**MA202: NUMERICAL METHODS**
**PROJECT REPORT**

**INDIAN INSTITUTE OF TECHNOLOGY GANDHINAGAR**

# PROJECT 1: UNSTEADY HEAT CONDUCTION PROBLEM IN CARTESIAN COORDINATES USING IMPLICIT AND EXPLICIT SCHEMES

*By*

GROUP 25

| | |
|---|---|
| Parth Deshpande | 21110151 |
| Amey Rangari | 21110177 |
| Nishant Tatar | 21110223 |
| Vihaan Jhaveri | 21110236 |
| Yash Patil | 21110244 |

**Semester II, 2022-23**

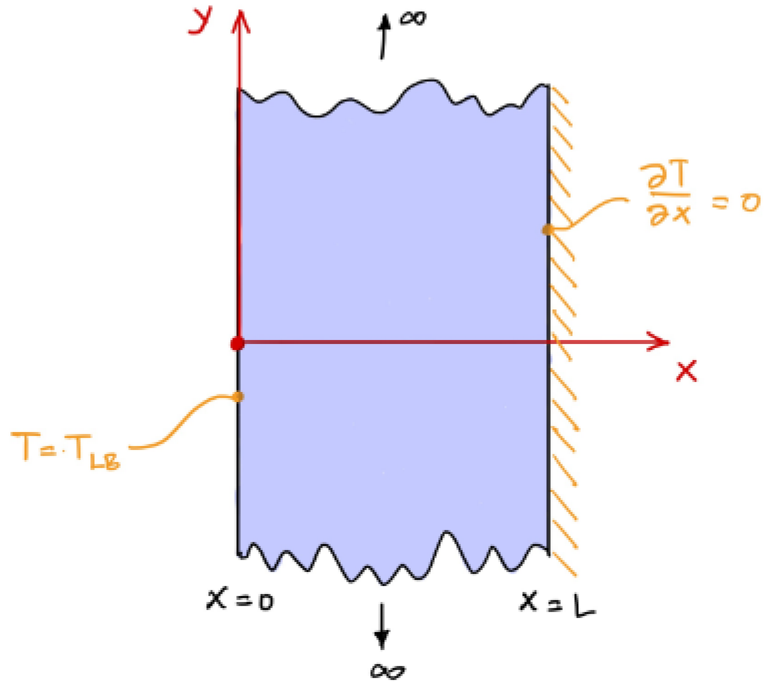
## PROBLEM STATEMENT

The temperature distribution in a long solid bar (see figure 1.1) of length $L$ is governed by the following equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2},$$

where $t$ is time, $T(x,t)$ is temperature and $\alpha$ is called the thermal diffusivity of the material.

The temperature distribution within the solid bar needs to be determined for a given initial condition and boundary conditions. The thermal diffusivity α of the solid bar is given as 10^−4 m^2/s (that of Aluminium). The length of the bar is between 1- 2 m. The code should compute the temperature from t = 0 until t = $\tau_{end}$ s, where $\tau_{end}$ is user-defined. As an output, the temperature distribution T(x, $\tau_{end}$) vs x needs to be plotted for both schemes.
The above equation is subject to the initial condition: T(x, 0) = 300 K and the following boundary conditions: T(0, t) = 400 K and (∂T/∂x)x=L = 0.



## THEORY

Transient heat transfer problems involve the analysis of heat transfer processes in which the temperature of a medium changes with respect to time. One-dimensional Cartesian coordinates are commonly used to solve such problems. The heat conduction equation for one-dimensional Cartesian coordinates in terms of thermal diffusivity α is given by:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

where T is the temperature, t and x are the time and spatial coordinates, and α is the thermal diffusivity.

