dx$; where dx is the spatial differential. A schematic is shown below for better understanding:

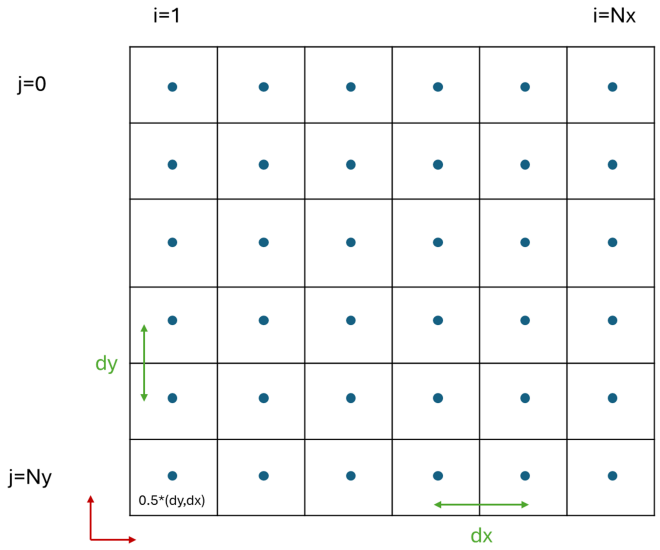

Figure 4: Mesh

Thus, two additional matrices of dimension N_yxN_x have been created for the coordinates of the X and Y nodes. The code used with all the numerical input parameters is shown below:

```
% Mesh
w=1;                        % width of the VC
Lx=1.1;     Ly=0.8;         % Length of each side
Nx=55;      Ny=40;          % Node discretization
dx=Lx/Nx;   dy=Ly/Ny;       % Differentials
ds=dx*w;    dv=dx*dy*w;

x=zeros(Ny,Nx);
y=zeros(Ny,Nx);

dt=1;
tend=10000;                  %time discretization
```

```
time=0:dt:tend; time=time.';
Nt=size(time,1);

% Construction of x and y coordinate node matrices
for i=1:Nx
    x(:,i)=0.5*dx+(i-1)*dx;
end

for i=1:Ny
    y(i,:)=0.5*dy+(i-1)*dy;
end
y=flipud(y);
```

2.2 Physical inputs

For the physical parameters, the corresponding variables have been created, as well as matrices of the same dimensions as the mesh for the different parameters λ, ρ and c_p have been created according to their coordinates xy. The code is shown below:

```
 % Material properties
mat1.rho=1500; mat1.cp=750; mat1.lambda=170;
mat2.rho=1600; mat2.cp=770; mat2.lambda=140;
mat3.rho=1900; mat3.cp=810; mat3.lambda=200;
mat4.rho=2500; mat4.cp=930; mat4.lambda=140;

%Construction of material matrices
rho=zeros(Ny,Nx);
cp=zeros(Ny,Nx);
lambda=zeros(Ny,Nx);

for i=1:Nx
    for j=1:Ny
        if x(j,i)<=0.5 && y(j,i)<=0.4
            mat=mat1;
        elseif x(j,i)>0.5 && y(j,i)<=0.7
            mat=mat2;
        elseif x(j,i)<=0.5 && y(j,i)>0.4
            mat=mat3;
        else
```

```
            mat=mat4;
        end
        rho(j,i)=mat.rho;
        cp(j,i)=mat.cp;
        lambda(j,i)=mat.lambda;
    end
end
```

It should be noted that the boundary conditions have not been entered because they are inputted into an additional function to calculate the temperatures.

2.3 Temperature Function

Para obtener el mapa de temperaturas se ha utilizado la resolucion numerica de la ecuacion de calor general:

$$\rho c_p \frac{T_p^{n+1} - T_p^n}{\Delta t} \Delta V = \lambda e \frac{T_E - T_P}{d_{PE}} \Delta S_e + \lambda w \frac{T_W - T_P}{d_{PW}} \Delta S_w + \lambda n \frac{T_N - T_P}{d_{PN}} \Delta S_n + \lambda s \frac{T_S - T_P}{d_{PS}} \Delta S_s$$

