Solution of 1-D transient heat conduction equation using FTCS method

Foundations of Computational Fluid Dynamics AM5630

Name - Anand Zambare

Roll no. - AM21S004

Summary

One dimensional transient heat conduction equation is solved using FTCS method i.e. Forward Difference in Time and Central Difference in Space. This is nothing but the implementation of Finite Difference Method(FDM) in Computational Fluid Dynamics (CFD).Here we have fixed the no. of grid points and hence the distance between them (delta x). We have used different time steps to solve the problem. We could see that we get solution only for the time step which satisfies the stability criteria of the given scheme. In other cases solution will diverge.

1 Problem Definition

Given problem is to solve for the steady state temperature profile in a given 1-D domain using FTCS method. The size of domain is user defined (taken as 1 units) and no. of grid points are also defined by user. We have to write a generalized code to solve an unsteady 1-D heat conduction equation using Forward Difference in Time and Central difference in Space scheme. The Boundary conditions are to be implemented on first and last node in the domain and Initial condition is implemented for all nodes for time = 0. We shall check the time steps for which we can solve the problem using explicit FTCS scheme and check if we follow the stability criteria of the given scheme. Finally obtain the steady state solution with some user defined tolerance and plot the temperature contours

L = Length of domain = 1 meter (Taken here)
N_x = No. of grid points in the domain = 11 (Taken here)
Thermal diffusivity = 1 m/s^2 (Taken here)

2 Governing Equation

For the given problem, to find out the steady state temperature distribution we need to solve Transient Heat Conduction equation which is given as follows.

$$\frac{\partial T}{\partial t} = \alpha^2 \cdot (\frac{\partial^2 T}{\partial x^2}) \tag{1}$$

We will discretize the given equation using Finite Difference Method(FDM). We will use Forward difference Scheme for time and Central difference scheme for space.

3 Initial and Boundary conditions

We have to implement initial condition for all the grid points and Boundary conditions for first and last node.

3.1 Initial Condition

At time t = 0, the initial condition is temperature at all nodes is 0 units. Hence T = 0 for all X_i Physically, This initial condition imposes the room temperature value on all the nodes at time = 0

3.2 Boundary Conditions

At X_0, $T_0 = 1.0$ and at X_N, $T_N = 0.0$ for all time t > 0.
Physically, The boundary condition at first node i.e. X_0 is rise of temperature to 1.0 and the boundary condition imposed is that the temperature value will be 0.0 at the last node i.e. X_N.

4 Numerical Formulation

Let's look at the numerical formulation of governing differential equation using Finite Difference Method (FDM). The governing equation is

$$\frac{\partial T}{\partial t} = \alpha \cdot (\frac{\partial^2 T}{\partial x^2}) \tag{2}$$

Here we will use Forward difference in time and central difference in Space (Hence the method is FTCS.) Forward difference in time gives us

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\delta t} \tag{3}$$

Central Difference in Space gives us

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\delta x)^2} \tag{4}$$

Using the above equation (3) and (4) we convert the governing equation into following equation.

$$\frac{T_i^{n+1} - T_i^n}{\delta t} = \alpha(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\delta x)^2}) \tag{5}$$

if we do some more simplification by assuming

$$\gamma = \frac{\alpha \delta t}{\delta x^2} \tag{6}$$

On further simplification and writing the unknown on one side of equation we get

$$T_i^{n+1} = \gamma T_{i+1}^n + (1 - 2\gamma)T_i^n + \gamma T_{i-1}^n \tag{7}$$

Using equation (7) we can solve for 'i'th node at 'n+1' th time step. This is the numerical formulation we will be using.

5 Algorithm Flow Chart

Following figure i.e. figure 1 is a flow chart of algorithm used for solving given problem. All the variables are predefined in the problem. Here the criteria for the steady state is error at any grid point must fall to 10^{-6} then we say that the steady state is achieved by the temperature values. The implemented algorithm is quite simple and easy to understand. As we are dealing with only one

