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

Aerospace Technology Engineering

EXERCISE 1

1D transient conduction case

APMC++ to Thermal Engineering Problems

Abril Bertran Garrigos

May 2025

Index

1 Case of study

For this exercise, the main objective is to analyse the heat behaviour of a copper bar. Therefore, the problem will be simplified in a 1D transient conduct case, as is shown in the following image:

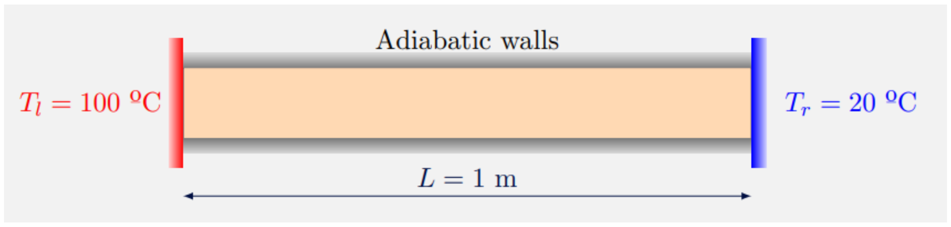


Figure 1: caption

The walls are considered adiabatic and the length of the tube is of 1m, while the boundary conditions consists of a temperature of $T_l = 100$ °C at the left side and $T_r = 20$ °C at the right side.

The copper properties are:

$$\lambda = 400\, W/(m \cdot K) \qquad \rho = 8960\, kg/m^3 \qquad c_p = 380\, J/(kg \cdot K)$$

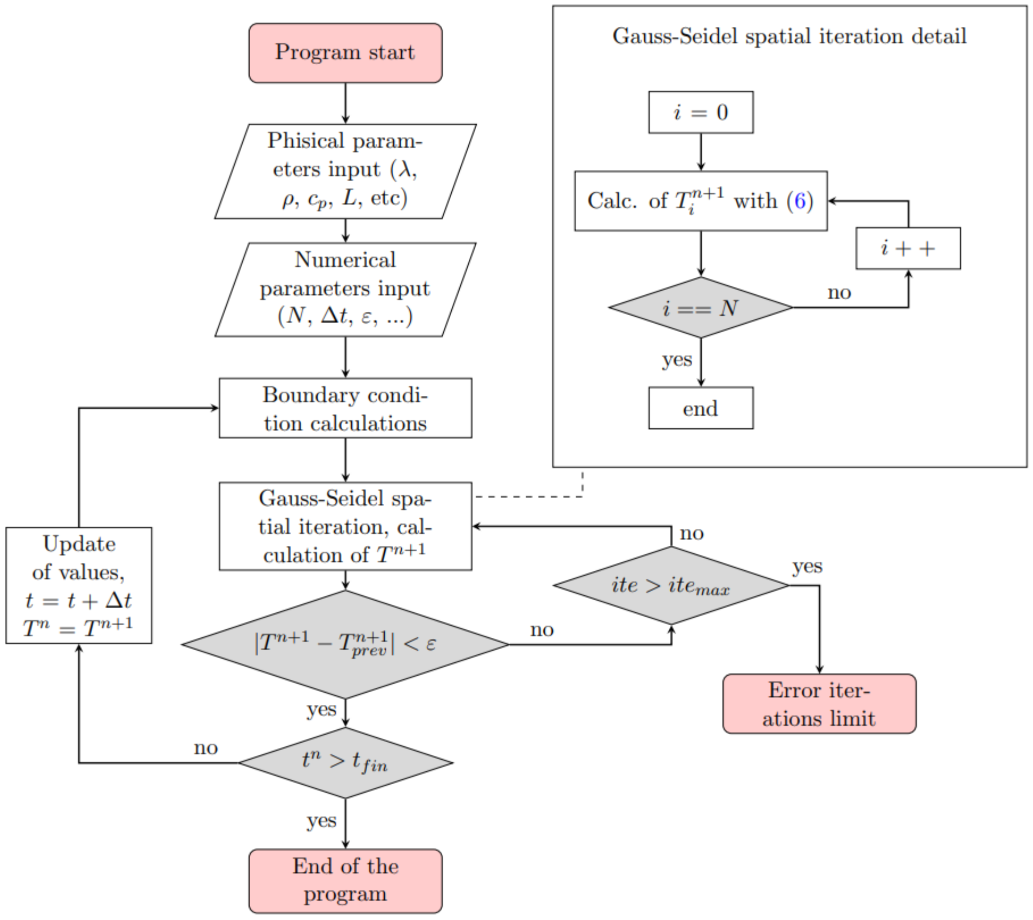
The exercise must be solved by numerical methods, therefore it will be implemented a MATLAB script. Moreover, it is asked to use an interative method like Gauss-Seidel and with the Crank-Nicholson scheme ($\beta = 0.5$).

