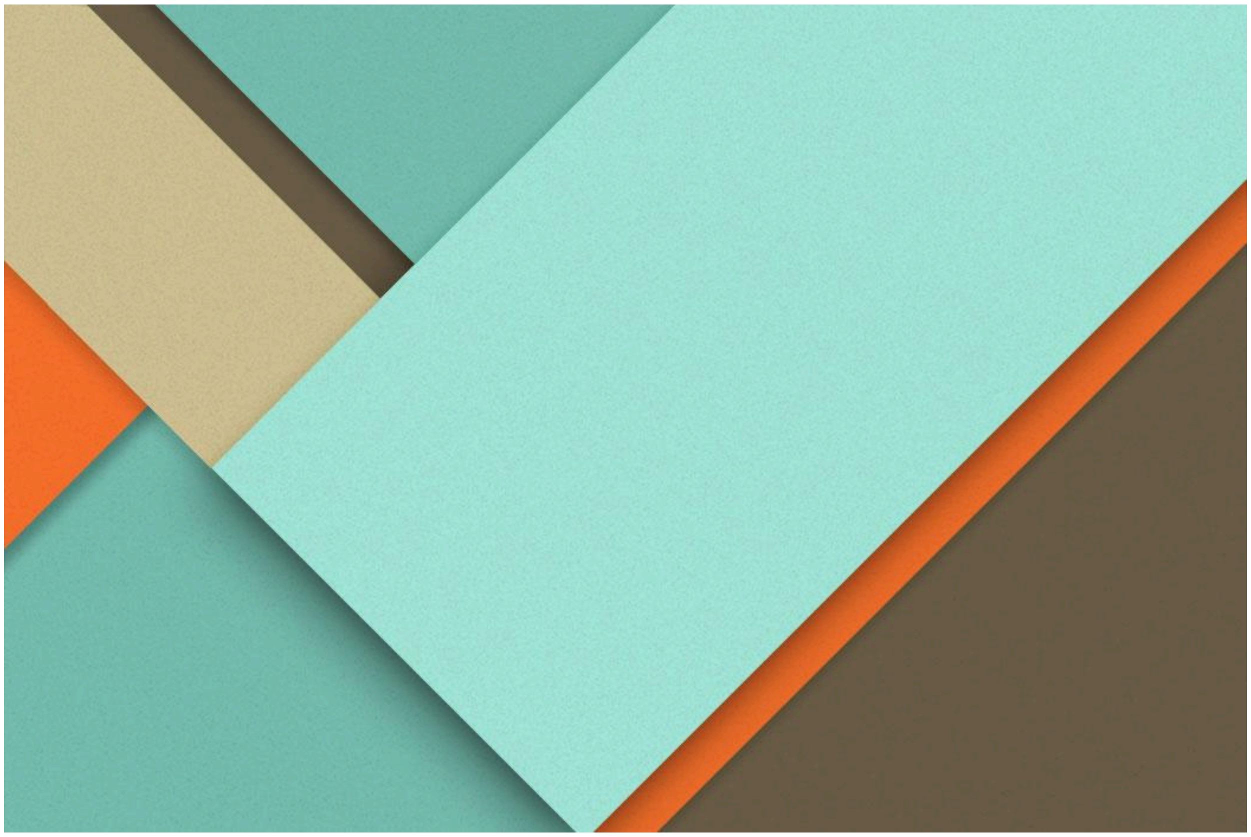


CFD LAB-1

Solving 2D heat equation using CDS in space and Crank Nicolson method in time.

08-11-2024

Hemanth Siva Kumar Ravanam - 23423928

Mohan Kumar Meesala - 23423951

Yuktha Priya Ummareddy - 23423932

Overview

In this report we use Central Difference Scheme in space and Crank-Nicolson method in time with grid(mesh) refinement, to numerically solve the non-dimensional partial differential equation (PDE) that governs 2D heat transfer equation and see under which condition Crank-Nicolson method is stable. MATLab program is used to generate plots for stable time steps for time and temperature(graph) using both Crank-Nicolson and Explicit Euler schemes.

Goals

1. Discretize 2D heat equation by CDS scheme in space and Crank-Nicolson method in time.
2. Write MATLab program for Crank-Nicolson and explicit Euler scheme to solve heat equation.