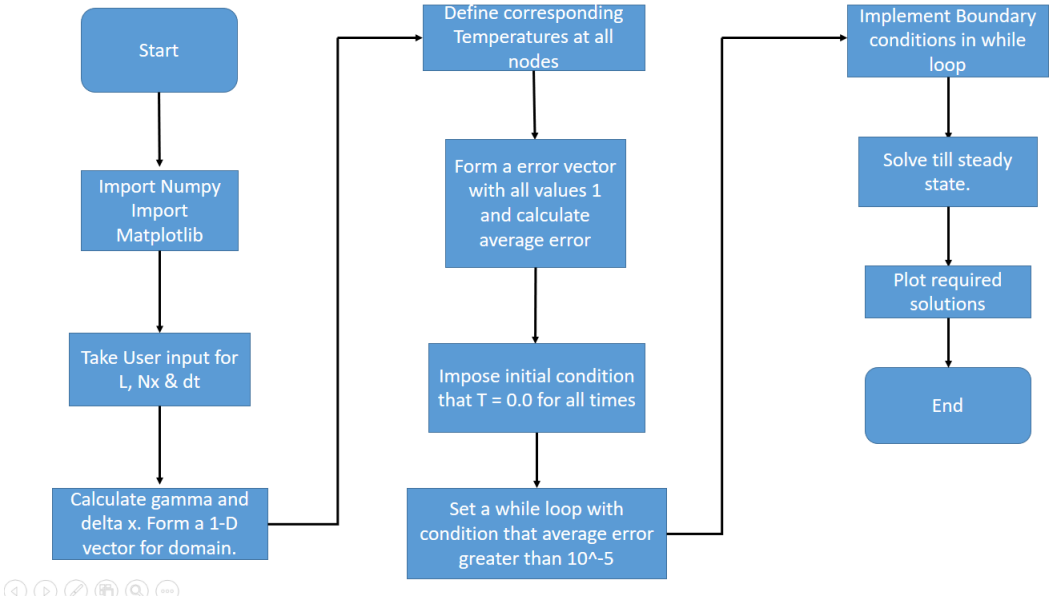


Figure 1: Flow chart for explicit FTCS method.

unknown. This is because we have implemented the explicit algorithm. Let's discuss the results.

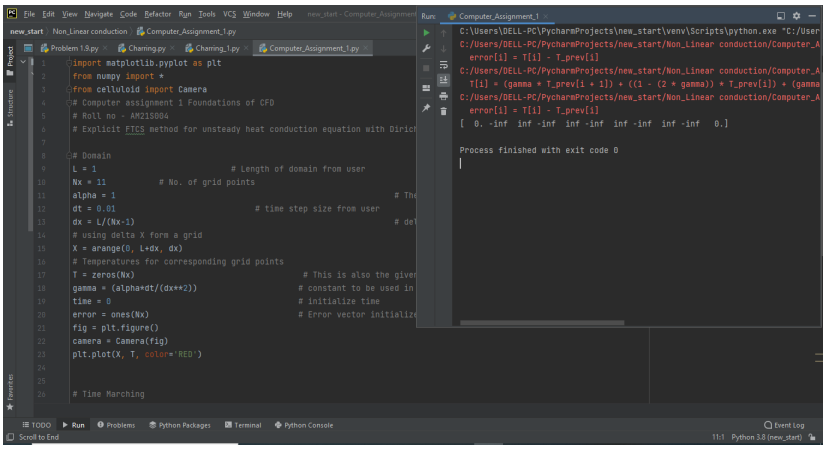
6 Results

For getting the solution we have to satisfy the stability criteria of the scheme as we are implementing explicit scheme. We have derived the stability criteria and it is as follows

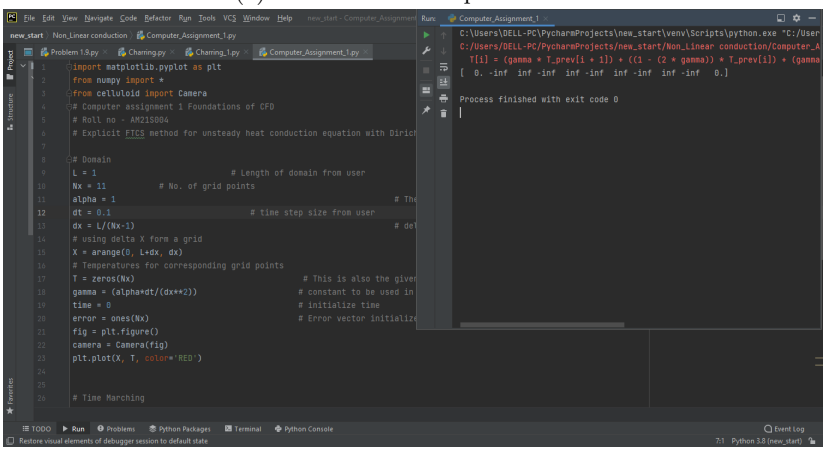
$$\gamma \leq \frac{1}{2} \tag{8}$$

We know that it depends only on time space and grid space as all other quantities are always positive and constant. Hence we can write it as

$$\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{9}$$



(a) Error for time step = 0.1 sec.

