University of Guanajuato
Finite Volume Method (Project III)

García Sánchez Marco Antonio[1]
[1]Division of Engineering, Irapuato-Salamanca Campus, University of Guanajuato, Salamanca, Gto.

Lecturer: Damián Ascencio César Eduardo

November 2019

Resumen

The Finite Volume method is presented to analyze the temporal evolution of the temperature distribution. Different discretization schemes are compared: explicit method, implicit method, and Crank–Nicolson (theoretical), and the implicit and explicit schemes are used to numerically solve the obtained equations.

1. Introduction

The Finite Volume method allows for the discretization and numerical solution of differential equations. It is an alternative method to finite difference and finite element methods. A discretization mesh of the domain is considered. Around each point of this mesh, a control volume is constructed that does not overlap with the volumes of neighboring points. Thus, the total volume of the domain is equal to the sum of the considered control volumes.
The differential equation to be solved is integrated over each control volume, which results in a discretized version of said equation. To perform the integration, it is necessary to specify variation profiles of the dependent variable between the mesh points to evaluate the resulting integrals. The main property of the resulting system of discretized equations is that the obtained solution exactly satisfies the considered conservation equations, regardless of the mesh size. This applies to both steady-state and transient problems.
The steady state is one in which there are no changes in the process variables. In the transient state, the process variables change over time due to the existence of energy or matter accumulation.

2. Governing Equations

A scheme along with the data of the problem to be addressed is shown in Figure 1.
The governing equation is given by the following expression (one-dimensional thermal diffusion equation):

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2}, \tag{1}$$

where T is the temperature, t the time, and D the thermal diffusivity (constant in this work).

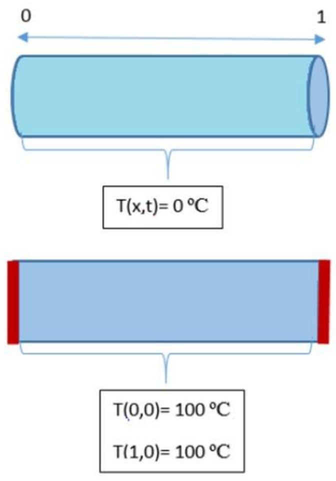


Figure 1. Problem scheme.

3. Discretization

Discretizing the governing equation is obtained by integrating over the control volume and the time interval $[t^n, t^{n+1}]$:

$$\int_V \int_{t^n}^{t^{n+1}} \frac{\partial T}{\partial t}\, dt\, dV = D \int_{t^n}^{t^{n+1}} \left(\int_A \frac{\partial T}{\partial x}\, dA \right) dt.$$

To unify explicit, implicit, and mixed schemes, the parameter $\Theta \in [0,1]$ is introduced, such that:

$$\Theta = 0 \Rightarrow \text{explicit scheme}, \qquad \Theta = 1 \Rightarrow \text{implicit scheme}, \qquad \Theta = \tfrac{1}{2} \Rightarrow \text{Crank–Nicolson}.$$

Assuming a uniform mesh with spatial step Δx and temporal step Δt, and denoting T_P^n as the temperature at node P at time t^n, a composite form (Θ discretization) for a central node is:

$$\begin{aligned} T_P^{n+1} = T_P^n + \frac{D\Delta t}{\Delta x^2} \Big[& \Theta\big(T_E^{n+1} - 2T_P^{n+1} + T_W^{n+1}\big) \\ & + (1-\Theta)\big(T_E^n - 2T_P^n + T_W^n\big) \Big]. \end{aligned} \tag{2}$$

Rearranging, the nodal equation for the interior nodes becomes:

$$a_P T_P^{n+1} - a_W T_W^{n+1} - a_E T_E^{n+1} = b, \tag{3}$$

with

$$a_W = a_E = -\Theta \frac{D\Delta t}{\Delta x^2}, \qquad a_P = 1 + 2\Theta \frac{D\Delta t}{\Delta x^2},$$

and

$$b = T_P^n + (1-\Theta)\frac{D\Delta t}{\Delta x^2}\big(T_E^n - 2T_P^n + T_W^n\big).$$

From here, the schemes are obtained:

- Explicit ($\Theta = 0$): T_P^{n+1} is solved for directly.
- Implicit ($\Theta = 1$): A linear system for T^{n+1} is solved.
- Crank–Nicolson ($\Theta = \frac{1}{2}$): A combination of both.

