Course Project Presentation

For CFD (ME-580)

Presentation Title:
Step 3 of CFD Python: 12 Steps to Navier-Stokes

Course Instructor : Dr. Navaneeth K.M.
Group No. Group 2

Name of Student	**Entry No.**
Meet Ghorecha	2024MEM1008
Manish Choudhary	2024MEM1015
Vivekanand Pandey	2024MEM1031
Jatin Sharma	2024MEM1042
Aditya Pandey	2024MEM1002

- # 1D Diffusion Equation

$$\frac{\partial u(x,t)}{\partial t} = v\frac{\partial^2 u(x,t)}{\partial x^2}$$

Where:

- *u(x, t)* is the concentration (or temperature, etc.)
- *v* is the diffusion coefficient
- *x* is the spatial coordinate
- *t* is time

How to solve This Equation?

we employ **numerical methods** by discretizing both the spatial and temporal domains. This approach transforms the partial differential equation into a set of algebraic equations that can be solved iteratively.

Spatial and Temporal Discretization. We discretize the spatial domain using a finite difference grid.

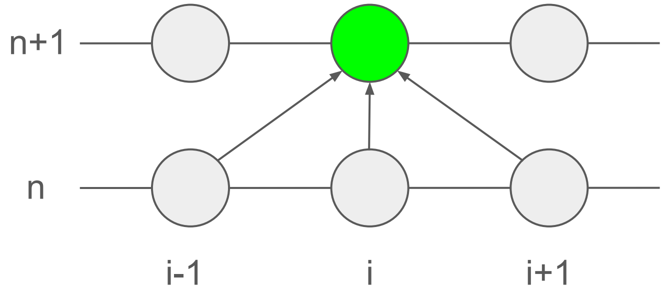
The choice of discretization scheme significantly affects the accuracy, stability, and computational efficiency of the solution.

- # Solve using Explicit Method

Scheme used : Here we are going discretize Forward in Time and Central in Space finite difference

Requirement to solve this Equation : Initial Condition for all spatial nodes and two Boundary Conditions

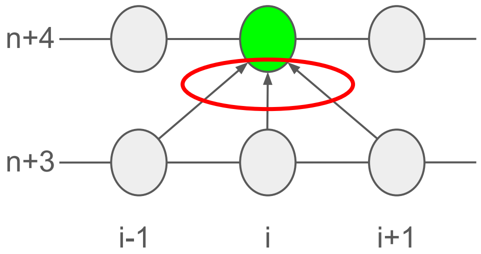
$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = v \cdot \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}$$



- Here from discretized equation we can see that value of u at n+1 is only unknown in equation and it depends on values at n.
- So here we are expressing one unknown in terms of the known values so that's why this method is called **Explicit method**.
- Stencil for this scheme is shown (stencil - Graphical Representation of scheme)

Drawback of using explicit method

We are using explicit scheme so if we are calculating u at n+4 then the information at the boundary at n+4 does not feed into computation so boundary information is one step behind the calculation step.



$$\frac{\partial u}{\partial t} = A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial u}{\partial x} + Cu$$

$$\frac{\partial u(x,t)}{\partial t} = v\frac{\partial^2 u(x,t)}{\partial x^2}$$

$$A = v, \quad B = 0, \quad C = 0.$$

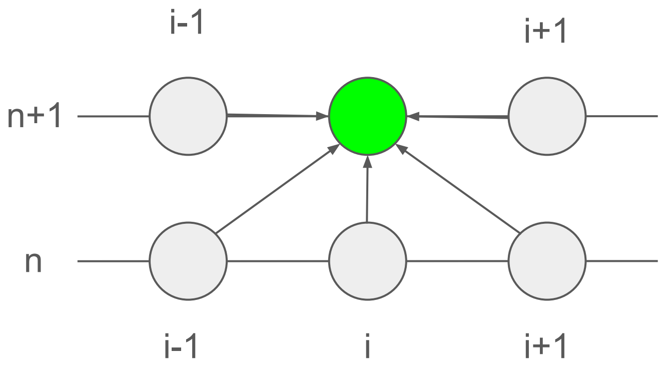
- This equation is parabolic equation for parabolic equation characteristic lines are lines of constant t. So all information at given time level should affect solution but it is not happening in numerical scheme, it only receive information at boundary from previous time step.
- It won't affect if boundary conditions are constant but it affect when boundary conditions are functions of time then boundary condition lag behind from the scheme by one time step.
- So to solve this equation we should use scheme which includes Boundary information at every time step.

