Name - Biswarup Karmakar
SR: 21055

Problem ①

ⓐ Given a rectangular plate $R = \{(x,y) : 2 \le x \le 3,\ 4 \le y \le 6\}$ with the heat conduction eqn $\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$

$T(0,y) = 30$ $\quad \frac{\partial T}{\partial y}(x,4) = 0, \quad \frac{\partial T}{\partial y}(x,6) = T(x,6) - 60$

$T(1,y) = 60$

Here we are using 2nd order central difference for 128 x 256 grid.

The step size $\Delta x = \frac{1}{127}, \quad \Delta y = \frac{2}{255}$

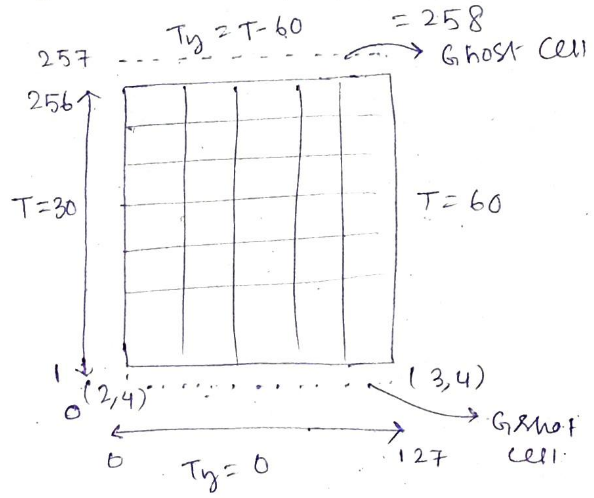
There are two extra rows added due to boundary condition (neuman).

Then number of points in x-direction TNX = 128

and " " " " y- " TNY = 256+2 = 258

Using 2nd central difference on $T_{xx} + T_{yy} = 0$

we have



$$\frac{T(i+1,j) - 2T(i,j) + T(i-1,j)}{(\Delta x)^2} + \frac{T(i,j+1) - 2T(i,j) + T(i,j-1)}{(\Delta y)^2} = 0$$

$$\Rightarrow \boxed{T(i-1,j) + (-2-2r)T(i,j) + T(i+1,j) + r\,T(i,j+1) + r\,T(i,j-1) = 0}$$

where $r = \frac{\Delta x^2}{\Delta y^2}$ $\quad$ for $i = 1:126,\ j = 1:256$ ——①

For right boundary condition,
$T(127,j) = 60$ for $j = 1$ to 256 ——②

For Left boundary condition
$T(0,j) = 30$ for $j = 1$ to 256 ——③

The other condition are
$T_y(x,4) = 0$

$$\Rightarrow \frac{T(i,i+1) - T(i,1-1)}{2\Delta y} = 0 \quad (j=1)$$ (using central difference w.r.t y variable)

$$\Rightarrow \boxed{T(i,0) - T(i,2) = 0 \quad \text{for } i=0 \text{ to } 127} \quad \text{——} \; (4)$$

Now $T_y(x,6) = T(x,6) - 60$

using central difference here

$$\frac{T(i,256+1) - T(i,256-1)}{2\Delta y} = T(i,256) - 60 \quad (\text{for } j=256)$$

$$\Rightarrow T(i,255) + 2\Delta y\, T(i,256) - T(i,257) = 120\Delta y \quad (\text{for } i=0{:}127)$$
—— (5)

(6) Now we get the matrix A of order 33204 x 33204

$$
\begin{array}{c}
128 \text{ rows} \\ \\ \\
128 \text{ rows} \\ \\ \\ \\
128 \text{ rows} \\ \\
\end{array}
\left[\begin{array}{ccccccccc}
1 & 0 & 0 & \cdots & -1 & 0 & \cdots & & \\
0 & 1 & 0 & \cdots & 0 & -1 & \cdots & & \\
 & & 1 & \ddots & & -1 & \ddots & & \\
\hline
0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & \\
0 & \cdots & r & \cdots & 1 & (-2-2r) & 1 & \cdots r & \cdots \\
0 & \cdots & & r & \cdots 1 & & (-2-2r) & 1 \cdots & r \cdots \\
\multicolumn{9}{c}{\text{same structure repeated 256 times}} \\
\hline
0 & \cdots & 0 & \cdots 1 & \cdots & 2\Delta y & \cdots & -1 & \cdots \\
\cdots & & \cdots & & 1 & & 2\Delta y & \cdots -1 & \cdots \\
\end{array}\right]
\left[\begin{array}{c}
T(0,0) \\ \vdots \\ T(127,0) \\ \hline T(0,1) \\ \vdots \\ T(127,1) \\ \vdots \\ \hline T(0,257) \\ T(1,257) \\ \vdots \\ T(127,257)
\end{array}\right]
=
\left[\begin{array}{c}
0 \\ \vdots \\ 0 \\ \hline 30 \\ 0 \\ \vdots \\ 60 \\ \hline \vdots \\ 120\Delta y \\ \vdots \\ 120\Delta y
\end{array}\right]
$$

A

Now First 128 rows corresponds to
$T(i,0) - T(i,2) = 0$

~~Now~~ for the interior region 128 rows is repeated ~~Now~~ 256 times satisfying condition (1), (2), (3)

Last 128 rows corresponds to eq^n (5).

Then we got a sparse matrix. Using Lapack for 128 x 256 grid it takes almost 90 minutes to run.