3.1. Boundary Nodes (Ghost Nodes)

To implement Dirichlet conditions at the ends $x = 0$ and $x = L$, ghost nodes are used when necessary to maintain the same difference scheme for all nodes (this is detailed in the specific equations below).

3.2. Particular Equations (Summary)

To keep the notation consistent with the original, the corrected and consistent expressions with $\Delta x, \Delta t$ are presented below.

3.2.1. Central Nodes (Compact Form)

$$\left(\frac{\Delta x}{\Delta t} + \Theta\frac{2D}{\Delta x}\right) T_P^{n+1} - \Theta\frac{D}{\Delta x}(T_E^{n+1} + T_W^{n+1}) = \left(\frac{\Delta x}{\Delta t} - (1-\Theta)\frac{2D}{\Delta x}\right) T_P^n + (1-\Theta)\frac{D}{\Delta x}(T_E^n + T_W^n). \tag{4}$$

(Equivalent to your original derivation, expressed with consistent notation.)

3.2.2. End Nodes

Node 1 (Left Boundary, with ghost node at W):

$$\begin{gathered}\left(\frac{\Delta x}{\Delta t} + \Theta\left(\frac{D_E}{\Delta x} + 2\frac{D_W}{\Delta x}\right)\right) T_1^{n+1} - \Theta\frac{D_E}{\Delta x} T_2^{n+1} = \\ \left(\frac{\Delta x}{\Delta t} + (1-\Theta)\left(-\frac{D_E}{\Delta x} + 2\frac{D_W}{\Delta x}\right)\right) T_1^n + (1-\Theta)\frac{D_E}{\Delta x} T_2^n + (2\Theta\frac{D_W}{\Delta x})T_a,\end{gathered} \tag{5}$$

where T_a is the imposed temperature at the wall/boundary according to the case.

Node n (Right Boundary, with ghost node at E):

$$\begin{gathered}\left(\frac{\Delta x}{\Delta t} + \Theta\left(2\frac{D_E}{\Delta x} + \frac{D_W}{\Delta x}\right)\right) T_n^{n+1} - \Theta\frac{D_W}{\Delta x} T_{n-1}^{n+1} = \\ \left(\frac{\Delta x}{\Delta t} + (1-\Theta)\left(-2\frac{D_E}{\Delta x} - \frac{D_W}{\Delta x}\right)\right) T_n^n + (1-\Theta)2\frac{D_E}{\Delta x} T_b^n + (1-\Theta)\frac{D_W}{\Delta x} T_{n-1}^n,\end{gathered} \tag{6}$$

where T_b is the imposed temperature at the right boundary according to the problem.

3.3. Explicit and Implicit Schemes (Particular Forms)

3.3.1. Explicit ($\Theta = 0$)

Central Nodes

$$\frac{\Delta x}{\Delta t} T_P^{n+1} = \left(\frac{\Delta x}{\Delta t} - \frac{2D}{\Delta x}\right) T_P^n + \frac{D}{\Delta x}(T_E^n + T_W^n). \tag{7}$$

Node 1 and Node n (The same formulas from the general derivation are maintained by applying $\Theta = 0$; in practice, ghost nodes are used to impose the boundary conditions as in your original version.)

3.3.2. Implicit ($\Theta = 1$)

Central Nodes

$$\left(\frac{\Delta x}{\Delta t} + \frac{2D}{\Delta x}\right) T_P^{n+1} - \frac{D}{\Delta x}(T_E^{n+1} + T_W^{n+1}) = \frac{\Delta x}{\Delta t} T_P^n. \tag{8}$$

End Nodes (Obtained analogously by applying $\Theta = 1$ to the boundary expressions; solved by solving the resulting linear system.)

4. Results and Mesh Independence

For the solution of the equations generated in the Finite Volume method, numerical methods can be used for the iterative solution of the systems of equations. For the proposed problem, a code in Python (see Appendices) was implemented that solves the discretized equations; the program allows control of the number of spatial and temporal nodes as well as the boundary conditions.

