Universidad de Guanajuato

Finite Volume Method (Project 1)

García Sánchez Marco Antonio[1]

[1]División de ingenierías campus Irapuato-Salamanca, Universidad de Guanajuato, Salamanca, Gto.

Instructor: César Eduardo Damián Ascencio, Ph.D.

May 2019

Contents

Abstract

This work analyzes a one-dimensional heat transfer problem using the Finite Volume Method (FVM). The objective is to determine the temperature distribution and to observe the differences resulting from the sensitivity of the employed mesh.

1 Introduction

The Finite Volume Method (FVM) was introduced in the 1970s by McDonald, MacCormack, and Paullay, and since then, it has become one of the preferred methods among scientists and engineers in the field of fluid mechanics.

The starting point of the method is the decomposition of the domain into small control volumes (CVs) where variables are stored at nodes. Usually, these control volumes are defined based on a structured mesh, and variables are evaluated at the centers or vertices of these volumes. Subsequently, the conservation equations are formulated in their integral form for each control volume, and the resulting system is solved numerically.

Although the Finite Element Method (FEM) has made significant advances in recent decades, the FVM remains the most practical choice for complex problems, especially those involving multiphase, reactive, or highly turbulent flows.

2 Governing Equations

A schematic of the problem is shown in Figure 1, while the corresponding data are presented in Table 1.

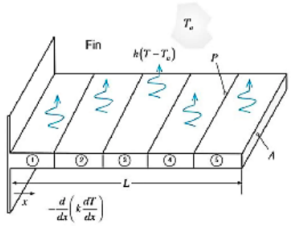

Figure 1: Scheme of the physical problem considered.

For the analysis, a differential element of the fin is taken, as shown in Figure 2.

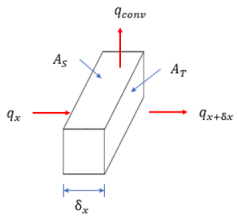

Figure 2: Differential element of the fin.

The following assumptions are made:

- One-dimensional heat transfer.
- Constant thermal conductivity.
- Negligible surface radiation.
- No internal heat generation.
- Uniform convection coefficient on the surface.

From the energy balance shown in Figure 2:

$$q_x = q_{x+dx} + q_{conv} \tag{1}$$

We know that $q_x = -A_t k \frac{dT}{dx}$ and $q_{conv} = -A_s h(T - T_\infty)$. With these relationships, the governing heat transfer equation in the fin is obtained:

$$-\frac{d}{dx}\left(k\frac{dT}{dx}\right) + \frac{hp}{A_t}(T - T_\infty) = 0 \tag{2}$$

3 Discretization

Integrating the previous equation over a control volume and applying central finite difference approximations, we obtain:

$$\frac{T_W - 2T_P + T_E}{\Delta x^2} + \frac{hp}{A_t k}(T_P - T_\infty) = 0 \tag{3}$$

This equation is valid for internal nodes. The following subsections describe the boundary conditions for different cases.

3.1 Case 1: Fixed temperatures at both ends

For the end nodes, ghost nodes are introduced and modified equations are obtained:

Initial node:

$$T_1\left(\frac{3k}{\Delta x}+\frac{hp\Delta x}{A_t}\right)+T_2\left(-\frac{k}{\Delta x}\right)=\frac{hp\Delta x}{A_t}T_\infty+2T_A\left(\frac{k}{\Delta x}\right) \tag{4}$$

Final node:

$$T_n\left(\frac{3k}{\Delta x}+\frac{hp\Delta x}{A_t}\right)+T_{n-1}\left(-\frac{k}{\Delta x}\right)=\frac{hp\Delta x}{A_t}T_\infty+2T_B\left(\frac{k}{\Delta x}\right) \tag{5}$$

3.2 Case 2: Constant heat flux at both ends

Initial node:

$$T_1\left(\frac{k}{\Delta x}+\frac{hp\Delta x}{A_t}\right)+T_2\left(-\frac{k}{\Delta x}\right)=\frac{hp\Delta x}{A_t}T_\infty+q_{x=0} \tag{6}$$

Final node:

$$T_n\left(\frac{k}{\Delta x}+\frac{hp\Delta x}{A_t}\right)+T_{n-1}\left(-\frac{k}{\Delta x}\right)=\frac{hp\Delta x}{A_t}T_\infty-q_{x=L} \tag{7}$$

4 Results and Mesh Independence

The discretized equations were solved using the Gauss–Seidel iterative method implemented in Python. The code used is shown in the appendix.

The properties and characteristics of the problem are presented in Table 1.