This is a C++ program that solves a 2D heat equation using finite difference method and LAPACK library. The program starts by defining the problem domain, including its limits and spacing. It then initializes the matrix A and vector b with zeros and applies boundary conditions. The program then calls the LAPACK library function dgesv_ to solve the system of linear equations represented by matrix A and vector b. The solution vector x is then written to a file.

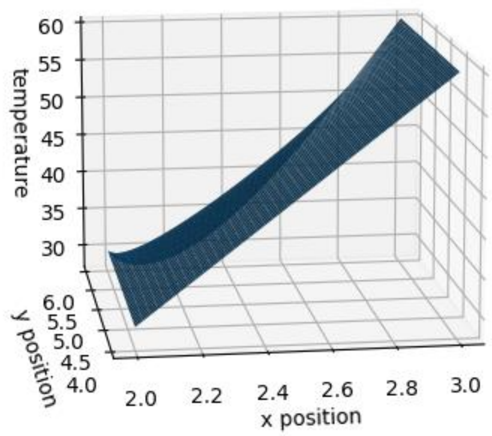
The finite difference method is used to discretize the heat equation, and the resulting system of linear equations is represented by a matrix A and vector b. The matrix A is a sparse matrix, and the LAPACK library function dgesv_ is used to solve the system efficiently.

The boundary conditions for the problem are as follows:

- Left boundary: Temperature is fixed at 30.0 degrees Celsius.
- Right boundary: Temperature is fixed at 60.0 degrees Celsius.
- Bottom boundary: Temperature gradient is zero.
- Top boundary: The rate of heat flow out of the top of the plate is proportional to the temperature difference between the top of the plate and the environment, and the constant of proportionality is 60.

The program uses dynamic memory allocation to create the matrix A and vector b, and it also checks whether the memory allocation was successful or not. The program also writes the solution vector x to a file named "temperature_values.txt".

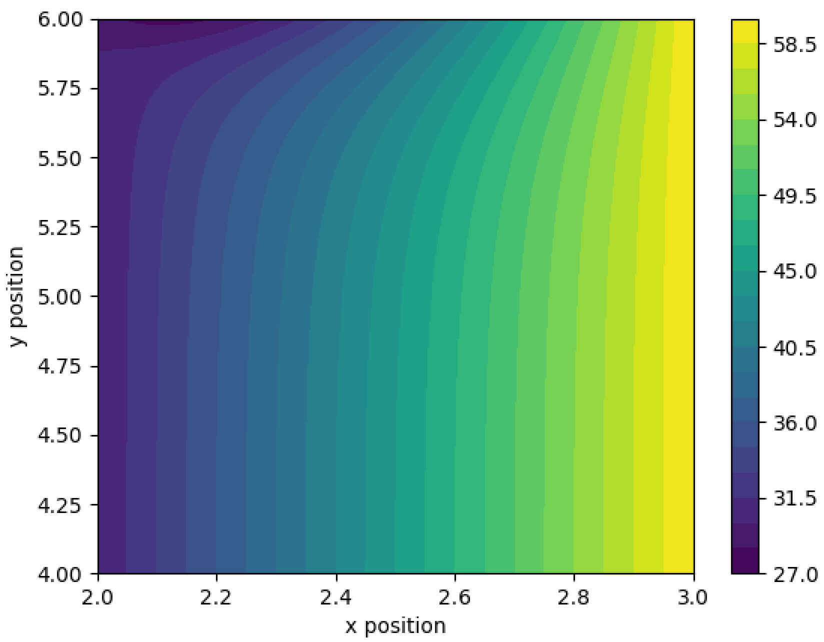
Overall, the program demonstrates the use of the finite difference method and LAPACK library to solve a 2D heat equation problem.



1(c):

Here we have give the surface plot of that equation using temperature distribution. It also satisfies all the boundary condition and intial condition given in the problem.

1(d): The contour plot of the temperature distribution obtained from the above code shows how the temperature varies in the rectangular domain of the problem. The contour lines are curves that connect points of equal temperature, so they provide a visual representation of the temperature field. Typically, contour plots use different colors or shades to represent different temperature levels.



Problem-2

Let's consider the diffusion equation:

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

where u = dependent variable
t = time, x = spatial co-ordinate
α = diffusion co-efficient

The two level scheme with $\theta \geq 0.5$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \alpha \left[\theta \frac{(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1})}{(\Delta x)^2} + (1-\theta) \frac{(u_{j+1}^n - 2u_j^n + u_{j-1}^n)}{(\Delta x)^2} \right.$$

putting $\alpha \frac{\Delta t}{(\Delta x)^2} = r_d$

Manipulating terms we get,

$$-r_d \theta u_{j-1}^{n+1} + (1+2r_d\theta) u_j^{n+1} - r_d \theta u_{j+1}^{n+1} =$$
$$r_d(1-\theta) u_{j-1}^n + (1-2r_d(1-\theta)) u_j^n + r_d(1-\theta) u_{j+1}^n \quad \text{--- (1)}$$

Here we use von-neuman stability analysis by using $u_j^n = v^n e^{ikx_j}$ [i = complex root of (-1)]

putting this on (1)

$$-r_d \theta v^{n+1} e^{-ik\Delta x} + (1+2r_d\theta) v^{n+1} - r_d \theta v^{n+1} e^{ik\Delta x}$$
$$= r_d(1-\theta) v^n e^{-ik\Delta x} + (1-2r_d(1-\theta)) v^n + r_d(1-\theta) v^n e^{ik\Delta x}$$