4.1. Implicit Scheme

Numerically solving the discretized equations made it possible to construct a temporal solution of the T distribution. The properties selected for the numerical solution are shown in Table 1.

Cuadro 1: Problem Properties

Data	**Value**	Units
$T(x=0,t)$	100	°C
$T(x=1,t)$	100	°C
L	1	m
nodes_x	30	-
nodes_t	100	-
D	0.5	-
Θ	$0 \leq \Theta \leq 1$	-
$E_{propuesto}$	0.001	-

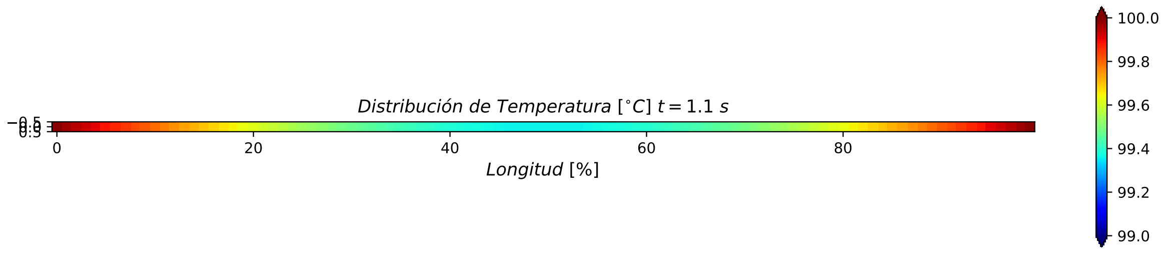
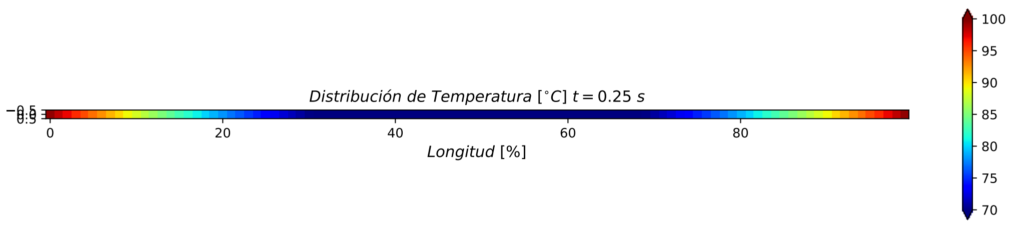
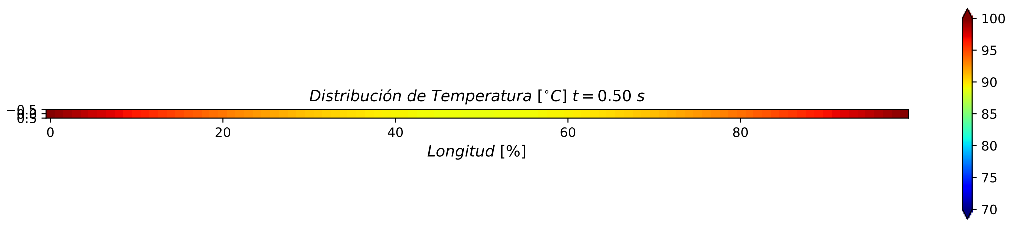


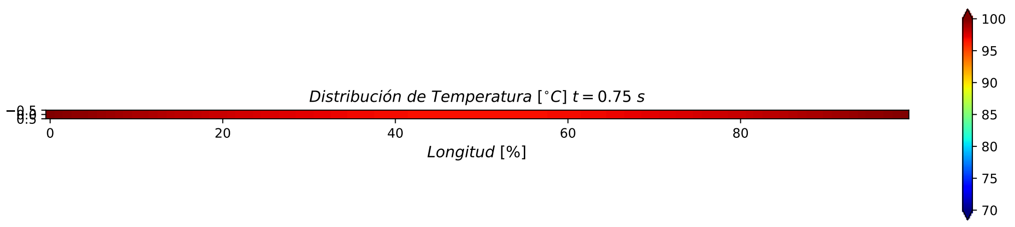
Figure 2. Implicit Method, $t = 1,1$ (approximate time to reach steady state)