### **Finite Difference Method**:

The Finite Difference Method is used to approximate solutions to differential equations by discretizing the domain into a grid of points and approximating the derivative at each point using the values of the function at nearby points. It is based on the Taylor series expansion of a function around a point and uses the function values at nearby points to approximate the derivative. The grid size, order of approximation, and smoothness of the function determine the accuracy of the method. It is widely used in various fields to solve differential equations numerically.

To solve the unsteady heat conduction problem, we must solve the following partial differential equation using numerical methods.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

To do so, we will use the finite difference method for both implicit and explicit schemes.

### **For Implicit Scheme**:

Using the Taylor series, the double derivative of a function f can be approximated as below

$f''(x_i) = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}$ ⇐Central difference approximation for the second derivative.

Similarly, the derivative of a function f using the Taylor series can be approximated as

$f'(x_i) = \frac{f_{i+1} - f_i}{\Delta x}$ ⇐ Forward difference approximation.

Now let t and x be the time and spatial coordinates respectively, so according to the given equation,

$$\frac{T_{x,t+1} - T_{x,t}}{\Delta t} = \alpha \times \frac{T_{x+1,t} - 2T_{x,t} + T_{x-1,t}}{\Delta x^2}$$

Now on solving the above equation, it gets simplified to the below equation

$$\boxed{T_{x,t+1} = T_{x,t} + \beta \times (T_{x+1,t} - 2T_{x,t} + T_{x-1,t})}$$

Where $\beta = \frac{\alpha \times \Delta t}{\Delta x^2}$.

This equation is then further solved using iterative methods.

## For Explicit Scheme:

There is not much of a difference between implicit and explicit schemes; the only difference is in the final equation, which is then further solved using iterative methods.

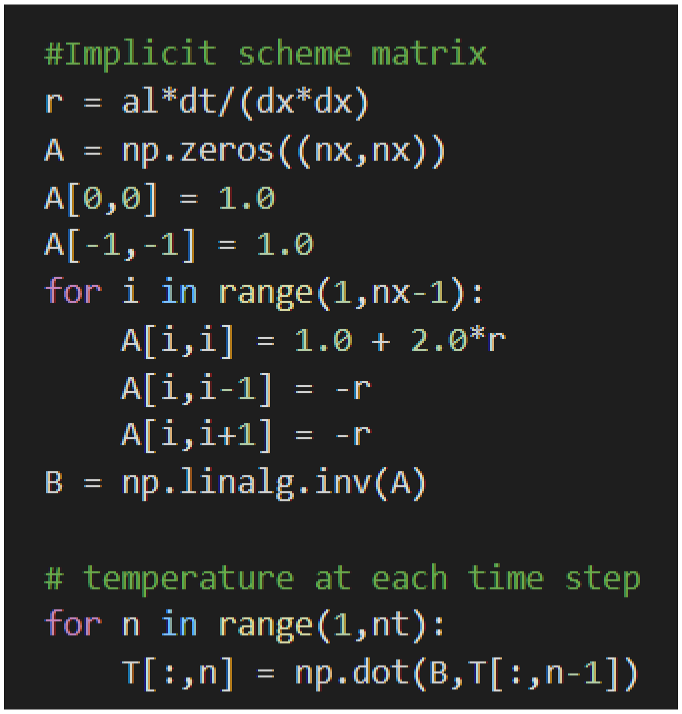
$$\boxed{T_{x,t+1} = T_{x,t} + \beta \times (T_{x+1,t+1} - 2T_{x,t+1} + T_{x-1,t+1})}$$

Where $\beta = \frac{\alpha \times \Delta t}{\Delta x^2}$. Now this is explicit because the time at t+1 depends on the components having the time before.

An implicit scheme is better than an explicit scheme because it is stable and convergent. However, the implicit scheme is computationally expensive.