Table 1: Problem properties

Parameter	Value	Units
T_A	200	°C
T_B	90	°C
T_∞	25	°C
h	100	W/m^{2}K
k	160	W/mK
L	0.1	m
A_s	10^{-5}	m^2
p	0.1004	m
$E_{proposed}$	0.001	–

The mesh independence results are presented in Table 2. It is observed that for a number of nodes greater than 30, the temperature variation stabilizes.

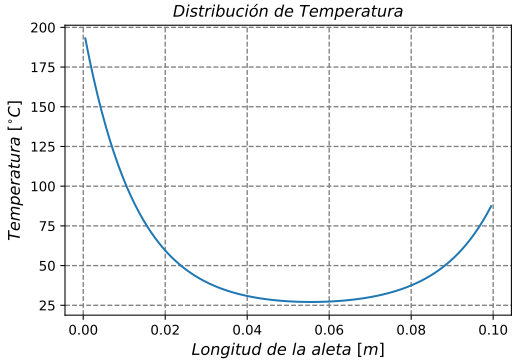

Figure 3: Temperature distribution in the fin (Case 1).

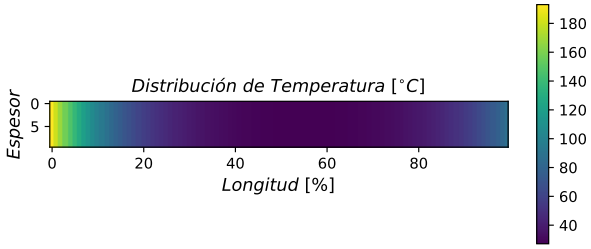

Figure 4: Temperature field and color map of the fin (Case 1).

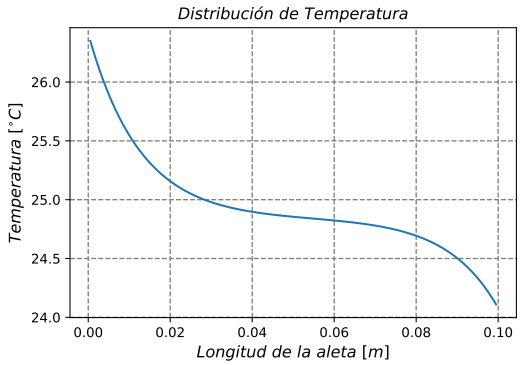

Figure 5: Temperature distribution in the fin (Case 2).

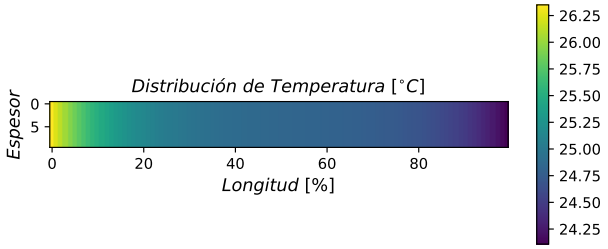

Figure 6: Temperature field and color map of the fin (Case 2).

5 Conclusions

The Finite Volume Method proves to be a fundamental tool for analyzing heat transfer and fluid flow phenomena. The discrete approximation of the domain allows the solution of problems where analytical solutions are not feasible, especially in complex geometries.

It is concluded that a mesh between 30 and 40 nodes provides sufficiently accurate results with a reasonable computational cost. The Gauss–Seidel method showed good convergence for the proposed error, validating its use in this type of simulation.