(cancelling e^{x_j} in both sides)

$$\Rightarrow -r_d \theta \cdot v \cdot 2\cos(k\Delta x) + v(1+2r_d\theta)$$
$$= r_d(1-\theta) \cdot 2\cos(k\Delta x) + (1-2r_d(1-\theta))$$

$$\Rightarrow v = \frac{r_d(1-\theta)(2\cos k\Delta x - 2) + 1}{r_d\theta(2 - 2\cos k\Delta x) + 1}$$

for decaying solution (stable solution)

$$\left| \frac{u_j^{n+1}}{u_j^n} \right| = |M| \leq 1$$

That is

$$\boxed{-1 \leq \frac{r_d(1-\theta)(2\cos k\Delta x - 2) + 1}{r_d\theta(2 - 2\cos k\Delta x) + 1} \leq 1}$$

Then we ~~got~~ have to show that it is independent of r_d

$$-1 \leq \frac{4r_d(\theta-1)\sin^2\left(\frac{K\Delta x}{2}\right)+1}{4r_d\,\theta\,\sin^2\left(\frac{K\Delta x}{2}\right)+1} \leq 1$$

For RHS, Now $r_d > 0$, $\sin^2\left(\frac{K\Delta x}{2}\right) > 0$, $\theta \geq 0.5$

$$4r_d(\theta-1)\sin^2\left(\frac{K\Delta x}{2}\right)+1 \leq 4r_d\,\theta\,\sin^2\left(\frac{K\Delta x}{2}\right)+1$$

so, one condition trivially satisfied.

For LHS,

$$-4r_d\,\theta\,\sin^2\left(\frac{K\Delta x}{2}\right)-1 \leq 4r_d(\theta-1)\sin^2\left(\frac{K\Delta x}{2}\right)+1$$

$$\Rightarrow +4r_d\sin^2\left(\frac{K\Delta x}{2}\right)(-\theta-\theta+1) \leq 2$$

$$\Rightarrow 4r_d\sin^2\left(\frac{K\Delta x}{2}\right)(1-2\theta) \leq 2$$

Now $\sin^2\left(\frac{K\Delta x}{2}\right) > 0$, $\theta \geq 0.5$, for $r_d > 0$

The above condition holds and it is independent of r_d

So The above method is unconditionally stable problem (3)

The diffusion equation is $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$

Now we discretize the above eq^n using explicit Euler and 2nd order central ~~eq~~^{n} difference

Then $\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t}$ (Discretizing w.r.t time variable)

$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$ (Discretizing w.r.t spatial variable)

i.e, we got

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha\left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\right] \quad \text{——— (1)}$$

The process of deriving the modified eq^n is very similar to computing the local truncation error.

Using Taylor series expansion we have

$$u_i^{n+1} = u_i^n + \Delta t \frac{\partial u_i^n}{\partial t} + \frac{\Delta t^2}{2!}\frac{\partial^2 u_i^n}{\partial t^2} + \frac{\Delta t^3}{3!}\frac{\partial^3 u_i^n}{\partial t^3} + \cdots$$

$$\Rightarrow \frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\partial u_i^n}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 u_i^n}{\partial t^2} + \frac{\Delta t^2}{6}\frac{\partial^3 u_i^n}{\partial t^3} + \cdots \quad \text{——— (2)}$$

Again $u_{i+1}^n = u_i^n + \Delta x \frac{\partial u_i^n}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 u_i^n}{\partial x^2} + \frac{\Delta x^3}{3!}\frac{\partial^3 u_i^n}{\partial x^3} + \cdots$

$u_{i-1}^n = u_i^n - \Delta x \frac{\partial u_i^n}{\partial x} + \frac{\Delta x^2}{2!}\frac{\partial^2 u_i^n}{\partial x^2} - \frac{\Delta x^3}{3!}\frac{\partial^3 u_i^n}{\partial x^3} + \cdots$

from the above equations

$$\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = \frac{\partial^2 u_i^n}{\partial x^2} + \frac{2}{4!}\Delta x^2 \frac{\partial^4 u_i^n}{\partial x^4} + \cdots$$

$$= \frac{\partial^2 u_i^n}{\partial x^2} + \frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} + \cdots \quad \text{——— (3)}$$

Substituting (2) and (3) in (1)

$$\frac{\partial u_i^n}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 u_i^n}{\partial t^2} + \frac{\Delta t^2}{6}\frac{\partial^3 u_i^n}{\partial t^3} + \cdots = \alpha\left[\frac{\partial^2 u_i^n}{\partial x^2} + \frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} + \cdots\right]$$

$$\Rightarrow \frac{\partial u_i^n}{\partial t} - \alpha \frac{\partial^2 u_i^n}{\partial x^2} = \alpha \frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} - \frac{\Delta t}{2}\frac{\partial^2 u_i^n}{\partial t^2} + \alpha \frac{\Delta x^4}{360}\frac{\partial^6 u_i^n}{\partial x^6} - \frac{\Delta t^2}{6}\frac{\partial^3 u_i^n}{\partial t^3} + \cdots \quad \text{——— (4)}$$

This is the modified equation

where
Truncation error (TE)

$$= \alpha \frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} + \alpha \frac{\Delta x^4}{360}\frac{\partial^6 u_i^n}{\partial x^6} - \frac{\Delta t}{2}\frac{\partial^2 u_i^n}{\partial t^2} - \frac{\Delta t^2}{6}\frac{\partial^3 u_i^n}{\partial t^3} + \cdots$$