The term on the right has used a simplified notation and has been evaluated at the instant t^{n+1}:

$$T_P \equiv T_P^{n+1} \qquad T_E \equiv T_E^{n+1} \qquad T_W \equiv T_W^{n+1} \qquad T_N \equiv T_N^{n+1} \qquad T_S \equiv T_S^{n+1}$$

Therefore, by rearranging the equation, the coefficients can be obtained for a clearer solution of the system:

$$\left(\rho c_p \frac{\Delta V}{\Delta t} + \lambda e \frac{\Delta S_e}{d_{PE}} + \lambda w \frac{\Delta S_w}{d_{PW}} + \lambda n \frac{\Delta S_n}{d_{PN}} + \lambda s \frac{\Delta S_s}{d_{PS}}\right) T_p^{n+1} = \lambda e \frac{\Delta S_e}{d_{PE}} T_E + \lambda w \frac{\Delta S_w}{d_{PW}} T_W + \lambda n \frac{\Delta S_n}{d_{PN}} T_N + \lambda s \frac{\Delta S_s}{d_{PS}} T_S + \rho c_p \frac{\Delta V}{\Delta t}$$

Considering that the mesh is uniform::

$$a_p T_p = a_e T_E + a_w T_W + a_n T_N + a_s T_S + b$$

$$a_e = \lambda e \frac{\Delta S}{d_P} \qquad a_w = \lambda w \frac{\Delta S}{d_P} \qquad a_n = \lambda n \frac{\Delta S}{d_P} \qquad a_s = \lambda s \frac{\Delta S}{d_P} \qquad b = \rho c_p \frac{\Delta V}{\Delta t} T_p^n$$

$$a_p = \rho c_p \frac{\Delta V}{\Delta t} + \frac{\Delta S}{d_P} (\lambda e + \lambda w + \lambda n + \lambda s) = \rho c_p \frac{\Delta V}{\Delta t} + a_e + a_w + a_n + a_s$$

For the parameter of thermal conductivity λ_f, it must be taken into account when there is a change of material. Therefore, the harmonic mean is used$\lambda_f = \frac{2}{1/\lambda_1 + 1/\lambda_2}$

These coefficients are valid for the interior nodes, but not for the edge ones since the general equation has modifications due to boundary conditions.

Left edge

The border nodes on the left are subject to convective heat transfer. Therefore, the general equation is:

$$\rho c_p \frac{T_p^{n+1} - T_p^n}{\Delta t} \Delta V = \lambda e \frac{T_E - T_P}{d_{PE}} \Delta S_e + \alpha (T_\infty - T_{wall}) \Delta S_w + \lambda n \frac{T_N - T_P}{d_{PN}} \Delta S_n + \lambda s \frac{T_S - T_P}{d_{PS}} \Delta S_s$$

The T_{wall} can be aproximated as $T_{wall} \approx T_p^{n+1}$. Hence, the new coefficients are:

$$a_e = \lambda e \tfrac{\Delta S}{d_P} \qquad a_w = 0 \qquad a_n = \lambda n \tfrac{\Delta S}{d_P} \qquad a_s = \lambda s \tfrac{\Delta S}{d_P} \qquad b = \rho c_p \tfrac{\Delta V}{\Delta t} T_p^n + \alpha \, T_\infty \Delta S$$

$$a_p = \rho c_p \tfrac{\Delta V}{\Delta t} + \tfrac{\Delta S}{d_P} (\lambda e + \lambda n + \lambda s) + \alpha \Delta S = \rho c_p \tfrac{\Delta V}{\Delta t} + a_e + a_n + a_s + \alpha \Delta S$$

Top edge

The top edge is subject to a Neumann condition, so the heat equation for these nodes results in:

$$\rho c_p \frac{T_p^{n+1} - T_p^n}{\Delta t} \Delta V = \lambda e \frac{T_E - T_P}{d_{PE}} \Delta S_e + \lambda w \frac{T_W - T_P}{d_{PW}} \Delta S_w + \dot{Q} \, dP_N + \lambda s \frac{T_S - T_P}{d_{PS}} \Delta S_s$$