Context:

Consider the dimensionless 2D heat equation: $\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2},$

The initial and boundary conditions: $\begin{cases} T(x,y,0) = 0, \\ T(0,y,t) = 1 - y^3, \\ T(1,y,t) = 1 - \sin\left(\frac{\pi y}{2}\right), \\ T(x,0,t) = 1, \\ T(x,1,t) = 0. \end{cases}$

The initial boundary condition is not consistent with boundary conditions except for $T(x,1,t) = 0.$

In this Task, we will numerically solve the dimensionless 2D heat equation in both Euler Explicit and Crank-Nicolson methods using the Central Difference Scheme. Through analysis after the result converges by applied boundary conditions we will compare the performance between these schemes and aim to study the behavior of the temperature distribution over a 2D surface or domain.

Approach:

Central Difference Scheme (CDS) :

CDS is derived using forward and backward difference schemes. These two schemes are derived using Taylor series.Taylor series is a mathematical function or expansion of an infinite summation of its derivatives at specific point, used to approximate the value of the function.

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \frac{f'''(x)}{3!}h^3 + \cdots + \frac{f^{(n)}(x)}{n!}h^n$$

From this Taylor series expansion we can derive:

Forward Difference Scheme: $f'(x) \approx \dfrac{f(x+h) - f(x)}{h}$

Backward Difference Scheme: $f'(x) \approx \dfrac{f(x) - f(x-h)}{h}$

Central Difference Scheme: $f'(x) \approx \dfrac{f(x+h) - f(x-h)}{2h}$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Implicit and Explicit schemes :

Implicit and Explicit schemes are used to solve partial differential equations (PDE's) numerically which are in the context of time dependent functions.

- **Implicit Scheme:** The next values or the coming forth values are determined by an equation using current time step and future time step. The resulting equation is as follows,

  For 1D equations : $u_l^{t+1} = u_l^t + \lambda\left(u_{l+1}^{t+1} - 2u_l^{t+1} + u_{l-1}^{t+1}\right), \quad \lambda = \dfrac{K\,\Delta t}{\Delta x^2}$

- **Explicit Scheme:** The next values or the coming forth values are determined by an equation using current time step or the known values. This scheme is simple but there are issues with stability.The resulting equation is as follows

  For 1D equations : $u_l^{t+1} = u_l^t + \lambda\left(u_{l+1}^{t} - 2u_l^{t} + u_{l-1}^{t}\right), \quad \lambda = \dfrac{k\,\Delta t}{\Delta x^2}$

Crank-Nicolson Method :

Crank-Nicolson method is a numerical method to solve time dependent PDEs like heat or diffusion equations, which is a finite difference method. It combines both Explicit and Implicit schemes making it a more stable and efficient way of solving your partial differential equations.It is unconditionally stable for linear model equations because in general it has no restriction on the time step size - (L3, pg:39) but having smaller step size yields better results, also uniform grid, smooth initial and boundary conditions also impacts on stability.

For 1D equations : $\dfrac{u_i^{n+1} - u_i^n}{\Delta t} = \dfrac{\alpha}{2(\Delta x)^2}\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1} + u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)$

Truncation Error in time :

$$u\left((t+\frac{\Delta t}{2}) - \frac{\Delta t}{2}\right) = u\left(t+\frac{\Delta t}{2}\right) - \frac{\Delta t}{2}u'\left(t+\frac{\Delta t}{2}\right) + \frac{\left(\frac{\Delta t}{2}\right)^2}{2!}u''\left(t+\frac{\Delta t}{2}\right) - \frac{\left(\frac{\Delta t}{2}\right)^3}{3!}u'''\left(t+\frac{\Delta t}{2}\right) \text{...Eqn(1)}$$

$$u\left((t+\frac{\Delta t}{2}) + \frac{\Delta t}{2}\right) = u\left(t+\frac{\Delta t}{2}\right) + \frac{\Delta t}{2}u'\left(t+\frac{\Delta t}{2}\right) + \frac{\left(\frac{\Delta t}{2}\right)^2}{2!}u''\left(t+\frac{\Delta t}{2}\right) + \frac{\left(\frac{\Delta t}{2}\right)^3}{3!}u'''\left(t+\frac{\Delta t}{2}\right) \text{...Eqn(2)}$$