● Solve using Implicit Method

We are using implicit scheme to include boundary information at time step in computation.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = v\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}$$

$$-\frac{D\Delta t}{(\Delta x)^2}u_{i-1}^{n+1} + \left(1 + 2\frac{D\Delta t}{(\Delta x)^2}\right)u_i^{n+1} - \frac{D\Delta t}{(\Delta x)^2}u_{i+1}^{n+1} = u_i^n$$



- Use Backward Difference approximation for time derivative
- In this we have 3 unknowns in equation. So we are writing one unknown in terms so other unknown so that's why it is called **Implicit Method.**
- To solve this we need set of equation found by writing this equation for all nodes.
- Then it will become Linear system of equation ,where we get Tridiagonal coefficient matrix.

Crank-Nicolson Method

It is obtained by taking average of Explicit and Implicit Scheme.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{v}{2}\left(\underbrace{\boxed{\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}}}_{\text{Implicit scheme}} + \underbrace{\boxed{\frac{u_{i+1}^{n} - 2u_i^{n} + u_{i-1}^{n}}{(\Delta x)^2}}}_{\text{Explicit scheme}}\right)$$

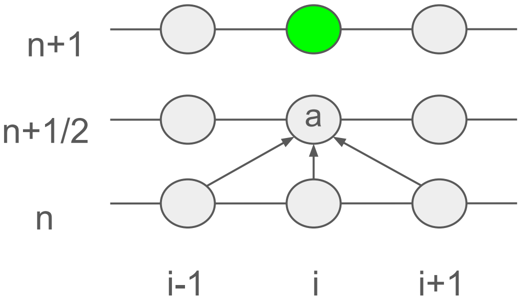
$$-\frac{\alpha}{2}u_{i-1}^{n+1} + (1+\alpha)u_i^{n+1} - \frac{\alpha}{2}u_{i+1}^{n+1} = \frac{\alpha}{2}u_{i-1}^{n} + (1-\alpha)u_i^{n} + \frac{\alpha}{2}u_{i+1}^{n}$$

$$\textbf{where } \alpha = \frac{v\,\Delta t}{(\Delta x)^2}$$

- Here average of explicit and implicit scheme is taken.
- In this we have 3 unknowns in equation. So we are writing one unknown in terms.
- To solve this we need set of equation found by writing this equation for all nodes.
- Then it will become Linear system of equation ,where we get Tridiagonal coefficient matrix.
- It is 2nd order accurate in time and space scheme.

Crank-Nicolson Method

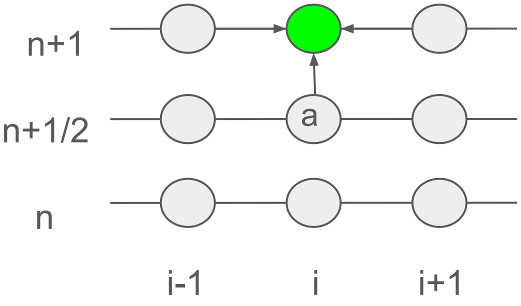
Explicit Scheme



First Using Explicit Scheme calculate half point “a”

$$\frac{u_i^{n+(1/2)} - u_i^n}{\frac{\Delta t}{2}} = \upsilon \left(\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \right)$$