Now we have to represent time derivative in terms of spatial derivative

Derivating eqn ④ w.r.t t.

$$\frac{\partial^2 u_i^n}{\partial t^2} - \alpha\frac{\partial^3 u_i^n}{\partial t\partial x^2} = \alpha\frac{\Delta x^2}{12}\frac{\partial^5 u_i^n}{\partial x^4 \partial t} + \alpha \frac{\Delta x^4}{360}\frac{\partial^7 u_i^n}{\partial x^6\partial t} - \frac{\Delta t}{2}\frac{\partial^3 u_i^n}{\partial t^3} - \frac{\Delta t^2}{6}\frac{\partial^4 u_i^n}{\partial t^4} + \cdots \quad \text{⑤}$$

Double differentiating w.r.t x, in eqn ④

$$\frac{\partial^3 u_i^n}{\partial x^2 \partial t} - \alpha\frac{\partial^4 u_i^n}{\partial x^4} = \alpha\frac{\Delta x^2}{12}\frac{\partial^6 u_i^n}{\partial x^6} + \alpha\frac{\Delta x^4}{360}\frac{\partial^8 u_i^n}{\partial x^8} - \frac{\Delta t}{2}\frac{\partial^4 u_i^n}{\partial x^2 \partial t^2} - \frac{\Delta t^2}{6}\frac{\partial^6 u_i^n}{\partial x^3 \partial t^3} \cdots \quad \text{⑥}$$

Now eqn ⑤ + α ⑥ gives.

$$\frac{\partial^2 u_i^n}{\partial t^2} - \alpha^2\frac{\partial^4 u_i^n}{\partial x^4} = O(\Delta x^2, \Delta t)$$

substituting this in error term we get

$$T.E = \alpha\frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} + \frac{\alpha \Delta x^4}{360}\frac{\partial^6 u_i^n}{\partial x^6} - \frac{\Delta t}{2}\left[\alpha^2 \frac{\partial^4 u_i^n}{\partial x^4} + O(\Delta t, \Delta x^2)\right] - \frac{\Delta t^2}{6}\left[\alpha^3\frac{\partial^6 u_i^n}{\partial x^6} + O(\Delta t, \Delta x^2)\right]$$

$$\Rightarrow TE = \alpha\frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} - \frac{\Delta t}{2}\alpha^2\frac{\partial^4 u_i^n}{\partial x^4} + O(\Delta t^2, \Delta x^4)$$

where $\Delta t = \frac{r_d \Delta x^2}{\alpha}$

ⓑ

$$TE = \alpha\frac{\Delta x^2}{12}\frac{\partial^4 u_i^n}{\partial x^4} - \frac{r_d \Delta x^2}{2\alpha}\alpha^2\frac{\partial^4 u_i^n}{\partial x^4} + O(\Delta t^2, \Delta x^4)$$

$$= \alpha\frac{\Delta x^2}{2}\left(\frac{1}{6} - r_d\right)\frac{\partial^4 u_i^n}{\partial x_i^4} + O(\Delta t^2, \Delta x^4)$$

We know that if ~~error~~ leading order of Truncation error contains even order derivatives then, it has dissipative error. That is it has artificial decay of solution, magnitude/amplitude of error decreases, the function would smear out.

On the other hand, if leading order of TE has odd order then it has ~~dissipation~~ disspersive error. It is characterised by oscillation or small deviations in the solution, waves of different length propagate or different speed.

In the given equation leading order term has only even order derivatives. So only dissipative errors are present and ~~dissipative~~ disspersive errors are not present.

problem (4) (a)

Here the diffusion eq^n

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \quad \text{with } \alpha = 1.5 \qquad \text{--- (1)}$$

The boundary condition: $u(0,t) = u(2\pi, t) = 0$

(periodic boundary condition)

and the initial condition $u(x,0) = \sin 4x + \sin x$

From method of separation of variable

$$u(x,t) = F(x).G(t)$$

substituting this in eq^n (1) we got

$$F(x)\,\dot{G}(t) = \alpha F''(x)\,G(t)$$

consider $\frac{\dot{G}}{\alpha G} = \frac{F''}{F} = -K$ (costant)

It can be split into 2 ODE's

$$\dot{G} + K\alpha G = 0 \quad \Big| \quad F'' + KF = 0$$

consider 2nd ode for $K<0$ (case I)

then $F(x) = C_1 e^{\sqrt{-K}x} + C_2 e^{-\sqrt{-K}\,x}$ ($C_1, C_2 \to$ constant)

using boundary condition

$$F(0) = C_1 + C_2 = 0$$

$$F(2\pi) = C_1 e^{+\sqrt{-K}\,2\pi} + C_2 e^{-\sqrt{-K}\,2\pi} = 0 \qquad \Rightarrow C_1 = C_2 = 0$$

This gives a trivial solution.

(case-II) if $K=0$,

$$F(x) = C_1 + C_2 x$$

$$F(0) = C_1 = 0$$

$$F(2\pi) = C_2 . 2\pi = 0$$

This also gives a trivial solution.