Subtracting Equation 1 from 2

$$u(t+\Delta t) - u(t) = \Delta t\, u'\left(t+\frac{\Delta t}{2}\right) + \frac{\left(\frac{\Delta t}{2}\right)^3}{3!}u'''\left(t+\frac{\Delta t}{2}\right)$$

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} = u'\left(t+\frac{\Delta t}{2}\right) + \frac{\Delta t^2}{48}u'''\left(t+\frac{\Delta t}{2}\right)$$

Where $\dfrac{\Delta t^2}{48}\,u'''\left(t+\dfrac{\Delta t}{2}\right)$ is Truncation error. $\mathcal{T} = \mathcal{O}(\Delta t^2)$

Truncation error in space :

$$T_{i+1,j}^t = T_{i,j}^t + \Delta x\left(\frac{\partial T}{\partial x}\right)_{i,j}^t + \frac{\Delta x^2}{2}\left(\frac{\partial^2 T}{\partial x^2}\right)_{i,j}^t + \frac{\Delta x^3}{6}\left(\frac{\partial^3 T}{\partial x^3}\right)_{i,j}^t + \frac{\Delta x^4}{24}\left(\frac{\partial^4 T}{\partial x^4}\right)_{i,j}^t + \text{HOT}$$

$$T_{i-1,j}^t = T_{i,j}^t - \Delta x\left(\frac{\partial T}{\partial x}\right)_{i,j}^t + \frac{\Delta x^2}{2}\left(\frac{\partial^2 T}{\partial x^2}\right)_{i,j}^t - \frac{\Delta x^3}{6}\left(\frac{\partial^3 T}{\partial x^3}\right)_{i,j}^t + \frac{\Delta x^4}{24}\left(\frac{\partial^4 T}{\partial x^4}\right)_{i,j}^t + \text{HOT}$$

$$\frac{T_{i+1,j}^t - 2T_{i,j}^t + T_{i-1,j}^t}{\Delta x^2} = \left(\frac{\partial^2 T}{\partial x^2}\right)_{i,j}^t + \frac{\Delta x^2}{12}\left(\frac{\partial^4 T}{\partial x^4}\right)_{i,j}^t + \text{HOT}$$

Where $\dfrac{\Delta x^2}{12}\left(\dfrac{\partial^4 T}{\partial x^4}\right)_{i,j}^t + \text{HOT}$ is Truncation error. $\mathcal{T} = \mathcal{O}(\Delta x^2)$

Similarly,

$$\frac{T_{i+1,j}^{t+1} - 2T_{i,j}^{t+1} + T_{i-1,j}^{t+1}}{\Delta x^2} = \left(\frac{\partial^2 T}{\partial x^2}\right)_{i,j}^{t+1} + \frac{\Delta x^2}{12}\left(\frac{\partial^4 T}{\partial x^4}\right)_{i,j}^{t+1} + \text{HOT} \qquad \mathcal{T} = \mathcal{O}(\Delta x^2)$$

$$\frac{T^t_{i,j+1} - 2T^t_{i,j} + T^t_{i,j-1}}{\Delta y^2} = \left(\frac{\partial^2 T}{\partial y^2}\right)^t_{i,j} + \frac{\Delta y^2}{12}\left(\frac{\partial^4 T}{\partial y^4}\right)^t_{i,j} + \text{HOT} \qquad \mathcal{T} = \mathcal{O}(\Delta y^2)$$

$$\frac{T^{t+1}_{i,j+1} - 2T^{t+1}_{i,j} + T^{t+1}_{i,j-1}}{\Delta y^2} = \left(\frac{\partial^2 T}{\partial y^2}\right)^{t+1}_{i,j} + \frac{\Delta y^2}{12}\left(\frac{\partial^4 T}{\partial y^4}\right)^{t+1}_{i,j} + \text{HOT} \qquad \mathcal{T} = \mathcal{O}(\Delta y^2)$$

Total Truncation error = $\mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta y^2)$

Thus we can say the Crank-Nicolson scheme is second order accurate in both space and time.