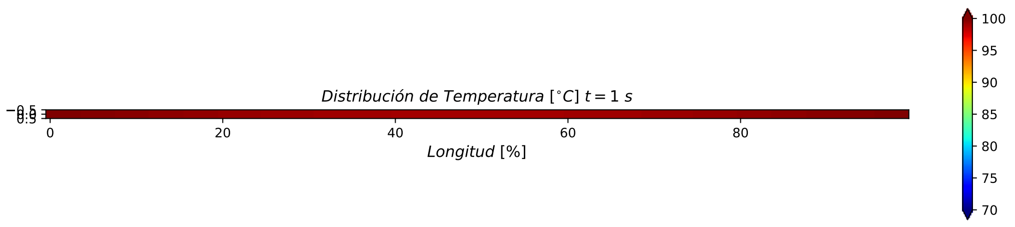
(a) Distribution of T at $t = 0{,}25$ s



(b) Distribution of T at $t = 0{,}5$ s



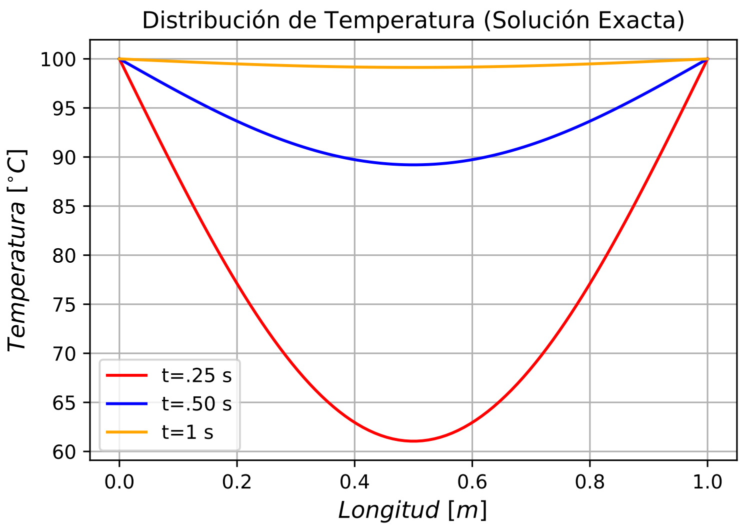
(c) Distribution of T at $t = 0{,}75$ s



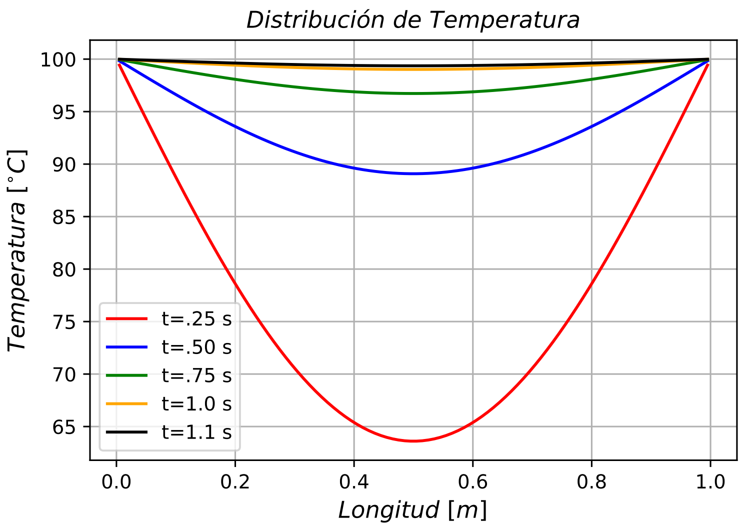
(d) Distribution of T at $t = 1$ s

Figura 1: Implicit Method: $0 \leq t \leq 1$.

In Figure 2, it is observed that the temperature varies approximately between 99,5 °C and 100 °C at time $t = 1{,}1$ s, which indicates that a state close to stationary is reached around 1.1 s with the imposed conditions. Figure 3 shows the temporal evolution at different times; it is observed that by $t = 1$ s the solution is already approaching the steady state, consistent with Figure 2.
Since the implicit scheme is not restricted by a conditional stability criterion, relatively large Δt and Δx can be used compared to the explicit scheme; in this case, 100 temporal nodes were chosen to divide the time interval until stability in steps of 0,01 s and appreciate the evolution.