Therefore the coefficients obtained for this case are:

$$a_e = \lambda e \tfrac{\Delta S}{d_P} \qquad a_w = \lambda w \tfrac{\Delta S}{d_P} \qquad a_n = 0 \qquad a_s = \lambda s \tfrac{\Delta S}{d_P} \qquad b = \rho c_p \tfrac{\Delta V}{\Delta t} T_p^n \; + \dot{Q} \, dP$$

$$a_p = \rho c_p \tfrac{\Delta V}{\Delta t} + \tfrac{\Delta S}{d_P} (\lambda e + \lambda w + \lambda s) = \rho c_p \tfrac{\Delta V}{\Delta t} + a_e + a_w + a_s$$

Right edge

The border nodes with the right-hand edge have a Dirichlet condition, so the corresponding heat equation is:

$$\rho c_p \frac{T_p^{n+1} - T_p^n}{\Delta t} \Delta V = \lambda e \frac{T_{wall} - T_P}{d_{PE}} \Delta S_e + \lambda w \frac{T_W - T_P}{d_{PW}} \Delta S_w + \lambda n \frac{T_N - T_P}{d_{PN}} \Delta S_n + \lambda s \frac{T_S - T_P}{d_{PS}} \Delta S_s$$

The coefficients obtained are:

$$a_e = 0 \qquad a_w = \lambda w \tfrac{\Delta S}{d_P} \qquad a_n = \lambda n \tfrac{\Delta S}{d_P} \qquad a_s = \lambda s \tfrac{\Delta S}{d_P} \qquad b = \rho c_p \tfrac{\Delta V}{\Delta t} T_p^n \; + \lambda_e \tfrac{T_{wall}}{D_P} \Delta S$$

$$a_p = \rho c_p \tfrac{\Delta V}{\Delta t} + \tfrac{\Delta S}{d_P} (\lambda e + \lambda w + \lambda n + \lambda s) = \rho c_p \tfrac{\Delta V}{\Delta t} + a_e + a_w + a_n + a_s$$

Bottom edge

The border nodes below also have a Dirichlet condition, so they follow the same logic as those on the right-hand edge. The coefficients are:

$$a_e = \lambda e \tfrac{\Delta S}{d_P} \qquad a_w = \lambda w \tfrac{\Delta S}{d_P} \qquad a_n = \lambda n \tfrac{\Delta S}{d_P} \qquad a_s = 0 \qquad b = \rho c_p \tfrac{\Delta V}{\Delta t} T_p^n \; + \lambda_S \tfrac{T_{wall}}{D_P} \Delta S$$

$$a_p = \rho c_p \tfrac{\Delta V}{\Delta t} + \tfrac{\Delta S}{d_P} (\lambda e + \lambda w + \lambda n + \lambda s) = \rho c_p \tfrac{\Delta V}{\Delta t} + a_e + a_w + a_n + a_s$$