Numerical Discretization:

Given 2D heat equation : $\dfrac{\partial T}{\partial t} = \dfrac{\partial^2 T}{\partial x^2} + \dfrac{\partial^2 T}{\partial y^2},$

Now, the right hand side term of the equation should be discretized using the CDS scheme in space and Crank-Nicolson method in time.

CDS Scheme: $\dfrac{\partial^2 u}{\partial x^2} \approx \dfrac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{(\Delta x)^2}$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u(y+\Delta y) - 2u(y) + u(y-\Delta y)}{(\Delta y)^2}$$

$$\frac{\partial u}{\partial t} \approx \frac{u^{n+1} - u^n}{\Delta t}$$

Numerical Discretization for the Crank-Nicolson Scheme 2D Heat Equation :

In our scheme we will solve the problem by averaging both implicit and explicit methods for partial derivatives in both X and Y directions, by discretizing the 2D heat equation with nodal positions in terms of i and j.

$$u^{t+1}_{i,j} - \tfrac{\lambda}{2}\big(u^{t+1}_{i+1,j} + u^{t+1}_{i-1,j} + u^{t+1}_{i,j+1} + u^{t+1}_{i,j-1} - 4u^{t+1}_{i,j}\big)$$

$$= u^t_{i,j} + \tfrac{\lambda}{2}\big(u^t_{i+1,j} + u^t_{i-1,j} + u^t_{i,j+1} + u^t_{i,j-1} - 4u^t_{i,j}\big)$$

$$\lambda = k\frac{\Delta t}{\Delta x^2}, \quad \Delta x = \Delta y$$

For every value of i & j in the 2D grid, we will get a sequence of equations, so these equations can be solved using the form A.x=B, where x is our solution, A is the matrix and B is a vector. While solving the equation we consider the LHS of the above equation as u_new, that is our updated result or new temperature, which will be calculated using the RHS part of the equation where the equation is formulated by the known temperature values at the grid point locations. The Crank-Nicolson method is more stable and efficient than both explicit and implicit methods because it updates values of both x and y direction i.e., grid points i and j in a single iteration of the computed results.

MATLAB Programme and Implementation:

Here we have solved or written a code to solve 2D unsteady heat equation using the Central Differencing Scheme in both Euler Explicit and Crank-Nicolson methods using MATLAB.

To write the code in MATLAB for both Euler Explicit and Crank-Nicolson methods we have changed the functions to algebraic form by discretizing the known functions which are machine understandable.

Here we have taken the size of the domain as 1 in both X and Y directions of cartesian coordinates i.e., Lx = Ly = 1. The distance between two adjacent nodes is taken as h = 1/40.

$$\Delta x = \Delta y = \frac{1}{40}$$

We have tried to evaluate both the Euler Explicit and Crank-Nicolson methods at different time steps Δt at 0.0001, 0.001 and 0.01 with the same provided or given boundary conditions. Here we evaluated both the schemes for total run time T = 0.16

λ is the stability parameter where it depends on both Δt and h

Here we initially constructed an "u" matrix with zeros in all the node points which are later given boundary conditions to get updated and get values at the outer rows and columns according to their boundary conditions.

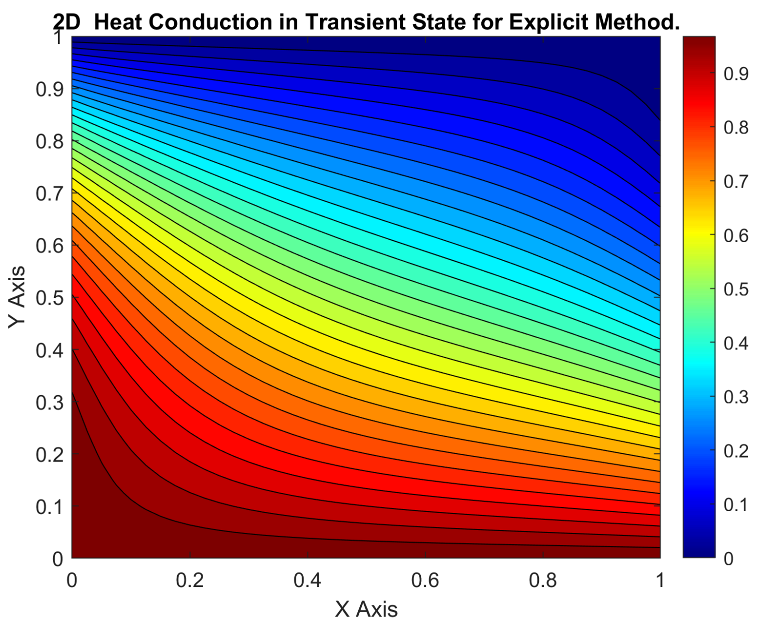
Here we run the scheme for inner nodes excluding boundary nodes as the boundary nodal values are fixed due to boundary conditions. Here we have a 41:41 initial u matrix so we have to calculate or update the temperatures at internal nodal points which will be a 39:39 matrix which excludes first and last, rows and columns as those values are fixed.