(a) Distribution of T (exact solution)



(b) Distribution of T (numerical, implicit)

Figura 2: Comparison of exact vs numerical (implicit) solution.

According to Figure 4 and calculating an average error by node-to-node difference (L_2 norm or average percentage, depending on your choice), the values reported in Table 2 were obtained.

Cuadro 2: Errors (Implicit) — Average Error

time	**Average Error**
$t = 0{,}25\ s$	$1.45\,\%$
$t = 0{,}50\ s$	$1.034\,\%$
$t = 1{,}0\ s$	$1.23\,\%$

4.2. Explicit Scheme

The explicit scheme has the particularity that its stability depends on the temporal and spatial step. The stability criterion for the diffusion problem is:

$$\Delta t \leq \frac{\Delta x^2}{2D}. \tag{9}$$

Failure to meet this condition causes numerical instability in the explicit scheme.

Cuadro 3: Problem Properties (Explicit)

Data	**Value**	Units
$T(x = 0, t)$	100	°C
$T(x = 1, t)$	100	°C
L	1	m
nodes_x	10	-
nodes_t	100	-
D	0.5	-
Θ	0	-
$E_{propuesto}$	0.001	-

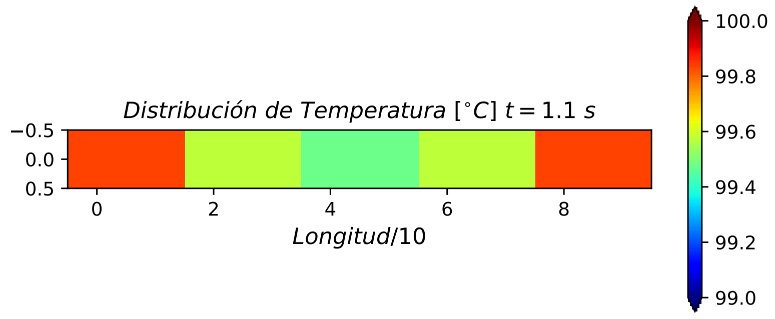
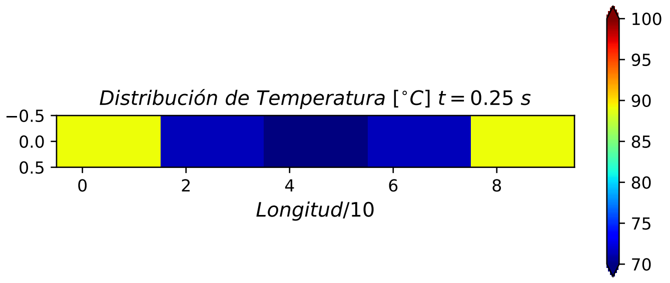
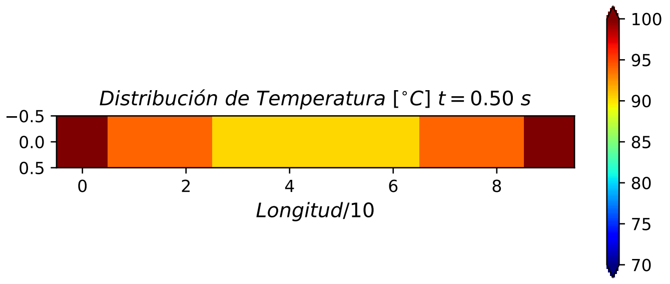


Figure 5. Explicit Method, $t = 1{,}1$ (approximate time to reach steady state)

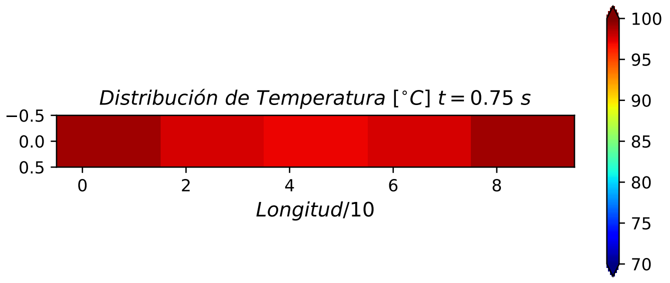
In Figure 5, the resolution is affected by the selection of spatial nodes (10 nodes here), which increases the error and makes the solution less smooth compared to the implicit scheme.