Implicit Scheme

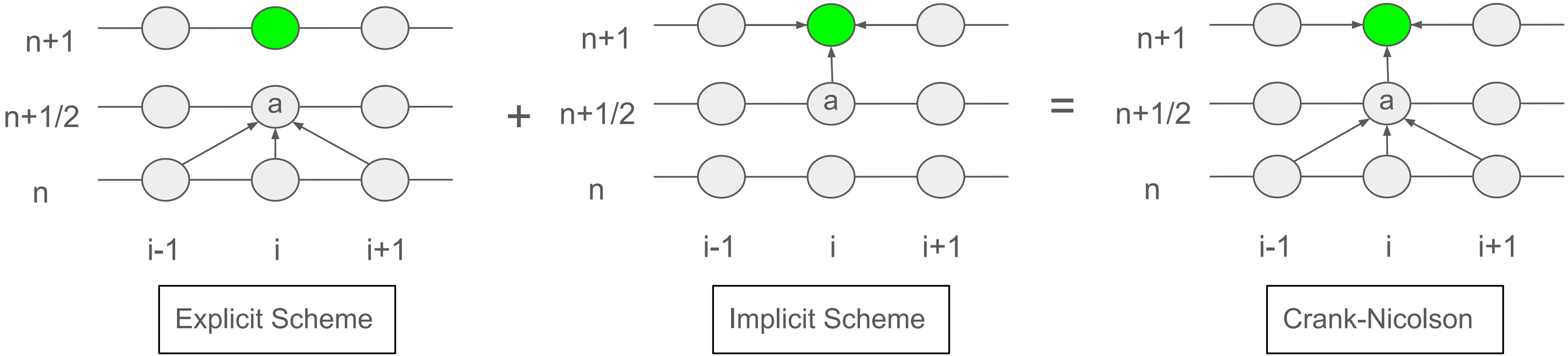


Now Using Implicit Scheme calculate u at n+1 from half point “a”

$$\frac{u_i^{n+1} - u_i^{n+(1/2)}}{\frac{\Delta t}{2}} = \nu \left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} \right)$$

● Crank-Nicolson Method

Now adding both scheme Explicit + implicit so we get Crank-Nicolson Method



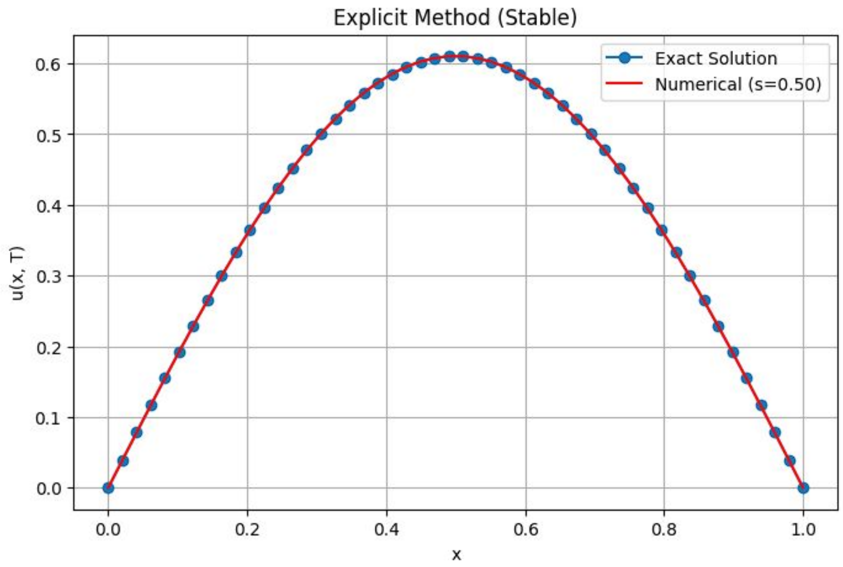
$$\frac{u_i^{n+(1/2)} - u_i^n}{\frac{\Delta t}{2}} = v\left(\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}\right) + \frac{u_i^{n+1} - u_i^{n+(1/2)}}{\frac{\Delta t}{2}} = \nu\left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}\right) = \frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{v}{2}\left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} + \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}\right)$$