## WORKING OF THE CODE

1. For implicit scheme:



```
#Implicit scheme matrix
r = al*dt/(dx*dx)
A = np.zeros((nx,nx))
A[0,0] = 1.0
A[-1,-1] = 1.0
for i in range(1,nx-1):
    A[i,i] = 1.0 + 2.0*r
    A[i,i-1] = -r
    A[i,i+1] = -r
B = np.linalg.inv(A)

# temperature at each time step
for n in range(1,nt):
    T[:,n] = np.dot(B,T[:,n-1])
```

Fig: Computational step for the implicit scheme

After obtaining the relation between successive elements of the temperature array, the iterative method was used to calculate the values at each position of the rod during a particular time interval. The calculation in the implicit scheme was further simplified by modelling the current problem into a system of equations, which was later solved using matrices.

If we let $\lambda = \alpha^2(\frac{k}{h^2})$,

$$\boldsymbol{w}^{(0)} = (f(x_1), f(x_2), \dots, f(x_{m-1}))^t,$$

$$\boldsymbol{w}^{(j)} = (w_{1,j}, w_{2,j}, \dots, w_{m-1,j})^t,$$

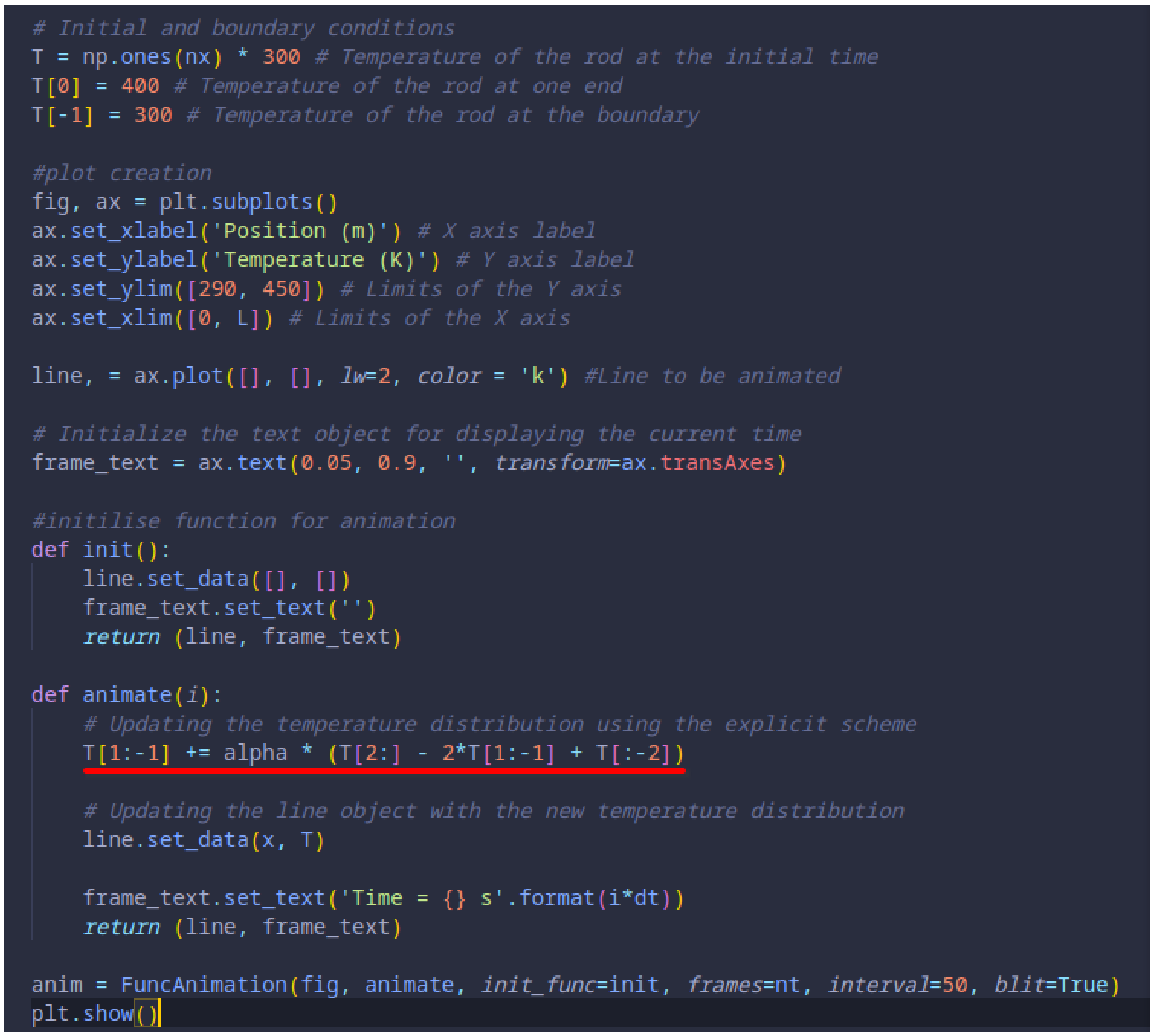
for each $j = 1, 2, \dots$, and $A = \begin{bmatrix} (1-2\lambda) & \lambda & 0 \dots\dots\dots & 0 \\ \lambda & (1-2\lambda) & \lambda \dots\dots\dots & \dots\dots \\ 0 & .. & .. & 0 \\ .. & .. & .. & \lambda \\ 0 & 0 & \lambda & (1-2\lambda) \end{bmatrix}$, then (e) is