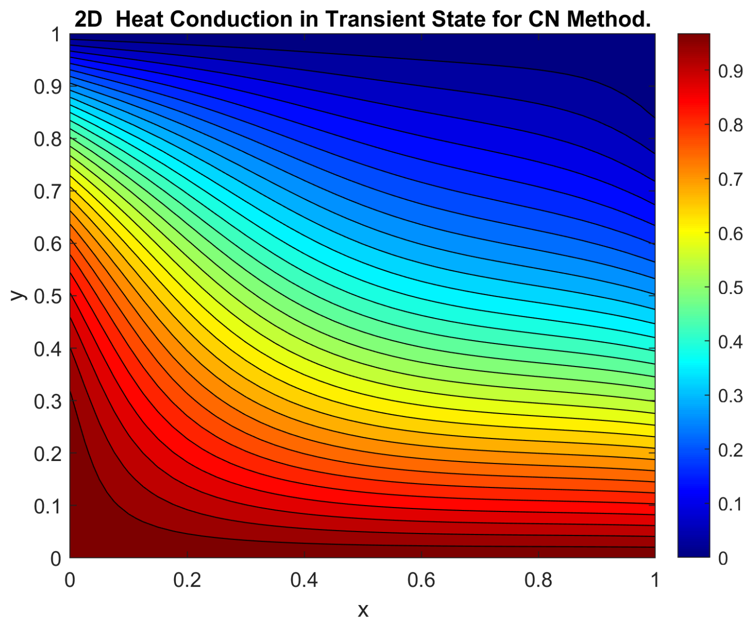
In both the schemes we use the "for" loop to run the iterations and the updated values are stored in u_new which will be again assigned to u values to run the iterations with updated temperature values.

Results and Discussion:

For the Euler Explicit scheme at x=y=0.4 the temperature value is 0.6639 when run dt = 0.0001 and temperature value of Crank-Nicolson method at same grid point is 0.5995

The below images are the both Explicit and Crank-Nicolson contour's , plotted at dt=0.0001