The results that will be delivered are:

1. Plot of the temperature of the point at x = 0.5 m over time until 2000 s.
2. Plot of the temperature profile as a function of x at t = 800 s
3. The numerical value of T (x = 0.75 m, t = 600 s)
4. Different Δt and temporal schemes (Explicit, Implicit and CN) comparing the accuracy, convergence, etc.

2 Algorithm for solving 1D conduction cases

To solve the problem, the algorithm that is followed has a similar diagram to the presented one:



Program start
Phisical parameters input (λ, ρ, c_p, L, etc)
Numerical parameters input (N, Δt, ε, ...)
Boundary condition calculations
Gauss-Seidel spatial iteration, calculation of T^{n+1}
$|T^{n+1} - T^{n+1}_{prev}| < \varepsilon$
$ite > ite_{max}$
Error iterations limit
$t^n > t_{fin}$
Update of values, $t = t + \Delta t$ $T^n = T^{n+1}$
End of the program
yes
no
Gauss-Seidel spatial iteration detail
$i = 0$
Calc. of T_i^{n+1} with (6)
$i == N$
$i++$
end

Figure 2: Algorithm Diagram

2.1 Physical parameters

The first step is to input the physical parameters mentioned before:

```
L= 1;                       %m
lambda= 400;                %W/(m*K)
rho= 8960;                  %kg/m^3
cp= 380;                    %J/(kg*K)
alpha=lambda/(rho*cp);
```

2.2 Numerical Parameters

Secondly, it is needed to enter the numerical parameters. To discretize the bar, the number of nodes selected is $N = 100$, and the distance x will be stored in a vector. Also, a constant distance step will be used and can be obtained as $dx = \frac{L}{N} = 0.01$, hence the code will be:

```
N=100;                                      %num of nodes
dx= L/N;                                    %step x
vec_x=dx/2:dx:L-dx/2;;                      %discretization of the bar
```

To ensure convergence, it is important to determine a proper time step that satisfies the following condition:

$$\gamma = \alpha \frac{\Delta t}{\Delta x^2} \leq 0.5 \tag{1}$$

Rearranging the terms:

$$\Delta t \leq 0.5 \frac{\Delta x^2}{\alpha} = 0.4256$$

Therefore, the time step is $\Delta t = 0.25s$. As it happened with the distance coordinates, a vector is created corresponding to each instant.

```
dt= 0.25;                                   %time step
tend=2000;                                  %final time
vec_t=0:dt:tend;                            %vector of time steps
gamma= alpha*dt/dx^2;
```

Finally, to generalize the code for the different schemes, the beta parameter is introduced, and a matrix is created to store the temperatures where each row corresponds to a node and each column to a time step.

```
    T= zeros(size(vec_x,2),size(vec_t,2));    %matrix of temperatures
    beta=0.5;
```

2.3 Boundary conditions and data inicialization

In this step of the code, the boundary conditions of the problem must be determined and introduced in the temperature matrix. Therefore, the nodes at the extremes (i=1 and i=N) will have the temperatures T_l and T_r correspondingly at any instant of time. On the other hand, the internal nodes for the initial instant will have the temperature T_0.

```
T0=30;
Tl=100;
Tr=20;
T(:,1)=T0;      %All the nodes at the initial instant is T0
T(1,:)=Tl;      %Boundary condition of the first node at all instant =Tl
T(end,:)=Tr;    %Boundary condition of the last node at all instant =Tr
```

Here is a scheme of the data model storage in the matrix, where the i corresponds to the index of the nodes while the n to the time.