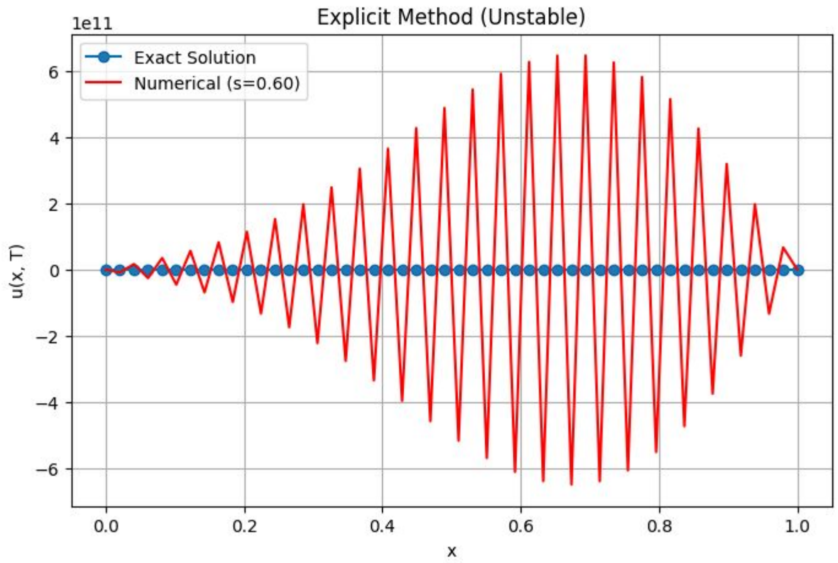
- Solving 1-D diffusion equation using different schemes and comparing with each other

Value of V	0.1
Initial Condition	$u(x,0) = \sin\left(\frac{\pi x}{L}\right), \quad x \in [0, L]$
Boundary Condition	$u(0,t) = 0 \quad \text{and} \quad u(L,t) = 0, \quad \forall t \geq 0$

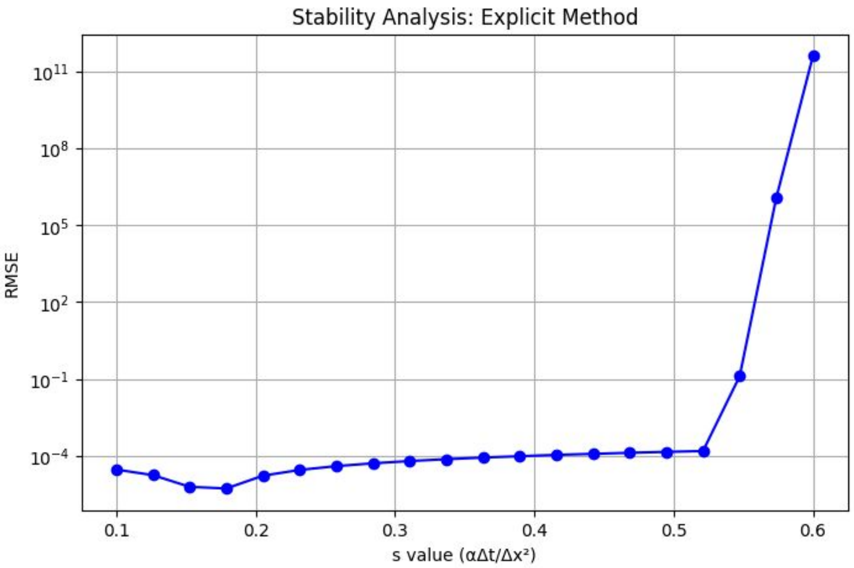
First Solving using Explicit Method

where, $S = \frac{v\,\Delta t}{(\Delta x)^2}$





- Solving 1-D diffusion equation using different schemes and comparing with each other