Appendix I: Solution Codes

```
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_t=800 #nodos temporales
error=.01 #error propuesto
A=np.zeros(((n,n)),float)
T=70*np.ones(((n)),float)
B=np.zeros(((n)),float)
Ba=np.zeros(((n)),float)
Xc=np.zeros(((n)),float)
E=np.zeros(((n)),float)
Ln=np.zeros(((n)),float)
Lnt=np.zeros(((8)),float)

Tn0=np.ones(((8)),float)
Tnm=np.ones(((8)),float)
Tnf=np.ones(((8)),float)

Tp2=np.ones(((n)),float)
Tp4=np.ones(((n)),float)
Tp6=np.ones(((n)),float)

L=.5/12
t=8/3600
dx=L/n
dt=t/n_t
k_unam=13
k_fimee=22.9
alpha_unam=.1775
alpha_fimee=.298
Tfusion_unam=2570
Tfusion_fimee=2770
T_0h=4500
T_0c=70
h_h=1000
h_c=300
Bh_u=h_h*dx/k_unam
Bh_f=h_h*dx/k_fimee
F_u=alpha_unam*dt/(dx**2)
F_f=alpha_fimee*dx/(dx**2)
Bc_u=h_c*dx/k_unam
Bc_f=h_c*dx/k_fimee


fimee=1
unam=2
material=fimee

if material==fimee:
    alpha=alpha_fimee
    Tfusion=Tfusion_fimee
    Bh=Bh_f
    Bc=Bc_f
    k=k_fimee
    F=F_f

if material==unam:
    alpha=alpha_unam
    Tfusion=Tfusion_unam
    Bh=Bh_u
    Bc=Bc_u
    k=k_unam
```

```
    F=F_u

explicito=1
implicito=2
CrankN=3

case=implicito
if case==1:
   a=1
   b=0
if case==2:
   a=0
   b=1
if case==3:
   a=.5
   b=.5


for r in range(0,8):
   Ba=T
   Tn0[r]=T[0]
   Tnm[r]=T[50]
   Tnf[r]=T[99]
   Lnt[r]=dt*r*3600*100
   if r==2:
      Tp2=T
   if r==4:
      Tp4=T
   if r==6:
      Tp6=T

   for i in range(0,n):
       aux1=i+1
       aux2=i-1
       Ln[i]=dx*i


       if i==0:
          A[i,i]=a*(1)+b*(1+2*Bh*F+2*F)
          A[i,aux1]=a*(0)-b*(2*F)
          B[i]=a*(Ba[i]*(1-2*F-2*F*Bh)+2*F*(Ba[aux1]+Bh*T_0h))+b*(Ba[i]+2*Bh*F*T_0h)
       if i==(n-1):
          A[i,i]=a*(1)+b*(1+2*Bc*F+2*F)
          A[i,aux2]=a*(0)-b*(2*F)
          B[i]=a*(Ba[i]*(1-2*F-2*F*Bc)+2*F*(Ba[aux2]+Bc*T_0c))+b*(Ba[i]+2*Bc*F*T_0c)
       if i!=0 and i!=n-1:
          A[i,i]=a*1+b*(1+2*F)
          A[i,aux1]=-a*(0)-b*(F)
          A[i,aux2]=-a*(0)-b*(F)
          B[i]=a*(F*(Ba[aux1]+Ba[aux2])+(1-2*F)*Ba[i])+b*(Ba[i])
   X=gauss_seidel(A,B,T,Xc,E,n,error,np)


plt.figure()
plt.plot(Lnt,Tn0,label = "x=0 ", color = 'red')
plt.hold(True)
plt.plot(Lnt,Tnm,label = "x=L/2 ", color = 'blue')
plt.plot(Lnt,Tnf,label = "x=l", color = 'green')
plt.grid(color = '0.5', linestyle = '--', linewidth = 1)

plt.savefig('GraficaT-t.pdf')
plt.show()


plt.figure()

plt.hold(True)
```

```
plt.plot(Ln,Tp2,label = "t=2 s", color = 'blue')
plt.plot(Ln,Tp4,label = "t=4 s", color = 'green')
plt.plot(Ln,Tp6,label = "t=6 s", color = 'yellow')
plt.plot(Ln,T,label = "t=8 s", color = 'red')
#plt.grid(True)
plt.grid(color = '0.5', linestyle = '--', linewidth = 1)

plt.savefig('GraficaT-x.pdf')
plt.show()

Xg=np.zeros(((10*n)),float)
for k in range(1,11):
   for j in range(0,11*n):
       if j<k*n and j>=(k-1)*n:
          Xg[j]=T[j-(k-1)*n]

plt.close('all')
arr = Xg.reshape((10,n))
fig = plt.figure(figsize=(7, 3))
im = plt.imshow(arr,cmap='jet')
plt.colorbar()
plt.savefig('perfil.pdf')
```

Listing 1: Main code: ProyectoTransfe2.py

```
def gauss_seidel(A,B,X,Xc,E,n,error,np):
   ite=0
   Error_prom=1
   while error<Error_prom and ite<10000:
      ite=ite+1
      for i in range(0,n):
         Xc[i]=X[i]
         suma=0
         for j in range(0,n):
            if j!=i:
               suma=suma+(A[i,j]*X[j])
         X[i]=(B[i]-suma)/A[i,i]
         E[i]=abs(X[i])-abs(Xc[i])
      #E[i]=abs(X[i])-abs(Xc[i])
      Error_prom=abs(np.sum(E,axis=0)/n)

   print('Matriz A')
   print(A)
   print('Vector B')
   print(B)
   print('X')
   print(X)
   print('Xc')
   print(Xc)
   print('iteracion')
   print(ite)
   print('Error Promedio')
   print(Error_prom)
   return(X)
```

Listing 2: Gauss–Seidel function: gauss_seidel.py