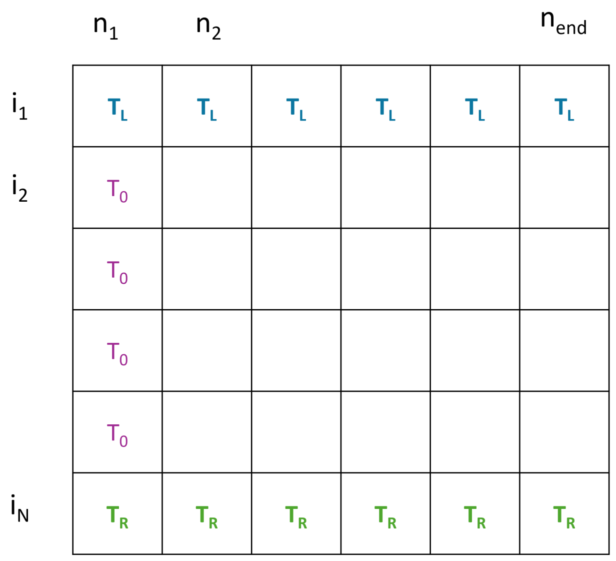


Figure 3: Matrix Initialization

2.4 Gauss-Seidel loop

For the Gauss-Seidel method, it is needed to iterate the solution of all the nodes of the bar for each concrete time instant.

The values that will be calculated and iterated are T_i^{n+1} with the formula given:

$$T_i^{n+1} = \frac{1}{1+2\gamma\beta}\left(T_i^n + \gamma\left[\beta\left(T_{i+1}^{n+1} + T_{i-1}^{n+1}\right) + (1-\beta)\left(T_{i+1}^n - 2T_i^n + T_{i-1}^n\right)\right]\right) \tag{2}$$

However, the formula will be rewritten with a different notation to understand better the code:

$$a_p T_p = a_E T_E + a_W T_W + b_p \tag{3}$$

Where

$$T_p = T_i^{n+1} \qquad T_W = T_{i-1}^{n+1} \qquad T_W = T_{i+1}^{n+1}$$

Therefore, rearranging the equation (2):

$$a_p = 1 + 2\gamma\beta \qquad a_E = \gamma\beta \qquad a_B = \gamma\beta$$

$$b_p = T_i^n + \gamma\,(1-\beta)\left[\,T_{i+1}^n - 2\,T_i^n + T_{i-1}^n\right]$$

By this way, it is first estimated the values of T^{n+1} and then iterated with the above-mentioned equation(3) until the error is lower than the tolerance established. The error is the absolute value of the difference between the new value obtained and the previous one.

$$error = \left|T^{n+1} - T^{n+1}_{prev}\right| \tag{4}$$

Also, a counter of the iterations has been added to check that the code has some coherence and is working correctly.

```
tol=1e-6;
contador= zeros(size(vec_t,2),1);
error= zeros(size(vec_t,2),1);

for n=1:size(T,2)-1
    error(n)=10;

    %Initial estimation
    T(:,n+1)=T(:,n);

    while error(n)>tol
        Told=T(:,n+1);
        for i=2:size(T,1)-1
            Tp_guess=T(i,n+1);
            Te=T(i+1,n+1);
            Tw=T(i-1,n+1);

            %Calcul coeficients
            ap=1+2*gamma*beta;
            ae=gamma*beta;
            aw=gamma*beta;
            bp=T(i,n)+gamma*(1-beta)*(T(i+1,n)-2*T(i,n)+T(i-1,n));

            %Calcul temperatura
            T(i,n+1)=1/ap*(ae*Te+aw*Tw+bp);
        end
        error(n)= max(abs(Told-T(:,n+1)));
        contador(n)=contador(n)+1;
    end
end
```

2.5 Analytical Solution

To get the analytical solution, it is needed to non-dimensionalize the variables:

$$t^* = t\frac{\alpha}{L^2} \qquad x^* = \frac{x}{L} \qquad u^* = \frac{T - T_{left}}{T_{right} - T_{left}}$$