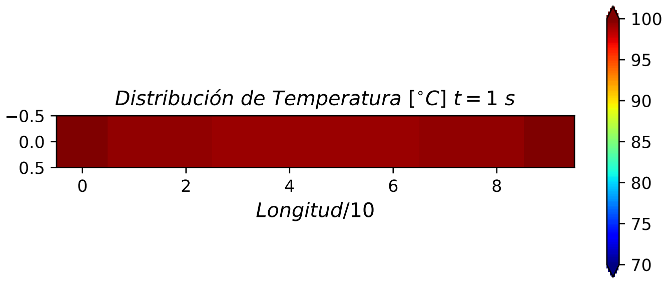
(a) Distribution of T at $t = 0{,}25$ s



(b) Distribution of T at $t = 0{,}5$ s

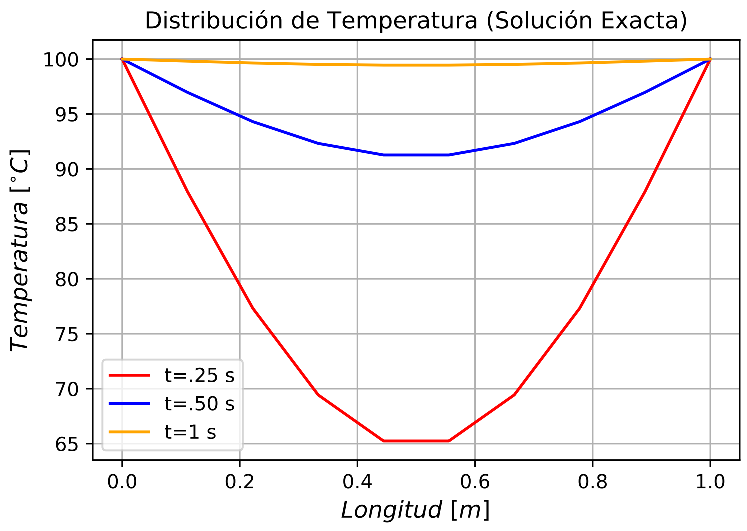


(c) Distribution of T at $t = 0{,}75$ s

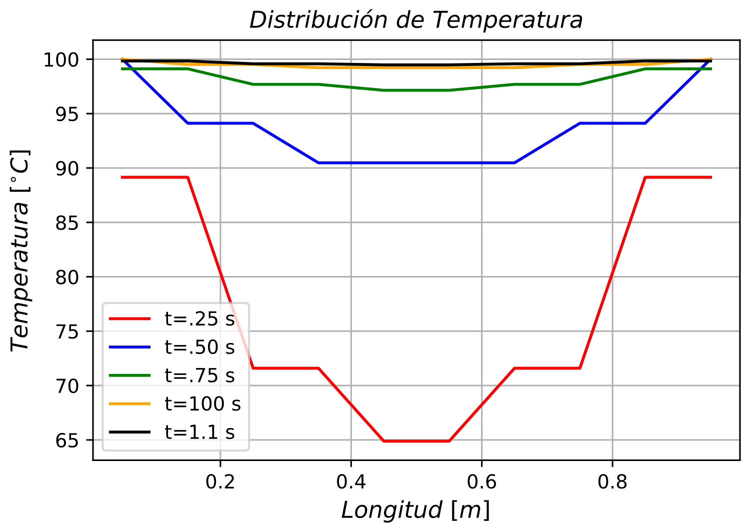


(d) Distribution of T at $t = 1$ s

Figura 3: Explicit Method: $0 \leq t \leq 1$.



(a) Distribution of T (exact solution)



(b) Distribution of T (numerical, explicit)

Figura 4: Comparison of exact vs numerical (explicit) solution.