The corner nodes must also be modified and turn out to be a superposition of the cases already mentioned. The function resultant to get the new temperature is:

```
function [Tnew]=getTemperature(i,j,Nx,Ny,lambda,rho,cp,dx,dy,ds,dv,dt,t,T,Tp)

%% PARAMETERS DEFINITION
% Boundary conditions
alpha=9;
Tbottom=23;
Tright=8+0.005*t*dt;
Tleft=33;
Qtop=60;

% Parameters for actual node P
rhoP=rho(j,i);
cpP=cp(j,i);

% Calculation of lambdas and harmonic mean
if i>1, lambW=1/(1/lambda(j,i)+1/lambda(j,i-1)); else, lambW=lambda(j,i); end
if i<Nx, lambE=1/(1/lambda(j,i)+1/lambda(j,i+1)); else, lambE=lambda(j,i); end
if j>1, lambN=1/(1/lambda(j,i)+1/lambda(j-1,i)); else, lambN=lambda(j,i); end
if j<Ny, lambS=1/(1/lambda(j,i)+1/lambda(j+1,i)); else, lambS=lambda(j,i); end



%% COEFFICIENTS GENERAL CASE ( valid for all internal nodes)
ae=lambE*ds/dx;
aw=lambW*ds/dx;
an=lambN*ds/dy;
as=lambS*ds/dy;
ap=rhoP*cpP*dv/dt+(ae+aw+an+as);
b=(rhoP*cpP*dv/dt)*Tp(j,i);


%% CASE STUDIES

% Left bottom corner  ---> aw=as=0
if i==1 && j==Ny
    ap=ap-aw+alpha*ds;
    b=b+alpha*Tleft*ds+lambS*ds/dy*Tbottom;
```

```
        Tnew=1/ap*(ae*T(j,i+1)+an*T(j-1,i)+b);

    % Right bottom corner --> ae=as=0
    elseif i==Nx && j==Ny
        b=b+lambE*ds/dx*Tright+lambS*ds/dy*Tbottom;
        Tnew=1/ap*(aw*T(j,i-1)+an*T(j-1,i)+b);

    % Left upper corner --> aw=an=0
    elseif i==1 && j==1
        ap=ap-aw-an+alpha*ds;
        b=b+Qtop*dx+alpha*Tleft*ds;
        Tnew=1/ap*(ae*T(j,i+1)+as*T(j+1,i)+b);

    % Right upper corner --> ae=an=0
    elseif i==Nx && j==1
        ap=ap-an;
        b=b+Qtop*dx+lambE*ds/dx*Tright;
        Tnew=1/ap*(aw*T(j,i-1)+as*T(j+1,i)+b);

    % Left side nodes --> aw=0
    elseif i==1
        ap=ap-aw+alpha*ds;
        b=b+alpha*Tleft*ds;
        Tnew=1/ap*(ae*T(j,i+1)+an*T(j-1,i)+as*T(j+1,i)+b);

    % Right side nodes --> ae=0
    elseif i==Nx
        b=b+lambE*ds/dx*Tright;
        Tnew=1/ap*(aw*T(j,i-1)+an*T(j-1,i)+as*T(j+1,i)+b);

    % Bottom side nodes --> as=0
    elseif j==Ny
        b=b+lambS*ds/dy*Tbottom;
        Tnew=1/ap*(ae*T(j,i+1)+aw*T(j,i-1)+an*T(j-1,i)+b);

    % Upper side nodes --> an=0
    elseif j==1
        ap=ap-an;
```

```
        b=b+Qtop*dx;
        Tnew=1/ap*(ae*T(j,i+1)+aw*T(j,i-1)+as*T(j+1,i)+b);

    % Internal nodes
    else
        Tnew=1/ap*(ae*T(j,i+1)+aw*T(j,i-1)+an*T(j-1,i)+as*T(j+1,i)+b);
    end

end
```

2.4 Iterative loop

Now it can be applied in the iterative loop through Gauss Seidel. Therefore, to store the data is used a matrix of 3 dimensions (Ny,Nx,t) to save the temperature map of every timestep.

```
epsilon=10^-6;

% Time loop to get T^n+1
for t=1:Nt-1
    err=1;

    % Iterative loop ensuring convergence
    while err>epsilon

        % Calculation to obtain T^n+1
        for j=Ny:-1:1
            for i=1:Nx
                T(j,i)=getTemperature(i,j,Nx,Ny,lambda,rho,cp,dx,dy,ds,dv,dt,t,T,Tp);
            end
        end
        err=max(max(abs(T-T_prev))); % Error calculation
        T_prev=T;               % Saves the previous results for next iteration
    end
    T_all(:,:,t+1)=T;
    Tp=T;                % Updates the temperature of T^n for the next timestep
        iteration

end
```

2.5 Data exportation

```
% Obtain the matrix index for the corresponding coordinates
x1=0.65; y1=0.56;
x2=0.74; y2=0.72;

[~, idx_x1] = min(abs(x(1,:) - x1));
[~, idx_y1] = min(abs(y(:,1) - y1));
[~, idx_x2] = min(abs(x(1,:) - x2));
[~, idx_y2] = min(abs(y(:,1) - y2));

% Obtain temperature vectors
T1=T_all(idx_y1,idx_x1,:); T1=T1(:);
T2=T_all(idx_y2,idx_x2,:); T2=T2(:);



% Arrange and export data
data=[time,T1,T2];
dlmwrite('temperaturas.txt', data, 'delimiter', '\t');
```

2.6 Completed code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   EXERCISE 3 - APPLICATIONS OF MATLAB TO THERMAL PROBLEMS
%   Ex 4: Option1 - 4 materials
%   Abril Bertran
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

format long
clc;
clear;