Where also the conditions are:

$$u(x,0) = u_0 = \frac{T_0 - T_{left}}{T_{right} - T_{left}}$$
$$u(0,t) = 0$$
$$u(1,t) = u_1 = 1$$

Hence, the vectors of time and space used in the code are transformed, as well as a vector u is initialized with the conditions calculated:

```
vec_t=0:dt:tend;
x_adim=x/L;
t_adim= vec_t.'*(alpha/L^2);
u0=(T0-Tl)/(Tr-Tl);
u1=1;

u=zeros(size(vec_t,2),1);
u=1*x_adim+u;               %Calculation of the initial term
```

Hence, the solution of the PDE is:

$$u(x^*,t^*) = u_1x^* + \frac{4u_0}{\pi}\sum_{n=1,3,5,\ldots}^{\infty}\left(\frac{\sin(n\pi x^*)}{n}e^{-n^2\pi^2t^*}\right) + \frac{2u_1}{\pi}\sum_{n=1}^{\infty}\left((-1)^n\frac{\sin(n\pi x^*)}{n}e^{-n^2\pi^2t^*}\right) \quad (5)$$

In order to obtain the summation terms, a loop has been made for each one independently and then all the terms have been summed as shown below:

```
% Sum of impair index
sum1=zeros(size(vec_t,2),1);
for i=1:2:100
    sum1=sum1+sin(i*pi*x_adim)/i*exp(-i^2*pi^2.*t_adim);
end
%Sum of all index
sum2=zeros(size(vec_t,2),1);
for i=1:100
    sum2=sum2+(-1)^i*sin(i*pi*x_adim)/i*exp(-i^2*pi^2.*t_adim);
end

u=u+4*u0/pi*sum1+2*u1/pi*sum2;
```

And finally it is transformed back the result the temperatures to ºC where:

$$T = T_{left} + u \cdot (T_{right} - T_{left})$$

```
T_analytical=Tl+u*(Tr-Tl);
```

For then compare the code for different dt at a given point x it is decided to put this part of the code as a function where the inputs are the dt,x and final time t_{end}. while the output is the time and temperature vectors:

```
function [T_analytical,vec_t]=analytical(dt,tend,x)

% INPUT PHYSICAL PARAMETERS
L= 1;                       %m
lambda= 400;                %W/(m*K)
rho= 8960;                  %kg/m^3
cp= 380;                    %J/(kg*K)
alpha=lambda/(rho*cp);

% BOUNDARY CONDITIONS
T0=30;        %C
Tl=100;       %C
Tr=20;        %C



% ANALYTICAL SOLUTION

% Calculation of the temperature at given x along the time
vec_t=0:dt:tend;
x_adim=x/L;
t_adim= vec_t.'*(alpha/L^2);
u0=(T0-Tl)/(Tr-Tl);
u1=1;

u=zeros(size(vec_t,2),1);
u=1*x_adim+u;                 %Calculation of the initial term



% Sum of impair index
sum1=zeros(size(vec_t,2),1);
```

```
    for i=1:2:100
        sum1=sum1+sin(i*pi*x_adim)/i*exp(-i^2*pi^2.*t_adim);
    end

    %Sum of all index
    sum2=zeros(size(vec_t,2),1);
    for i=1:100
        sum2=sum2+(-1)^i*sin(i*pi*x_adim)/i*exp(-i^2*pi^2.*t_adim);
    end

    u=u+4*u0/pi*sum1+2*u1/pi*sum2;
    T_analytical=Tl+u*(Tr-Tl);

end
```