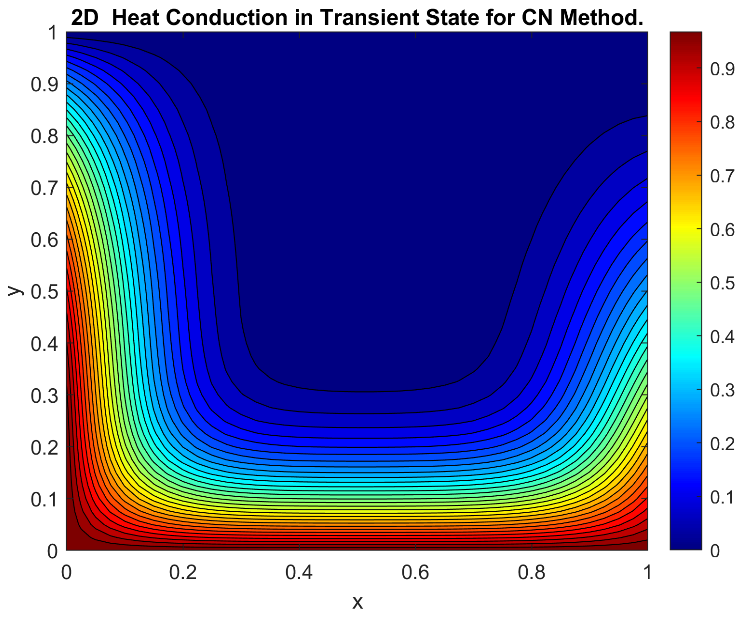


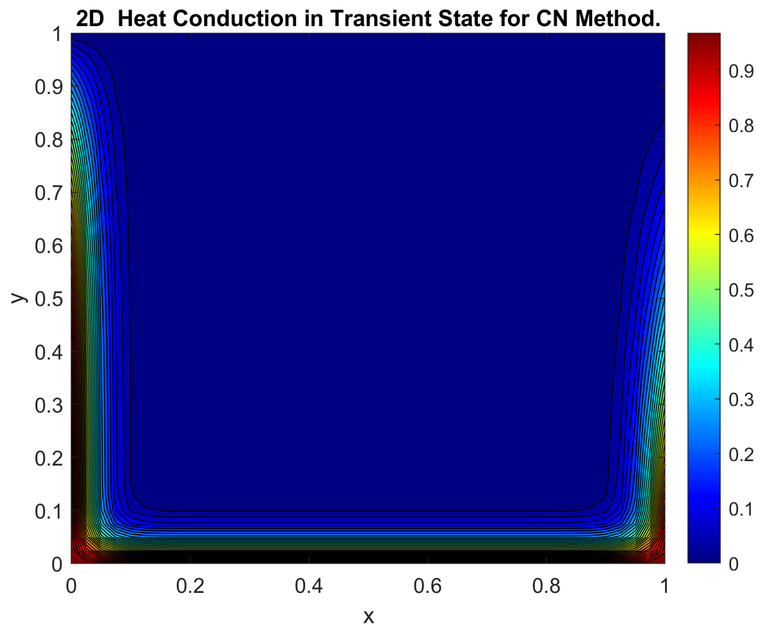


Here both the plots look very similar, but when the contour is getting updated for Explicit, the change is very rapid or fast when compared to Crank-Nicolson because we know Crank-Nicolson is an unconditionally stable scheme where the Explicit scheme has stability issues.

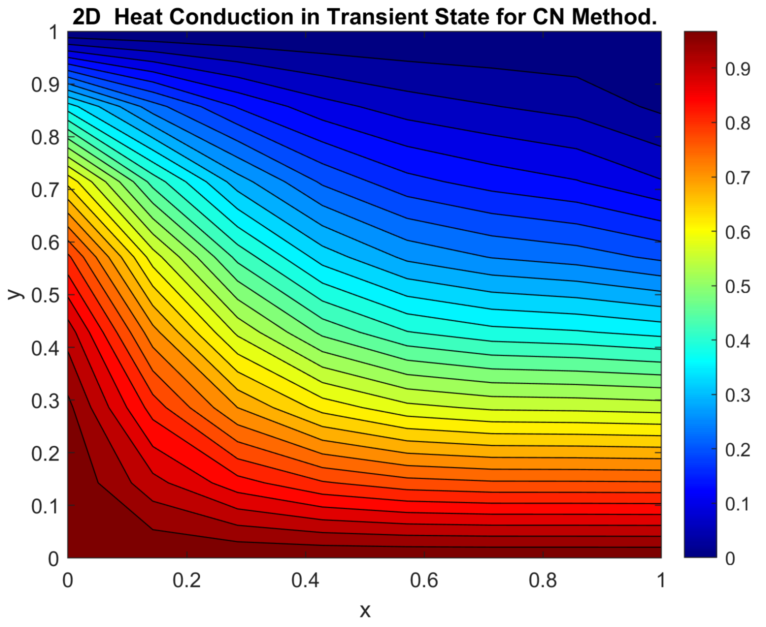
When running the code for different time steps at dt = 0.001 and 0.01 the results are diverging for both the schemes, but as we know CN in more stable for the lambda (stability parameter) is less than 0.1