%% Numerical Properties

% Mesh
w=1;                        % width of the VC
Lx=1.1;     Ly=0.8;         % Length of each side
Nx=55;      Ny=40;          % Node discretization
```

```
dx=Lx/Nx;   dy=Ly/Ny;       % Differentials
ds=dx*w;    dv=dx*dy*w;

x=zeros(Ny,Nx);
y=zeros(Ny,Nx);

dt=1;
tend=10000;                  %time discretization
time=0:dt:tend; time=time.';

Nt=size(time,1);

% Construction of x and y coordinate node matrices
for i=1:Nx
    x(:,i)=0.5*dx+(i-1)*dx;
end

for i=1:Ny
    y(i,:)=0.5*dy+(i-1)*dy;
end
y=flipud(y);

%% Physical Properties

% Material properties
mat1.rho=1500; mat1.cp=750; mat1.lambda=170;
mat2.rho=1600; mat2.cp=770; mat2.lambda=140;
mat3.rho=1900; mat3.cp=810; mat3.lambda=200;
mat4.rho=2500; mat4.cp=930; mat4.lambda=140;

%Construction of material matrices
rho=zeros(Ny,Nx);
cp=zeros(Ny,Nx);
lambda=zeros(Ny,Nx);

for i=1:Nx
    for j=1:Ny
        if x(j,i)<=0.5 && y(j,i)<=0.4
```

```
            mat=mat1;
        elseif x(j,i)>0.5 && y(j,i)<=0.7
            mat=mat2;
        elseif x(j,i)<=0.5 && y(j,i)>0.4
            mat=mat3;
        else
            mat=mat4;
        end
        rho(j,i)=mat.rho;
        cp(j,i)=mat.cp;
        lambda(j,i)=mat.lambda;
    end
end


%% Matrices inicialization to save the temperatures
T_all=zeros(Ny,Nx,Nt);              % Matrix temperature to save all data
T_all(:,:,1)=ones(Ny,Nx)*8;         % Temperature at t=0;
T=zeros(Ny,Nx);                     % Matrix temperature

Tp=T_all(:,:,1);                    % Temperature at T^n
T_prev=Tp;                          % Initial estimation for the first iteration


%% Gauss Seidel loop

epsilon=10^-6;

% Time loop to get T^n+1
for t=1:Nt-1
    err=1;

    % Iterative loop ensuring convergence
    while err>epsilon

        % Calculation to obtain T^n+1
        for j=Ny:-1:1
            for i=1:Nx
```

```
                T(j,i)=getTemperature(i,j,Nx,Ny,lambda,rho,cp,dx,dy,ds,dv,dt,t,T,Tp);
            end
        end
        err=max(max(abs(T-T_prev))); % Error calculation
        T_prev=T;               % Saves the previous results for next iteration
    end
    T_all(:,:,t+1)=T;
    Tp=T;                 % Updates the temperature of T^n for the next timestep
        iteration

end


%% Export data

% Obtain the matrix index for the corresponding coordinates
x1=0.65; y1=0.56;
x2=0.74; y2=0.72;

[~, idx_x1] = min(abs(x(1,:) - x1));
[~, idx_y1] = min(abs(y(:,1) - y1));
[~, idx_x2] = min(abs(x(1,:) - x2));
[~, idx_y2] = min(abs(y(:,1) - y2));

% Obtain temperature vectors
T1=T_all(idx_y1,idx_x1,:); T1=T1(:);
T2=T_all(idx_y2,idx_x2,:); T2=T2(:);


% Arrange and export data
data=[time,T1,T2];
dlmwrite('temperaturas.txt', data, 'delimiter', '\t');


%% plot the isotherms for diferent timesteps
t1=2000;
t2= 3000;
t3= 4000;
```

```
t4= 5000;