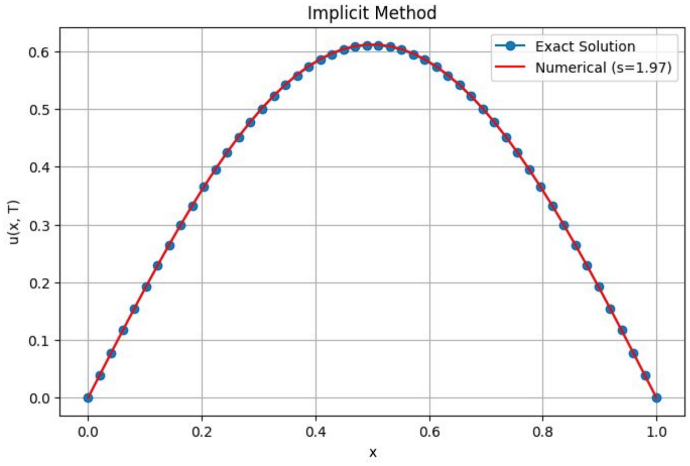


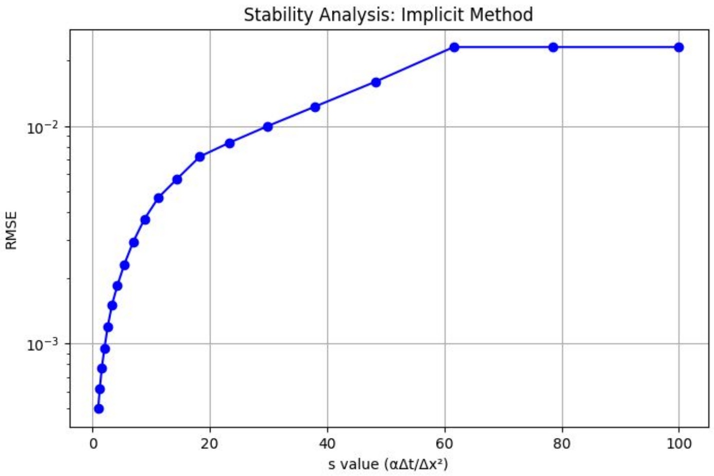
- As seen on previous slide for s=0.5 explicit method is stable and matching with exact solution but as we increase s to 0.60 it become unstable and not match with exact solution.
- Stability of this method calculated using RMSE between actual and Numerical solution.
- So as value of s increases RMSE starts increasing rapidly which shows method becomes unstable as value of s increases,

- Solving 1-D diffusion equation using different schemes and comparing with each other

Now Solving using Implicit Method

- Here using implicit method it is observed that it is matching with exact solution for s=1.97.
- And from stability analysis it is observed that till value of s=60 RMSE increases afterwards it become constant it shows stability of method.

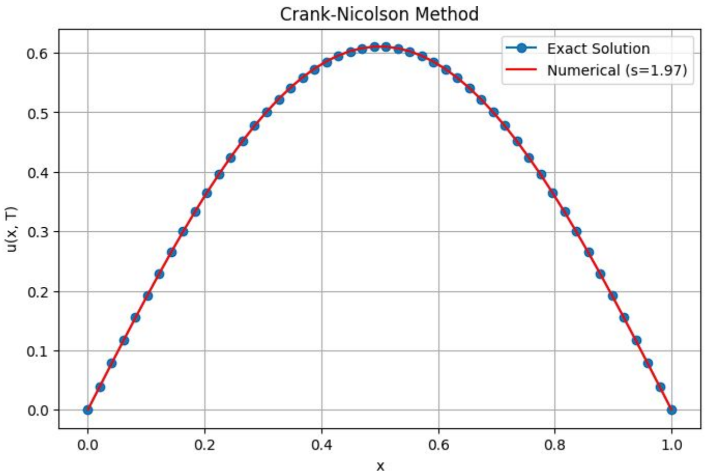


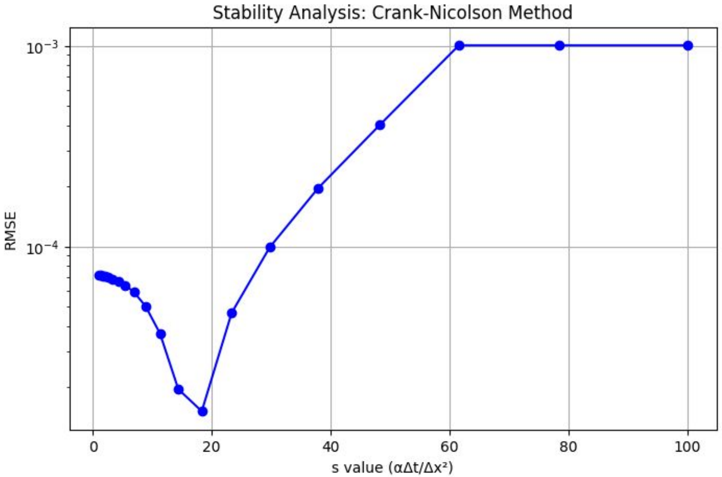


- Solving 1-D diffusion equation using different schemes and comparing with each other

Now Solving using Crank-Nicolson Method

- Here using implicit method it is observed that it is matching with exact solution for s=1.97.
- And from stability analysis it is observed that till value of s=60 RMSE increases afterwards it become constant it shows stability of method.
- So among three most Stable and accurate method is Crank-Nicolson Method.