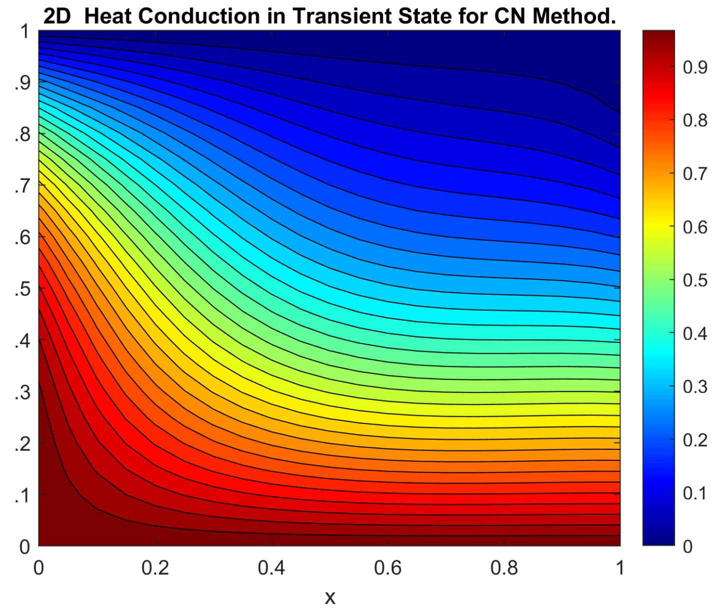
To make CN method to work and to converge the results we have to forcefully maintain our lambda value at 0.1 but the "h" and "dt" values are fixed to h=1/40, so to converge my results for different time steps im changing my grid points to maintain stability, as my time is increasing from 0.0001 to 0.001 we are decreasing number of grid points while maintaining our domain size at Lx=Ly=1. When we run the code as mentioned above decreasing my grid points when increasing dt, our results are being converged.





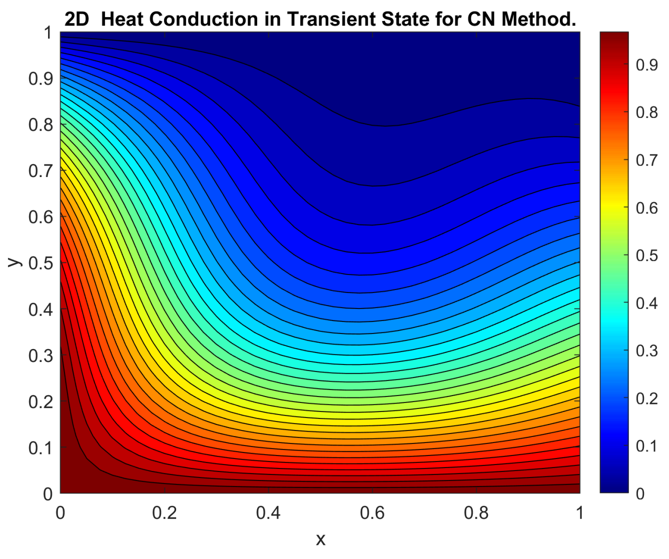
The above attached images are plotted using Crank-Nicolson at different time steps 0.001 and 0.01 while forcefully fixing my lambda at 0.1 which is a conditionally stable parameter.