[~, idx_t1] = min(abs(time - t1));
[~, idx_t2] = min(abs(time - t2));
[~, idx_t3] = min(abs(time - t3));
[~, idx_t4] = min(abs(time - t4));

figure
subplot(2, 2, 1);
contourf(x,y,T_all(:,:,idx_t1),'ShowText','on');
set(gca, 'YDir', 'normal');
axis equal;
title('t=2000s');

subplot(2, 2, 2);
contourf(x,y,T_all(:,:,idx_t2),'ShowText','on');
set(gca, 'YDir', 'normal');
axis equal;
title('t=3000s')

subplot(2, 2, 3);
contourf(x,y,T_all(:,:,idx_t3),'ShowText','on');
set(gca, 'YDir', 'normal');
axis equal;
title('t=4000s')

subplot(2, 2, 4);
contourf(x,y,T_all(:,:,idx_t4),'ShowText','on');
set(gca, 'YDir', 'normal');
title('t=5000s')
axis equal;

%% Video Animation

if exist('temperatura_video.mp4','file')
    delete('temperatura_video.mp4');
end

```

```
v = VideoWriter('temperatura_video.mp4', 'MPEG-4');
v.FrameRate = 40;
open(v);

figure('Name','Animation of temprature along time')
ax = axes;
colorbar;
xlabel('x [m]');
ylabel('y [m]');

for t_idx = 2:32:Nt
    contourf(ax, x, y, T_all(:,:,t_idx),'ShowText','on');
    title(['Time: ', num2str(time(t_idx)), ' s']);
    drawnow;
    frame = getframe(gcf);
    writeVideo(v, frame);
end

close(v);
```

3 Results and conclusions

Once the code has been executed, the temperature isotherm graphs have been obtained for given times.

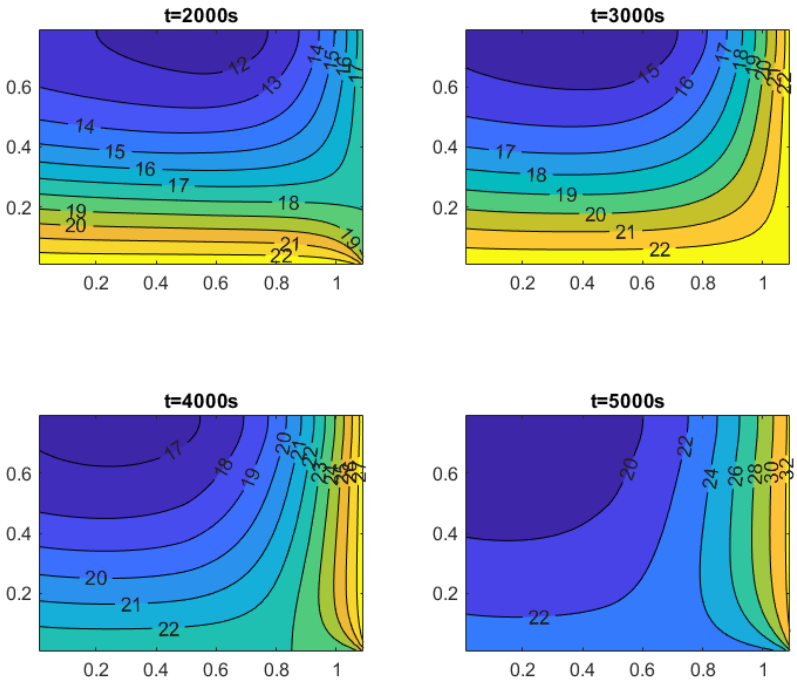

Figure 5: Isotherms of temperature for different time instants

On the other hand, the temperatures T1 of the point (0.65 0.56) and T2 (0.74 0.72) have also been exported in a text file with three columns. It can be accessed by clicking here as well as it is located in the same folder.

Some of the results are shown below:

Time [s]	T1 [ºC]	T2 [ºC]
0	8	8
1000	9.691	8.98
2000	12.716	11.953
3000	15.657	15.292
4000	18.466	18.578
5000	21.153	21.762

Table 1: Results of the temperatures T1 and T2 for different time instants

A video showing the temporal evolution of the isotherms has also been created. It is located in the folder and can also be accessed via the following link

As can be seen from the results, they show similarities with the samples given by the teacher. The isotherms follow the same pattern and have temperatures close to the expected ones, even though there are some differences that may be due to the error and precision of the iterations. As for the exported data, T1 and T2 are also close to what they should be, although there is a slight difference.
As a conclusion, it could be tried to improve the results by refining the mesh, as well as decreasing the time increments and the allowed error.