Cuadro 4: Errors (Explicit) — Average Error

time	**Average Error**
$t = 0{,}25\ s$	5.12 %
$t = 0{,}50\ s$	3.67 %
$t = 1{,}0\ s$	1.87 %

5. Conclusions

This work presented a comparison between the implicit method and the explicit method for the solution of a transient problem treated with the Finite Volume method. Observing the results, it is concluded that the most convenient method is the **implicit method**, as it is **unconditionally stable** compared to the explicit scheme. However, the implicit scheme involves solving a system of equations at each time step, which demands more operations.

The percentage error of each method compared to the exact solution largely depends on the number of spatial nodes selected: a greater number of elements makes it easier to capture variations and reduce the error.

The results show that the system reaches the steady state in approximately 1,1 s; the comparisons at $t = 0{,}25, 0{,}50, 0{,}75$, and 1,0 s allow verification of the evolution towards thermal relaxation. The final reported value (approximate variation of 0,003 °C at $t = 1{,}2$ s) is consistent with the stability observed in the simulations.

Referencias

[1] H. Versteeg and W. Malalasekera, “An introduction to computational fluid dynamics,” *Finite Volume Method, Essex, Longman Scientific & Technical*, 1995.

Appendix I: Solution Codes

```python
print('\033[H\033[J')
import numpy as np
import matplotlib.pyplot as plt
from gauss_seidel import gauss_seidel
n=100 #orden del sistema (numero de nodos)
n_time=110 #orden del tiempo (numero de nodos temporales)
time=1.1 #segundos
error=.001 #error propuesto

A=np.zeros(((n,n)),float)
X=np.zeros(((n)),float)
B=np.zeros(((n)),float)
Xc=np.zeros(((n)),float)
E=np.zeros(((n)),float)
Ln=np.zeros(((n)),float)
B0=np.zeros(((n)),float)

#vectores para gracicar cada 25% de avance del tiempo de solucion
X25=np.zeros(((n)),float)
X50=np.zeros(((n)),float)
X75=np.zeros(((n)),float)
X100=np.zeros(((n)),float)
X110=np.zeros(((n)),float)


theta=1
T_a=100
T_b=100
L=1
dx=L/n
dt=time/n_time
De=Dw=.5
a=dx/dt
ae=De/dx
aw=Dw/dx

for j in range(0,n_time):
   for i in range(0,n):
       aux1=i+1
       aux2=i-1
       Ln[i]=aux1*dx-(dx/2)
       if i==0:
          A[i,i]=a+(theta)*(ae+2*aw)
          A[i,aux1]=-theta*ae
          B[i]=(a+(1-theta)*(-ae-2*aw))*B0[i]+(ae*(1-theta))*B0[aux1]+(2*aw*(1-theta))*T_a
    +(2*aw*theta)*T_a
       if i==(n-1):
          A[i,i]=a+(theta)*(2*ae+aw)
          A[i,aux2]=-theta*aw
          B[i]=(a+(1-theta)*(-2*ae-aw))*B0[i]+(ae*(1-theta))*B0[aux2]+(2*ae*(1-theta))*T_b
    +(2*ae*theta)*T_b
```

```
       if i!=0 and i!=n-1:
          A[i,i]=a+theta*(ae+aw)
          A[i,aux1]=-theta*ae
          A[i,aux2]=-theta*aw
          B[i]=(a+(1-theta)*(-ae-aw))*B0[i]+(ae*(1-theta))*B0[aux1]+(aw*(1-theta))*B0[aux2]
   AA=np.linalg.inv(A)
   sol=np.dot(AA,B)
   #X=gauss_seidel(A,B,X,Xc,E,n,error,np)
   B0=sol
   if j==25:
       X25=sol
   if j==50:
       X50=sol
   if j==75:
       X75=sol
   if j==100:
       X100=sol
   if j==109:
       X110=sol

plt.figure()
plt.plot(Ln,X25,label = "t=.25 s", color = ’red’)
plt.hold(True)