2.6 Convergence Study

To study the convergence for different time steps as well as different temporal schemes, it is implemented a function of the Gauss-Seidel method developed before where the inputs are the β, Δt and the position x and time t to evaluate and compare the values.

```
    function [Tnum]=solver(dt,beta,x,t)
    L= 1;                        %m
    lambda= 400;                 %W/(m*K)
    rho= 8960;                   %kg/m^3
    cp= 380;                     %J/(kg*K)
    alpha=lambda/(rho*cp);

    % INPUT NUMERICAL PARAMETERS

    N=100;                                        %num of nodes
    dx= L/N;                                      %step x
    vec_x=dx/2:dx:L-dx/2;;                        %discretization of the bar
    vec_t=0:dt:800;                              %vector of time steps
    T= zeros(size(vec_x,2),size(vec_t,2));        %matrix of temperatures

    gamma= alpha*dt/dx^2;

    % BOUNDARY CONDITIONS
```

```
T0=30;        %C
Tl=100;       %C
Tr=20;        %C

% DATA INITIALIZATION
T(:,1)=T0;      %All the nodes at the initial instant is T0
T(1,:)=Tl;      %Boundary condition of the first node at all instant =Tl
T(end,:)=Tr;    %Boundary condition of the last node at all instant =Tr


% GAUSS - SEIDEL
tol=1e-13;

err= zeros(size(vec_t,2),1);

for n=1:size(T,2)-1
    err(n)=10;

    %Initial estimation
    T(:,n+1)=T(:,n);

    while err(n)>tol
        Told=T(:,n+1);

        for i=2:size(T,1)-1
            Te=T(i+1,n+1);
            Tw=T(i-1,n+1);

            %Calcul coeficients
            ap=1+2*gamma*beta;
            ae=gamma*beta;
            aw=gamma*beta;
            bp=T(i,n)+gamma*(1-beta)*(T(i+1,n)-2*T(i,n)+T(i-1,n));

            %Calcul temperatura
            T(i,n+1)=1/ap*(ae*Te+aw*Tw+bp);
        end

```

```
            err(n)= max(abs(Told-T(:,n+1)));
        end
    end

    [~, idx1] = min(abs(vec_x - x));
    [~, idx2] = min(abs(vec_t - t));
    Tnum=T(idx1,idx2);

end
```

Then a vector of different timesteps is created, as well as it is obtained the analytical value with the function of section 2.5. Then a loop is created where the numerical values for each value of time step is calculated as well as the different type of errors with the formula:

$$\varepsilon = |\varepsilon_{\Delta x} - \varepsilon_{\Delta t}|$$

Where the spatial discretization error is defined as:

$$\varepsilon_{\Delta x} = |T_{numeric}^{\Delta t \to 0} - T_{analytic}|$$

While the time discretization error is:

$$\varepsilon_{\Delta x} = |T_{numeric} - T_{numeric}^{\Delta t \to 0}|$$

Therefore the code is as follows:

```
%% Numerical Temperatures
vec_dt=linspace(0.001,10,60);     %vector to iterate diferent timesteps
vec_dt1=linspace(0.001,0.4,25);   %vector to iterate diferent timesteps for explicit
    method
x=0.75;
tend=600;

% Calc of the analytical solution
for i=1:size(vec_dt,2)
    [T_analytical,~]=analytical(vec_dt(i),tend,x);
end

% Explicit beta=0
[Tnum0]=solver(10^-4,0,0.75,tend);

err_dt1=zeros(size(vec_dt1,2),1);
```

```matlab
err_dx1=ones(size(vec_dt1,2),1)*abs(Tnum0-T_analytical(end));

for i=1:size(vec_dt1,2)
    [Tnum]=solver(vec_dt1(i),0,x,tend);
    err_dt1(i)=abs(Tnum0-Tnum);
end



% Implicit beta= 1
[Tnum0]=solver(10^-4,1,0.75,tend);

err_dt2=zeros(size(vec_dt,2),1);
err_dx2=ones(size(vec_dt,2),1)*abs(Tnum0-T_analytical(end));

for i=1:size(vec_dt,2)
    [Tnum]=solver(vec_dt(i),1,x,tend);
    err_dt2(i)=abs(Tnum0-Tnum);
end

% Crank-Nickolson beta= 0.5
[Tnum0]=solver(10^-4,0.5,0.75,tend);

err_dt3=zeros(size(vec_dt,2),1);
err_dx3=ones(size(vec_dt,2),1)*abs(Tnum0-T_analytical(end));

for i=1:size(vec_dt,2)
    [Tnum]=solver(vec_dt(i),0.5,x,tend);
    err_dt3(i)=abs(Tnum0-Tnum);
end
```

3 Solutions and plots

3.1 Plot for x=0.5 over time until t=2000s

The problem asked to plot the node corresponding to x=0.5 over time up to 2000s, as well as the one along the bar for t=800s.