(case III) so we must take $K>0$, suppose $K = b^2$,

Then $F(x) = A\cos bx + B\sin bx$ ($A, B \to$ Constant)

Boundary condition implies

$$F(0) = 0 \Rightarrow A = 0$$

$$F(2\pi) = 0 \Rightarrow B \sin b\,2\pi = 0$$

$$\Rightarrow \sin b . 2\pi = \sin n\pi \quad (n \in \mathbb{N})$$

$$\Rightarrow b = \frac{n}{2} \quad n \in \{1, 2, \ldots\}$$

we consider general solution:

$$F_n(x) = B_n \sin\left(\frac{nx}{2}\right)$$

from 1st ODE

$$\dot{G} + \frac{n^2}{4} \alpha G = 0$$

$$\Rightarrow \dot{G} + \lambda_n^2 G = 0 \qquad \left(\lambda_n = \frac{n}{2}\sqrt{\alpha} \right)$$

Then $G_n(t) = K_n e^{-\lambda_n^2 t}$

Hence general solution has n-th term as

$$U_n(x,t) = F_n(t)\, G_n(t)$$
$$= B_n \sin\left(\frac{nx}{2}\right) K_n e^{-\lambda_n^2 t}$$
$$= D_n \sin\left(\frac{nx}{2}\right) e^{-\lambda_n^2 t} \qquad (\text{suppose } D_n = K_n \cdot B_n)$$

Then $u(x,t) = \sum_{n=1}^{\infty} D_n \sin\left(\frac{nx}{2}\right) e^{-\lambda_n^2 t}$

$$= \sum_{n=1}^{\infty} D_n \sin\left(\frac{nx}{2}\right) e^{-\frac{n^2}{4}\alpha t}$$

Now $u(x,0) = \sin 4x + \sin x$

so, $D_8 = 1$, $D_2 = 1$ All other D_n's are zero.

The analytical solution

$$u(x,t) = \sin 4x \; e^{-\frac{8^2}{4} \times 1.5 \times t} + \sin x \; e^{-\frac{2^2}{4} \times 1.5 \times t}$$

$$= \boxed{\sin(4x)\, e^{-24t} + \sin x\, e^{-1.5t}}$$

4(e) Here we got abnormal order of accuracy for $r_d = \frac{1}{6}$

From problem 3.(b)

we have Truncation error

$$TE = \alpha \frac{\Delta x^2}{2} \left(\frac{1}{6} - r_d \right) \frac{\partial^4 u_i^n}{\partial x_i^4} + O(\Delta t^2, \Delta x^4)$$

That is for $r_d = \frac{1}{6}$, $TE = O(\Delta t^2, \Delta x^4)$

when N is small, Δx is large so

$O(\Delta x^4)$ term is dominant

when N increases, Δx decreases and Δt increases

so dominant term order decreases.

This leads to reduction in slope.

On the other sider for $r_d = \frac{1}{2}$,

dominant term in TE contains Δx^2, so

we got order of accuracy $= 2$ for 1st case.

4(b): The code provided is a C++ implementation of the numerical solution of the one-dimensional diffusion equation using the explicit euler and central finite difference method. The program calculates the solution for different grid sizes and time step sizes and compares the calculated solutions with the analytical solution to evaluate the accuracy of the numerical method. The calculated errors are then written into a file named "error_matrix.txt".

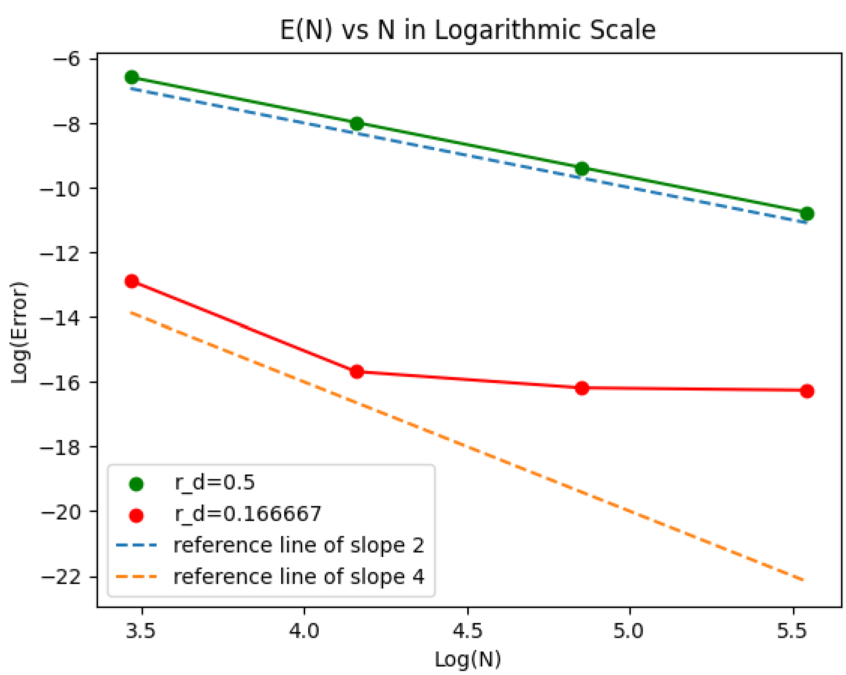
Here we used the discretized form from problem(3) as

$$u_i^{n+1} = u_i^n + r_d[u_{i+1}^n - 2u_i^n + u_{i-1}^n]$$

4(c): Here we are giving the average error for different values of **r_d** and **N**