plt.plot(Ln,X50,label = "t=.50 s", color = ’blue’)
plt.plot(Ln,X75, label = "t=.75 s", color = ’green’)
plt.plot(Ln,X100, label = "t=1 s", color = ’orange’)
plt.plot(Ln,X110, label = "t=1.1 s", color = ’black’)
plt.grid(True)
plt.legend(loc="lower left")
plt.xlabel(r"$Longitud\ [m]$", fontsize = 12, color =’black’)
plt.ylabel(r"$Temperatura\ [^{\circ} C]$", fontsize = 12, color = ’black’)
plt.title(r"$Distribucion\ de\ Temperatura$",fontsize = 12, color = ’black’,
    verticalalignment = ’baseline’, horizontalalignment = ’center’)
plt.savefig(’Grafica1.pdf’)
plt.show()

espesor=int(n/10)
X25_1=np.zeros(((espesor*n)),float)
X50_1=np.zeros(((espesor*n)),float)
X75_1=np.zeros(((espesor*n)),float)
X100_1=np.zeros(((espesor*n)),float)
X110_1=np.zeros(((espesor*n)),float)
for k in range(1,espesor+1):
   for l in range(0,(espesor+1)*n):
       if l<k*n and l>=(k-1)*n:
          X25_1[l]=X25[l-(k-1)*n]
          X50_1[l]=X50[l-(k-1)*n]
          X75_1[l]=X75[l-(k-1)*n]
          X100_1[l]=X100[l-(k-1)*n]
          X110_1[l]=X100[l-(k-1)*n]

plt.figure(1)
arr = X25.reshape((1,n))
fig = plt.figure(figsize=(15, 3))
im1 = plt.imshow(arr, cmap=’jet’)
plt.colorbar(extend=’both’)
plt.clim(70, 100);
plt.xlabel(r"$Longitud\ [\%]$", fontsize = 12, color = ’black’)
plt.title(r"$Distribucion\ de\ Temperatura\ [^{\circ} C]\ t=0.25\ s$",fontsize = 12, color =
     ’black’, verticalalignment = ’baseline’, horizontalalignment = ’center’)
```

```
plt.savefig('Grafica2_a.pdf')

plt.figure(2)
arr2 = X50.reshape((1,n))
fig = plt.figure(figsize=(15, 3))
im2 = plt.imshow(arr2, cmap='jet')
plt.colorbar(extend='both')
plt.clim(70, 100);
plt.xlabel(r"$Longitud\ [\%]$", fontsize = 12, color = 'black')
plt.title(r"$Distribucion\ de\ Temperatura\ [^{\circ} C]\ t=0.50\ s$",fontsize = 12, color =
    'black', verticalalignment = 'baseline', horizontalalignment = 'center')
plt.savefig('Grafica2_b.pdf')

plt.figure(3)
arr3 = X75.reshape((1,n))
fig = plt.figure(figsize=(15, 3))
im3 = plt.imshow(arr3, cmap='jet')
plt.colorbar(extend='both')
plt.clim(70, 100);
plt.xlabel(r"$Longitud\ [\%]$", fontsize = 12, color = 'black')
plt.title(r"$Distribucion\ de\ Temperatura\ [^{\circ} C]\ t=0.75\ s$",fontsize = 12, color =
    'black', verticalalignment = 'baseline', horizontalalignment = 'center')
plt.savefig('Grafica2_c.pdf')

plt.figure(4)
arr4 = X100.reshape((1,n))
fig = plt.figure(figsize=(15, 3))
im4 = plt.imshow(arr4, cmap='jet')
plt.colorbar(extend='both')
plt.clim(70, 100);
plt.xlabel(r"$Longitud\ [\%]$", fontsize = 12, color = 'black')
plt.title(r"$Distribucion\ de\ Temperatura\ [^{\circ} C]\ t=1\ s$",fontsize = 12, color = '
    black', verticalalignment = 'baseline', horizontalalignment = 'center')
plt.savefig('Grafica2_d.pdf')

plt.figure(4)
arr5 = X110.reshape((1,n))
fig = plt.figure(figsize=(15, 3))
im4 = plt.imshow(arr5, cmap='jet')
plt.colorbar(extend='both')
plt.clim(99, 100);
plt.xlabel(r"$Longitud\ [\%]$", fontsize = 12, color = 'black')
plt.title(r"$Distribucion\ de\ Temperatura\ [^{\circ} C]\ t=1.1\ s$",fontsize = 12, color =
    'black', verticalalignment = 'baseline', horizontalalignment = 'center')
plt.savefig('Grafica2_e.pdf')
```