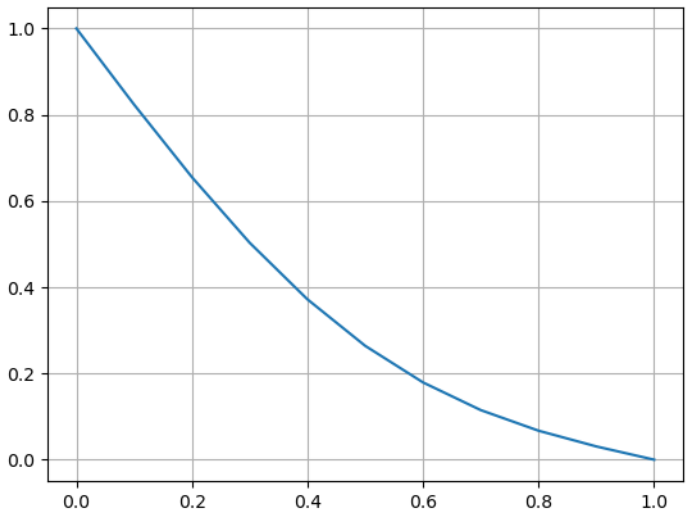


(b) Error for time step = 0.01 sec.

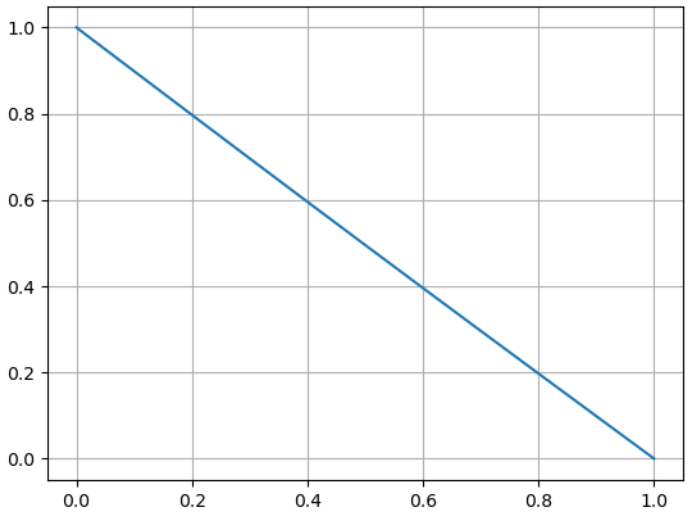
Figure 2

While solving the problem here we have kept the no. of grid points fix. This can also be seen as the grade spacing is uniform and constant. The value is 0.1 between grid spacing so for scheme to be stable the maximum time step we can take is 0.005 i.e. 5×10^{-3}. Here during the solution we have checked what do we get for time step greater than 5×10^{-3} i.e. for 0.1 0.01 second as a time step. For this steps the solution gets diverged and the error will be shown. Following Figure 2a and Figure 2b shows the error for this time steps. So for time step less than or equal to 5×10^{-3} we will get solution. Here I have used time step as 1×10^{-3} i.e. 0.001 and solved till the steady state is achieved. I used the steady state criteria as error to fall down to 1×10^{-6} between in previous and next time step. It took 0.832 seconds for achieving steady state. The error values are all below the threshold value which was given in the problem i.e. 1×10^{-5}

Figure 3 shows the final output of the code. First three lines are input given from user and next lines are the results. Figures 3a, 3b, 4a and 4b are the Temperature plots at 0.1 sec, 0.5 sec, 1.0 sec and 2.5 sec respectively. Since the steady state is achieved at 0.8 sec approximately we can't see any difference in between figure 6 i.e. temperature at 1.0 sec and figure 7 i.e temperature at 2.5 sec.