N	Average absolute error for CFL=0.5	Average absolute error for CFL=0.166667
32	**0.00138943**	**2.557540e-06**
64	**3.416820e-04**	**1.551477e-07**
128	**8.475753e-05**	**9.382825e-08**
256	**2.104395e-05**	**8.719156e-08**

4(d) Here the plotting between Error vs number of grid sizes in logarithmic scale



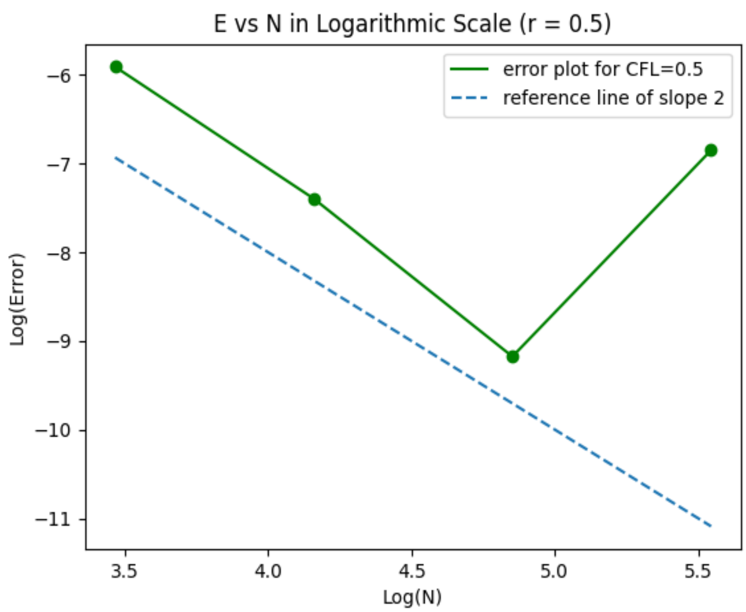
Here For the first plot (that is for CFL=0.5) the slope is **-2.0149812717521134.** Now since N-1=L/dx, we got the order of accuracy=2. This means the truncation error is propotional to $(dx)^2$.

As we can see, for the 2nd case that is for CFL=0.166667,we don't observe a straight line. But we see that order of accuracy for first segment line is around **-4.043042589154455**, for second segment was **-0.7255480468636528**, and for third segment was **-0.10583386644905732**. We observe that for the first segment order of accuracy 4 was larger and it gradually decreases for the next segment.

4(e): The explicit Euler method is conditionally stable for solving the diffusion equation. This means that there is a limit to the time step size that can be used, beyond which the solution becomes unstable and the errors grow uncontrollably. Specifically, the maximum time step size that can be used with the explicit Euler method is proportional to the square of the spatial step size. Therefore, as the mesh size is refined, the time step size must be decreased accordingly to maintain stability. If the time step size is not reduced appropriately, the solution becomes unstable and the errors grow rapidly. As a result, the error plot for the explicit Euler method will show an increasing trend as the mesh size is refined, unless the time step size is also decreased appropriately.

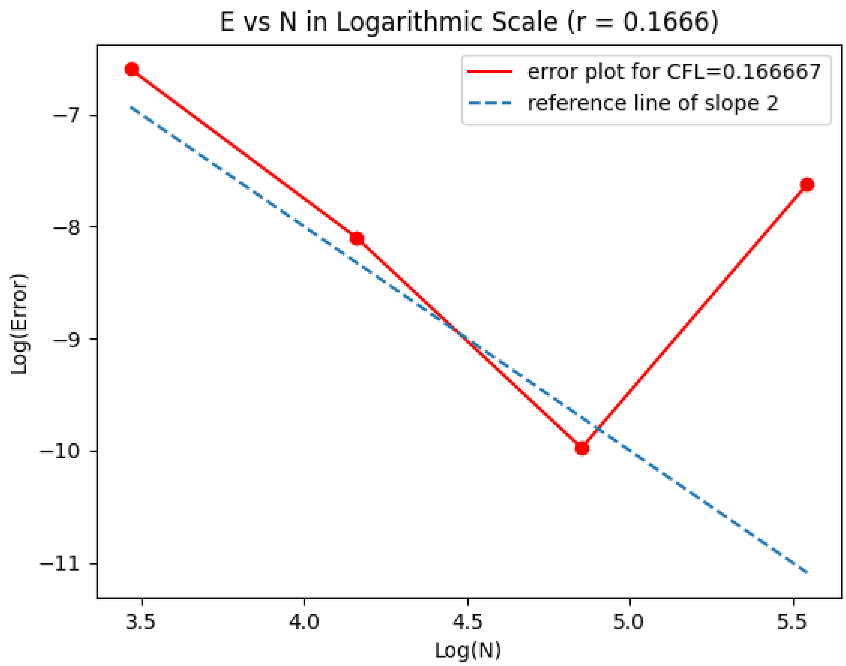
5(a): Here we implemented implicit Euler scheme for the diffusion equation with different CFL number r_d grid sizes N .