equivalent to

$$\boldsymbol{w}^{(j)} = A\boldsymbol{w}^{(j-1)},$$

The above shows a similar conversion, where λ corresponds to β, $\alpha^2$ to al, k to dt, h to dx, and w to T in our code. This formula was used to compute the temperature.

2. For explicit scheme:



```
# Initial and boundary conditions
T = np.ones(nx) * 300 # Temperature of the rod at the initial time
T[0] = 400 # Temperature of the rod at one end
T[-1] = 300 # Temperature of the rod at the boundary

#plot creation
fig, ax = plt.subplots()
ax.set_xlabel('Position (m)') # X axis label
ax.set_ylabel('Temperature (K)') # Y axis label
ax.set_ylim([290, 450]) # Limits of the Y axis
ax.set_xlim([0, L]) # Limits of the X axis

line, = ax.plot([], [], lw=2, color = 'k') #Line to be animated

# Initialize the text object for displaying the current time
frame_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)

#initilise function for animation
def init():
    line.set_data([], [])
    frame_text.set_text('')
    return (line, frame_text)

def animate(i):
    # Updating the temperature distribution using the explicit scheme
    T[1:-1] += alpha * (T[2:] - 2*T[1:-1] + T[:-2])

    # Updating the line object with the new temperature distribution
    line.set_data(x, T)

    frame_text.set_text('Time = {} s'.format(i*dt))
    return (line, frame_text)

anim = FuncAnimation(fig, animate, init_func=init, frames=nt, interval=50, blit=True)
plt.show()
```

In case of solving using the explicit scheme, the iterative method can be implemented directly using loops, because of the fewer computations involved. For ease of understanding, both codes were animated over a series of time. The iterative calculation for the explicit scheme has been implemented in the animate function, underlined in red in the above image.

## RESULTS

### 1. Implicit scheme output:

The output we get is in the form of an animated graph. We can see a curve that starts from (0,400), going till (2,300), having a slope of almost negative infinity at t=0. As time passes, the curve transforms into a straight line. Hence, after a long time, we observe that the temperature curve becomes linear. This follows the heat equation for the one-dimensional rod. The results are given below:

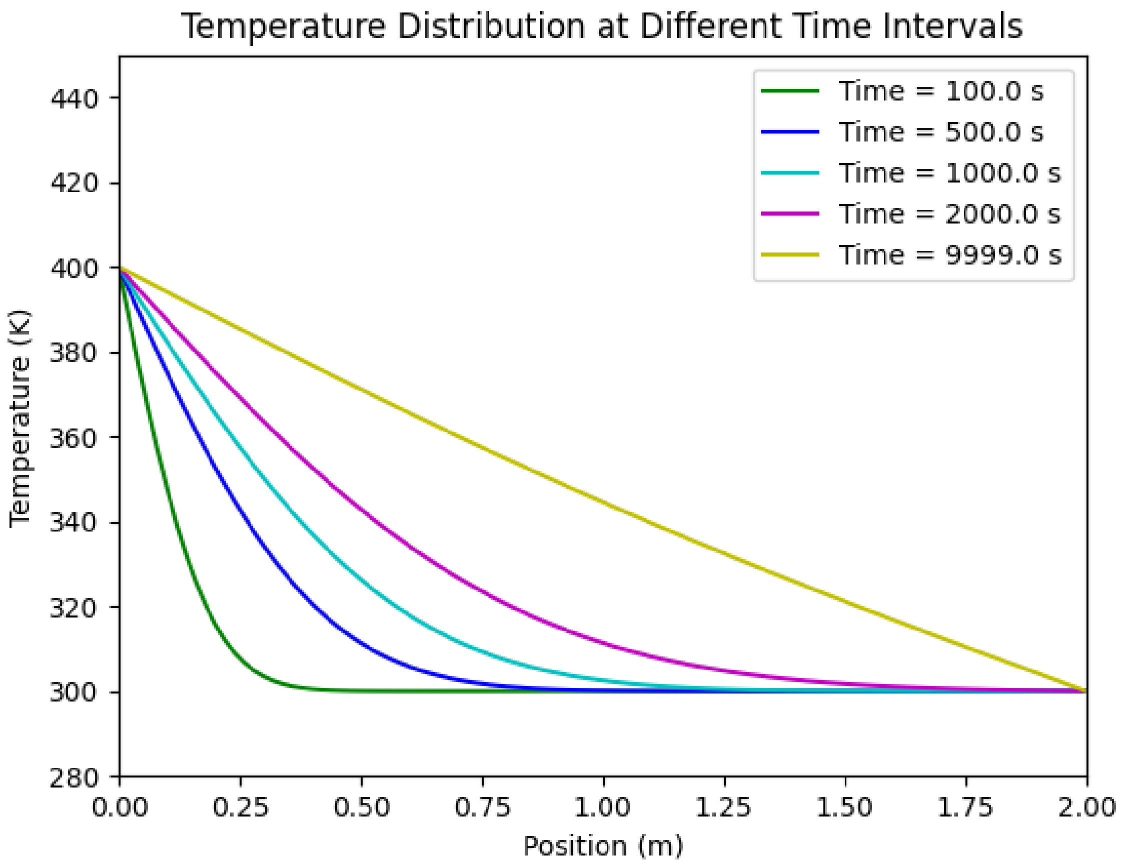


Fig: Image at multiple times

As we can see, the longer the time is, the more linear the curve becomes as it approaches steady state.

## 2. Explicit scheme output:

The output we get is the same animated curve, from (0,400), going to (2,300). So, we get the same results as the implicit curve. The results are given below:

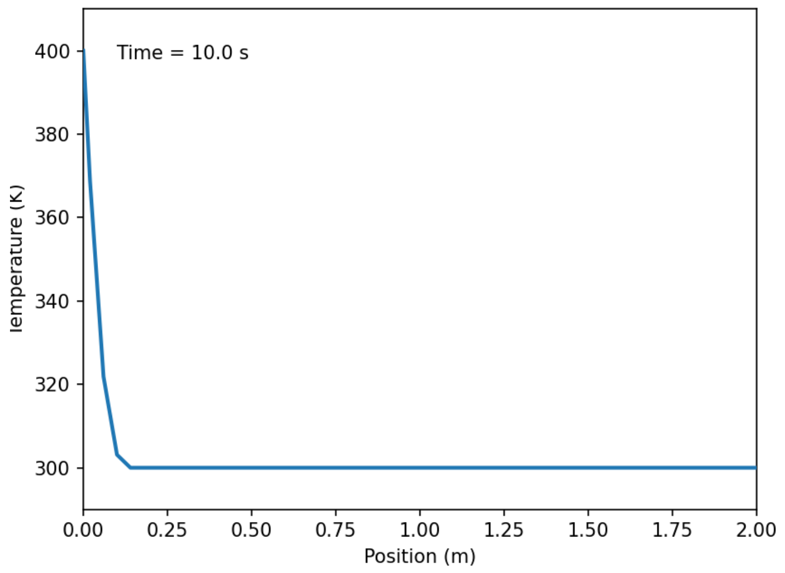


Fig: Image taken at initial time

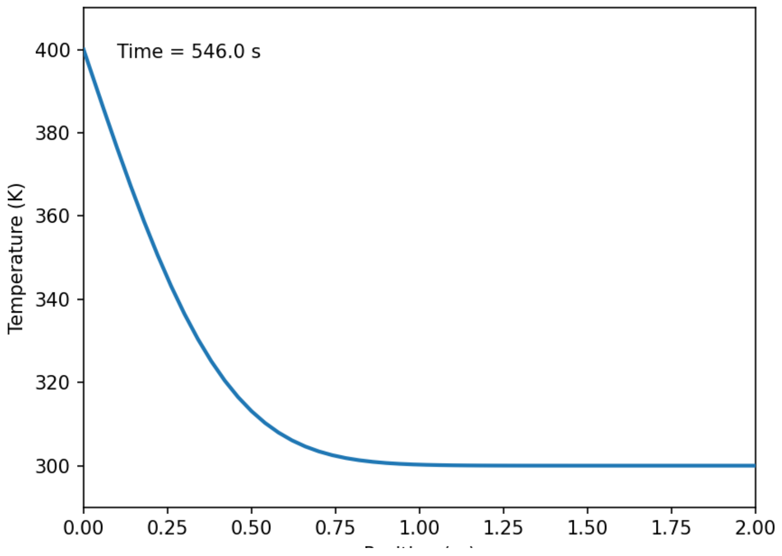


Fig: Image taken at an arbitrary time t

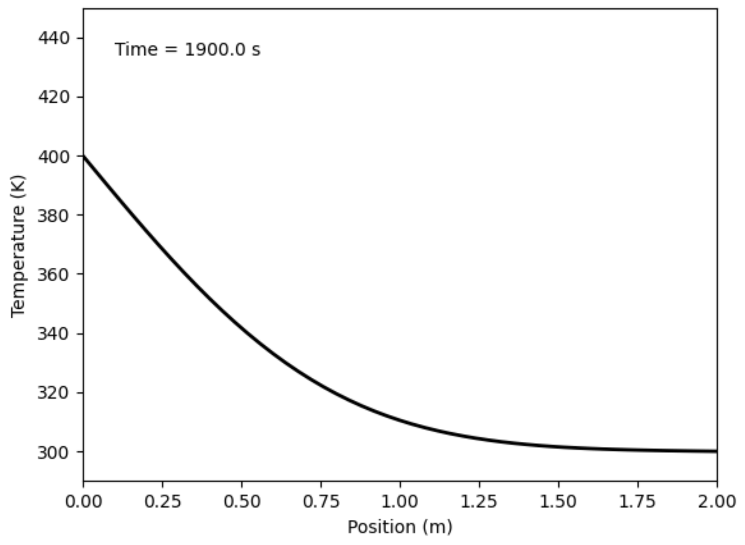


Fig: Image taken at near end time

## References

1. https://opencommons.uconn.edu/cgi/viewcontent.cgi?article=1118&context=srhonors_theses
2. https://www.youtube.com/@kvyi/featured