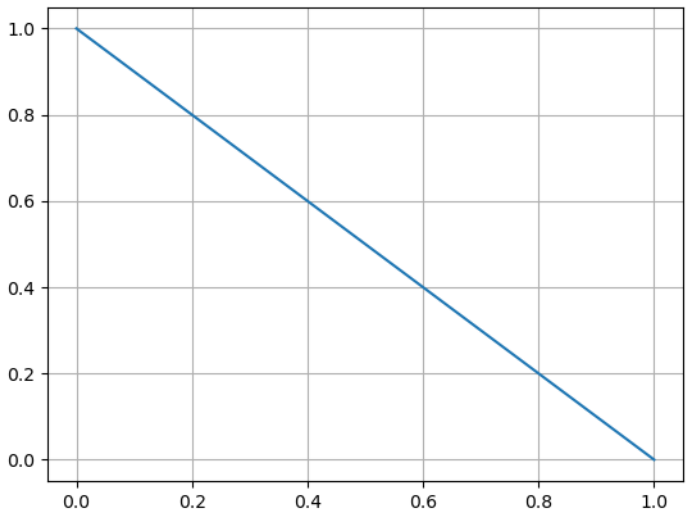


(a) Temperature distribution after 0.1 sec

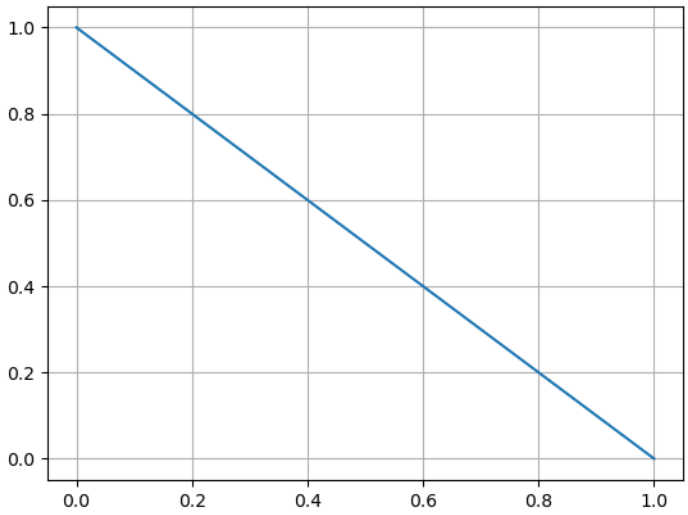


(b) Temperature distribution after 0.5 sec

Figure 3: Temperature distributions at various time steps



(a) Temperature distribution after 1 sec



(b) Temperature distribution after 2.5 sec

Figure 4: Temperature distributions at various time steps

7 Appendix

```
Python code for FTCS method import matplotlib.pyplot as plt
from numpy import *
Computer assignment 1 Foundations of CFD
Roll no - AM21S004
FTCS method for unsteady heat conduction equation with Dirichlet boundary conditions
Domain
L = float(input('Enter the length of domain'))  Length of domain from user
Nx = float(input('Enter the no. of grid points in the domain'))  No. of grid points
alpha = 1  Thermal diffusivity
dt = float(input('Enter the time step size'))  time step size from user
dx = L/(Nx-1)  delta X in the domain
using delta X form a grid
X = arange(0, L+dx, dx)
Temperatures for corresponding grid points
T = zeros(Nx)  This is also the given initial condition
gamma = (alpha*dt/(dx**2))  constant to be used in equation
time = 0  initialize time
error = ones(Nx)  Error vector initialize
fig = plt.figure()
camera = Camera(fig)
plt.plot(X, T, color='RED')
Time Marching
while err ¿ (10**-6):
time = time + dt
T[0] = 1.0  Type 1 boundary condition
Tprev = T.copy()
for i in range(1, Nx-1):
T[i] = (gamma * Tprev[i + 1]) + ((1 - (2 * gamma)) * Tprev[i]) + (gamma * Tprev[i - 1])
T[0] = 1.0  Type 1 boundary condition
T[Nx - 1] = 0.0  Type 1 boundary condition
for i in range(Nx):
error[i] = T[i] - Tprev[i]
plt.plot(X, T, color='RED')
camera.snap()
err = average(error)
animation = camera.animate(interval=1)
fig = plt.figure()
plt.show()
plt.figure(1)
plt.plot(X, T)
plt.grid()
plt.show()
print('The steady state is achieved around', time, 'seconds')
print('The error at each grid point is shown by error vector',error)
```