N	Average Absolute error for CFL=0.5	Average Absolute error for CFL=0.166667
32	0.00271848	1.369204e-03
64	6.142116e-04	3.052546e-04
128	1.035393e-04	4.648890e-05
256	1.060109e-03	4.892379e-04

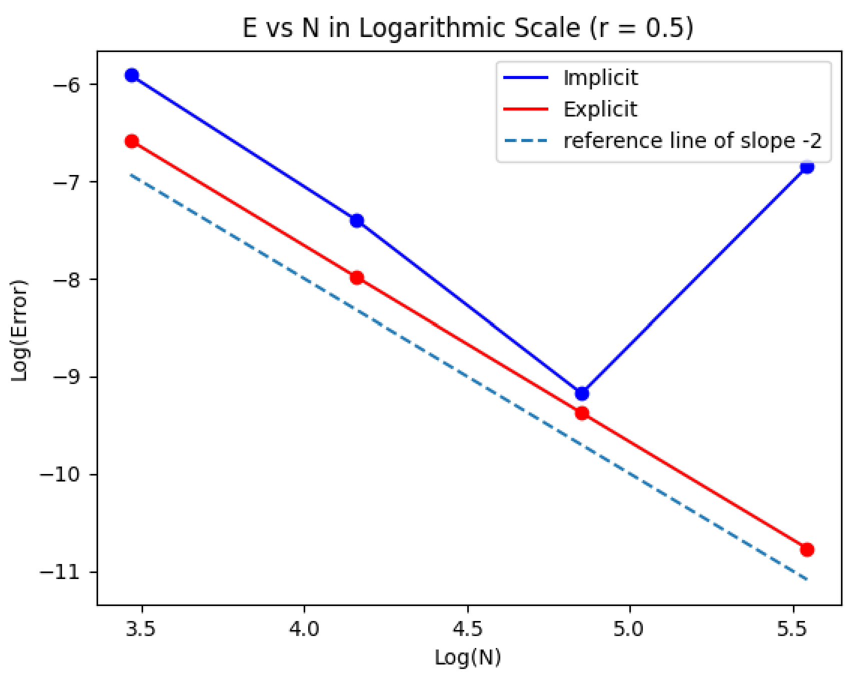


The program is a C++ code that solves the one-dimensional diffusion equation using an implicit Euler scheme and Jacobi algorithm. The heat equation describes how temperature changes with time and space in a medium. The program uses finite difference method to discretize the space and time dimensions. The analytical solution of the heat equation is computed, and the absolute error between the analytical and predicted solutions is calculated.

Here we used the discreteizatio form as $-r_d u_{i+1}^{n+1} + (1 + 2r_d)u_i^{n+1} - r_d u_{i-1}^{n+1} = u_i^n$. Then we got the tridiagonal system $AU^{n+1} = U^n$ using proper boundary condition.



5(b): Comparison between the error with explicit method and euler method:



By using the explicit Euler approach, we see that error consistently drops as mesh size increases, however when using the implicit Euler method, error initially decreases and subsequently climbs as mesh size increases.

The effect of dx is crucial in the explicit Euler method, because error diminishes when Δx (or N is increased) is reduced. However, in the implicit Euler approach, for constant stability constant r_d, dt (time domain gap) becomes coarser with decreasing mesh size, which might result in inaccurate approximations in time axis leading to bad performance when N increases .The initial decreasing error part of curve is due to finer mesh in spatial domain leading to better result, however with decreasing mesh size, dt gets larger and larger leading poor performance that resulted in increase of error as N increased in case of implicit Euler scheme.

It is not uncommon for the error to increase with a finer mesh size in the case of the implicit Euler method for solving the diffusion equation. This is because the implicit Euler method involves solving a system of linear equations at each time step, and as the mesh size becomes finer, the size of the system of equations also increases. Solving larger systems of equations can be computationally more expensive and less accurate than solving smaller systems.

problem 6

The transient 1D heat conduction eq^n.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$ where $T(x,t)$ = temperature, α = thermal ~~conductivity~~ diffusivity

The eq^n is approximated using

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^{n-k} - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$ where $T_i^n = T(x_i, t_n)$

Δx = grid spacing, Δt = time step.

(a) we use Taylor series expansion for T_{i+1}^{n-k} using two variable.

$$T_{i+1}^{n-k} = T(x_i + \Delta x, t_n - k\Delta t)$$

$$= T_i^n + \Delta x \frac{\partial T_i^n}{\partial x} + (-k\Delta t)\frac{\partial T_i^n}{\partial t} + \frac{1}{2!}\left[(\Delta x)^2 \frac{\partial^2 T_i^n}{\partial x^2} + 2(-k\Delta t)\Delta x \frac{\partial^2 T_i^n}{\partial x \partial t} + (-k\Delta t)^2 \frac{\partial^2 T_i^n}{\partial t^2}\right] + \frac{1}{3!}\left[(\Delta x)^3 \frac{\partial^3 T_i^n}{\partial x^3}\right] + \frac{1}{2!}(-k\Delta t)^2 \Delta x \frac{\partial^3 T_i^n}{\partial x \partial t^2} + \frac{1}{2!}(-k\Delta t)(\Delta x)^2 \frac{\partial^3 T_i^n}{\partial x^2 \partial t} + \frac{1}{3!}(-k\Delta t)^3 \frac{\partial^3 T_i^n}{\partial t^3} + \cdots$$

similiarly

$$T_{i-1}^n = T(x_i - \Delta x, t_n + 0)$$

$$= T_i^n - \Delta x \frac{\partial T_i^n}{\partial x} + \frac{1}{2!}(\Delta x)^2 \frac{\partial^2 T_i^n}{\partial x^2} - \frac{1}{3!}(\Delta x)^3 \frac{\partial^3 T_i^n}{\partial x^3} + \cdots$$