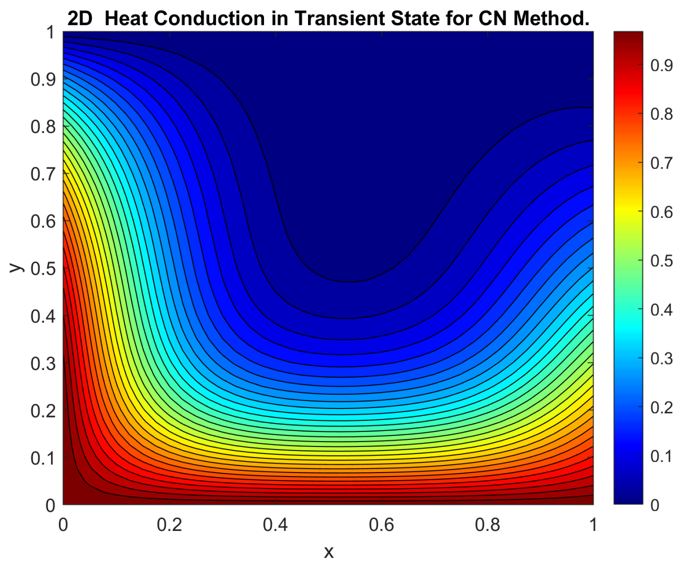


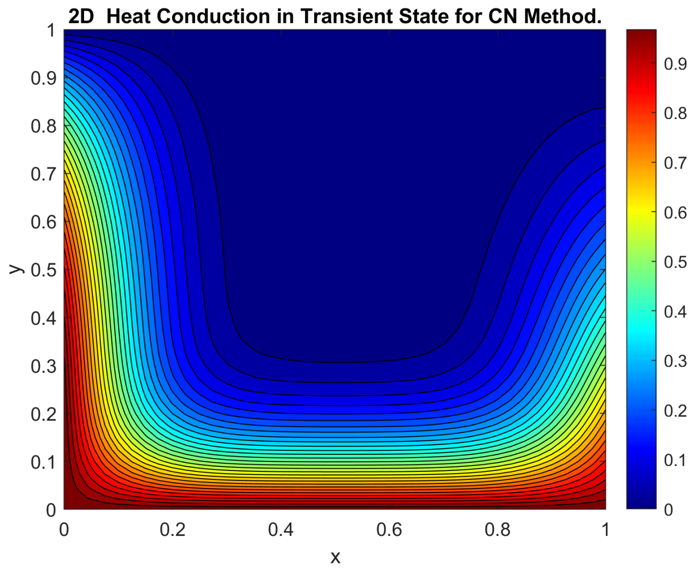


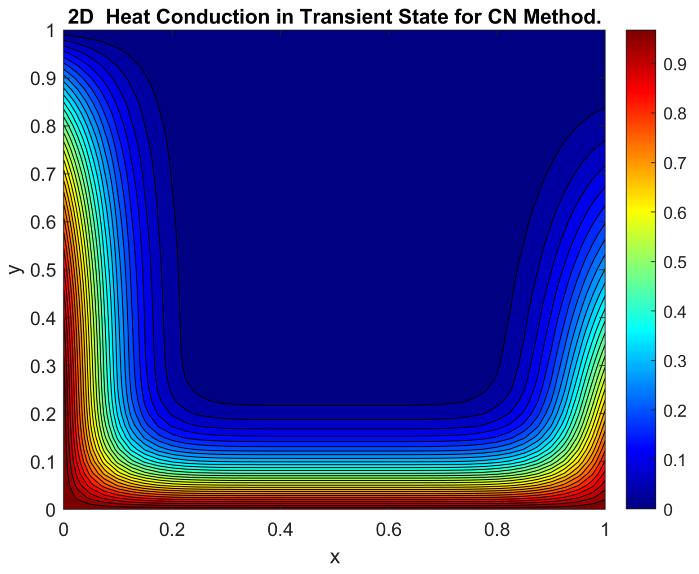
While changing my grid points as increase in dt to 0.001 and 0.01 the contour looks like the above images.

The drawback of doing this method will cost us the accuracy of the results ,as we are decreasing our grid points.This gives less accurate values.









The graphs correspond to contour plots for different total time's T = 0.08 , T= 0.04 , T= 0.02 and T= 0.01

We can observe that as the total time decreases, the number of iterations are being decreased accordingly and the code is being terminated very early which can be seen in the above extracted contour's.

We choose the value of total time T because we are interested in evaluating our temperature value after a particular extent of time passed. If T increases the heat will be more diffused as the runtime increases.This is also the reason we choose T that are factors of 2.

Conclusion:

Hereby we conclude that the Crank-Nicolson method is more stable than Euler Explicit.

Stability parameter plays a major role for the results to be converged properly and to achieve accuracy of the solution.

Refinement of mesh is important for the results to converge.

We always calculate the values of internal nodes excluding external because our outer node values will be fixed accordingly to our boundary values. For this we write a loop in the code mentioning that i runs from 2 to Nx-1 and j runs from 2 to Ny-1, constructing and updating only internal node values.

Updating the new values calculated and assigning it to older values is important because if we don't do so, our values won't be updating and there will be no diffusion contour plot will be updated,even though iterations run, code will again calculate the first iteration step or first diffusion step for many iterations, leaving us with a blank and empty contour.

References:

References were taken from NPTEL courses Explicit and Implicit methods to solve heat diffusion.

https://www.youtube.com/watch?v=gje7QDlmGyU

J.D. Anderson's Computational Fluid Dynamics.