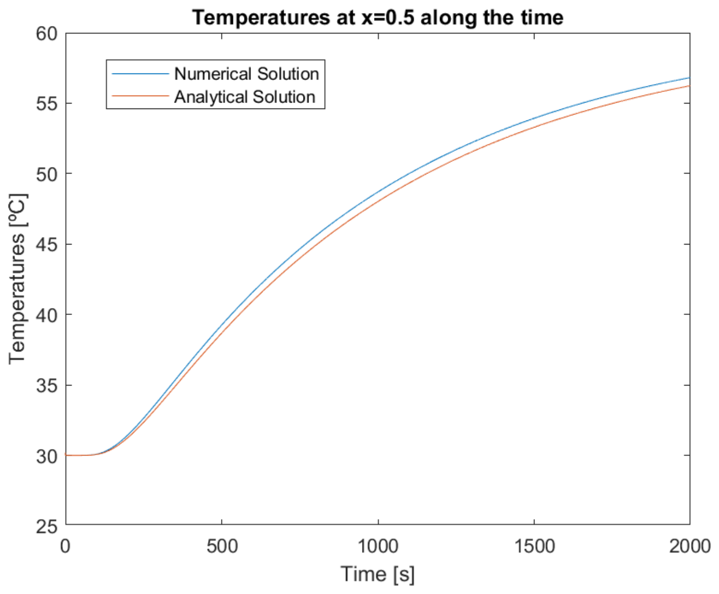


Figure 4: Numerical and analytical solutions for x=0.5 along the time

3.2 Plot of temperature over x at t=800s

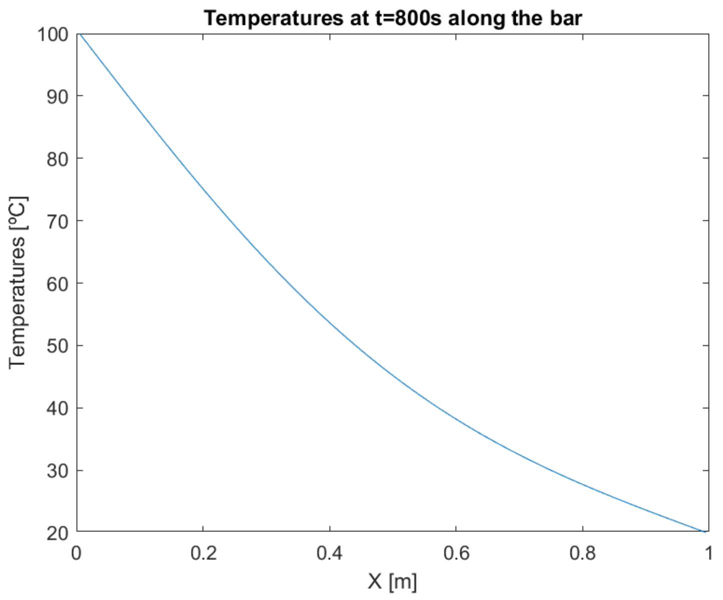


Figure 5: Numerical solution over x at t=800s

3.3 Numerical value of T at x=0.75m and t=600s

It is asked to obtain the numerical value of the temperature at x=0.75 and t=600 s. The value is:

$$T_{numerical}(x = 0.75m, t = 600s) = \boxed{28.29\,C}$$

As for the analytical solution, the result is:

$$T_{analytical}(x = 0.75m, t = 600s) = \boxed{28.09\,C}$$

3.4 Convergence Plots

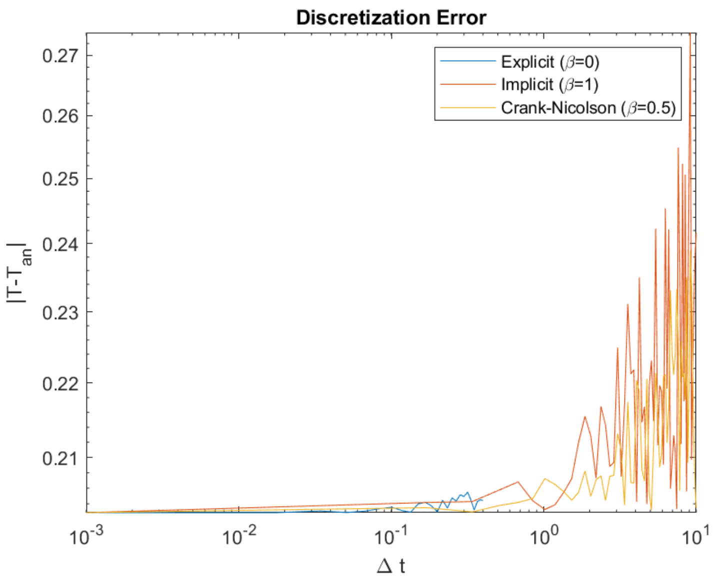


Figure 6: Error for different Δt at x=0.75 and t=600

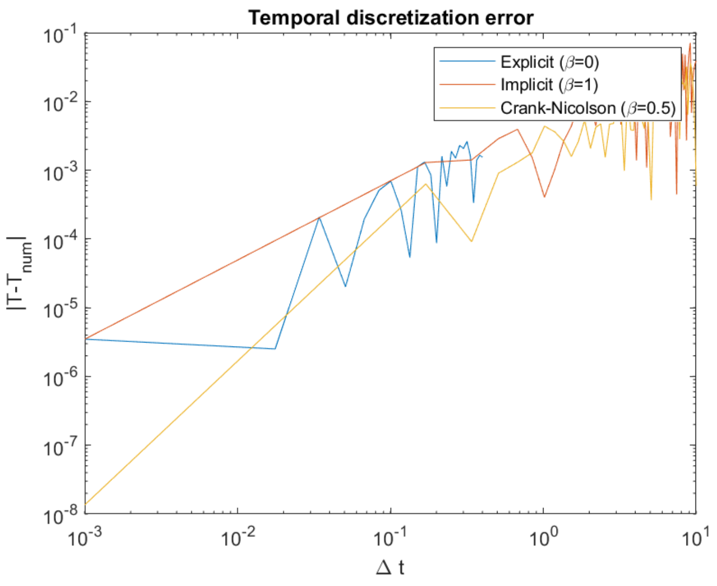


Figure 7: Temporal discretization error for different Δt at x=0.75 and t=600

3.5 Discussion and conclusions

These graphs are consistent with the problem data where they present a continuous solution.
In the first plot (fig:4), it is seen that at t=0 the initial condition of the temperature is 30 degrees is fulfilled, but it progressively increases given that at one of the borders there is a temperature of 100ºC and at the other 20ºC. Knowing that the heat goes through the direction from the highest to the lowest temperature, it can be seen how the temperature in the middle of the bar increases until it stabilizes to find a new equilibrium. This stability can be seen in the last instants of time, since the positive slope is reduced.
Also, it is compared the solution with the analytical solution and both present similar form with an error that increases over time, this could happen because of an error that is accumulated, so the results will be less precise when the time is longer.
The second graph (fig:5) also makes physical sense as it shows how the temperature decreases along the bar given the boundary conditions. At the first node it has a temperature of 100ºC while at the last one 20ºC and following the argument that heat moves from higher to lower, the profile shown can be appreciated.

Regarding the numerical result of the third section (3.3), it presents a similar value to the theoretical one with a small error, also being near the end of the bar, on the right, it makes sense that its value is close to 30 (T_r).

Finally, as for the study of the convergence, it can be seen in the plot (fig.6) that for small time steps until the order of the 10^-1 the error is minimum and the solution converges. However, for time steps higher, the error increases significantly, as well as the explicit method is instable. If it is wanted more precision, a possible solution would be to increase the number of nodes, so the spatial discretization error would be minimized.
In the second plot (fig.7), it can be observed the temporal discretization error. It is observed that for the C-N, the error is the lowest in almost the entire range for low Δt, making it the most accurate scheme. As well as the implicit method presents a much more stable behavior compared to the explicit one which has more oscillations. For $\beta = 1$ and $\beta = 0.5$ it can be seen some kind of linear behavior for a range of small Δt, the slope of the explicit one is much less pronounced while the Crank-Nicholson is higher as expected, as well as it results in a more precise method with the lowest error.