Then RHS transformed to

$$\alpha\left[\frac{T_{i+1}^{n-k} - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}\right]$$

$$= \frac{\alpha}{(\Delta x)^2}\left[-k\Delta t \frac{\partial T_i^n}{\partial t} + \frac{1}{2}(k\Delta t)^2 \frac{\partial^2 T_i^n}{\partial t^2} - (k\Delta t)\Delta x \frac{\partial^2 T_i^n}{\partial x \partial t} + (\Delta x)^2 \frac{\partial^2 T_i^n}{\partial x^2} + \frac{1}{2}k^2(\Delta t)^2 \Delta x \frac{\partial^3 T_i^n}{\partial x \partial t^2} + \frac{1}{2}(-k\Delta t)(\Delta x)^2 \frac{\partial^3 T_i^n}{\partial x^2 \partial t} + \frac{1}{6}(-k\Delta t)^3 \frac{\partial^3 T_i^n}{\partial t^3} + \cdots\right] \quad (1)$$

For LHS, we represent T_i^{n+1} in terms of T_i^n

$$T_i^{n+1} = T_i^n + \Delta t \frac{\partial T_i^n}{\partial t} + \frac{(\Delta t)^2}{2!}\frac{\partial^2 T_i^n}{\partial t^2} + \frac{(\Delta t)^3}{3!}\frac{\partial^3 T_i^n}{\partial t^3} + \cdots$$

$$\Rightarrow \frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{\partial T_i^n}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 T_i^n}{\partial t^2} + \frac{(\Delta t)^2}{6}\frac{\partial^3 T_i^n}{\partial t^3} + \cdots \quad (2)$$

Now from ①, ② equating both sides.

we have

$$\frac{\partial T_i^n}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 T_i^n}{\partial t^2} + \frac{(\Delta t)^2}{6}\frac{\partial^3 T_i^n}{\partial t^3} + \cdots$$

$$= -\alpha K \frac{\Delta t}{\Delta x^2}\frac{\partial T_i^n}{\partial t} + \frac{\alpha K^2}{2}\frac{\Delta t^2}{\Delta x^2}\frac{\partial^2 T_i^n}{\partial x^2} - \alpha K \frac{\Delta t}{\Delta x}\frac{\partial^2 T_i^n}{\partial x \partial t}$$

$$+ \alpha \frac{\partial^2 T_i^n}{\partial x^2} + \frac{\alpha K^2}{2}\frac{(\Delta t)^2}{\Delta x}\frac{\partial^3 T_i^n}{\partial x \partial t^2} - \frac{1}{2}\alpha K \Delta t \frac{\partial^3 T_i^n}{\partial x^2 \partial t} - \frac{\alpha K^3}{6}\frac{(\Delta t)^3}{(\Delta x)^2}\frac{\partial^3 T_i^n}{\partial t^3}$$

$$+ \frac{1}{12}\alpha (\Delta x)^2 \frac{\partial^4 T_i^n}{\partial x^4} \cdots + \frac{\alpha}{4} K^2 (\Delta t)^2 \frac{\partial^4 T_i^n}{\partial x^2 \partial t^2} + \frac{1}{24}\alpha K^4 \frac{(\Delta t)^4}{(\Delta x)^2}\frac{\partial^4 T_i^n}{\partial t^4}$$

$$\Rightarrow \frac{\partial T_i^n}{\partial t} - \alpha \frac{\partial^2 T_i^n}{\partial x^2} = \Big[-\alpha K \frac{\Delta t}{\Delta x^2}\frac{\partial T_i^n}{\partial t} + \frac{\alpha K^2}{2}\frac{\Delta t^2}{\Delta x^2}\frac{\partial^2 T_i^n}{\partial x^2} - \alpha K \frac{\Delta t}{\Delta x}\frac{\partial^2 T_i^n}{\partial x \partial t}$$

$$\cancel{+\alpha \frac{\partial^2 T_i^n}{\partial x^2}} + \frac{\alpha K^2}{2}\frac{(\Delta t)^2}{\Delta x}\frac{\partial^3 T_i^n}{\partial x \partial t^2} - \frac{1}{2}\alpha K \Delta t \frac{\partial^3 T_i^n}{\partial x^2 \partial t} - \frac{\alpha K^3}{6}\frac{(\Delta t)^3}{(\Delta x)^2}\frac{\partial^3 T_i^n}{\partial t^3}$$

$$+ \frac{1}{12}\alpha (\Delta x)^2 \frac{\partial^4 T_i^n}{\partial x^4} + \cdots \Big] - \frac{\Delta t}{2}\frac{\partial^2 T_i^n}{\partial t^2} - \cdots$$

This is the truncation error.

(b) The leading order term in TE

$$= \frac{1}{12}\alpha (\Delta x)^2 \frac{\partial^4 T_i^n}{\partial x^4} - \frac{\Delta t}{2}\frac{\partial^2 T_i^n}{\partial t^2}$$

$\therefore$ order of accuracy $= O(\Delta x^2, \Delta t)$

(c) This equation is not consistent finite difference approximation.

As we expect finite difference eq^n converge to PDE i.e $\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$ as $\Delta x, \Delta t \to 0$

Independently in the TE expression most of the term $\to 0$ as $\Delta x, \Delta t \to 0$

but $-2\alpha k \frac{\Delta t}{\Delta x^2}$, this term will converge to zero when $\Delta t \to 0$ faster than $\Delta x \to 0$.

Thus there is dependency between Δx & Δt for FDE converging to PDE.

Hence approximation is <u>inconsistent</u>.