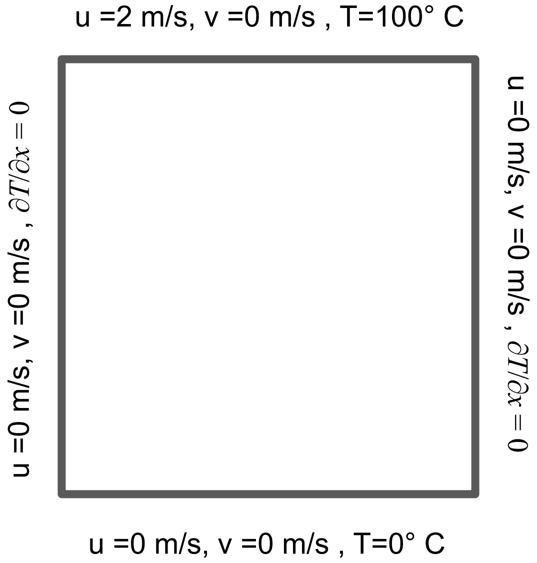




- Heated Lid-Driven Cavity Problem

Problem Setup	
Geometry	2D Square Cavity (1m x 1m)
Boundary Condition	
Top Wall	u = 2 m/s
	T = 100 ° C
Bottom Wall	u = 0 m/s, v = 0 m/s
	T = 0 ° C
Left Wall	u = 0 m/s, v = 0 m/s
	$\partial T/\partial x = 0$
Right Wall	u = 0 m/s, v = 0 m/s
	$\partial T/\partial x = 0$

Initial Conditions: For walls initial conditions are same as boundary conditions and for inferior points at t=0 Temperature and velocities both are zero.



- Heated Lid-Driven Cavity Problem

Governing Equations for this problem which we have to solve

Continuity:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

Momentum:

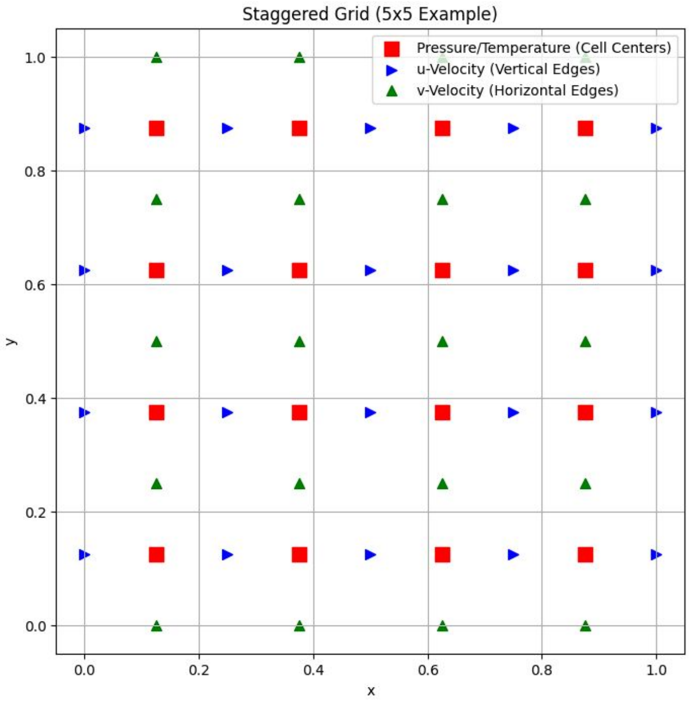
$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \beta g (T - T_{\text{ref}})\mathbf{j} \tag{2}$$

Energy:

$$\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla) T = \alpha \nabla^2 T \tag{3}$$

- Heated Lid-Driven Cavity Problem

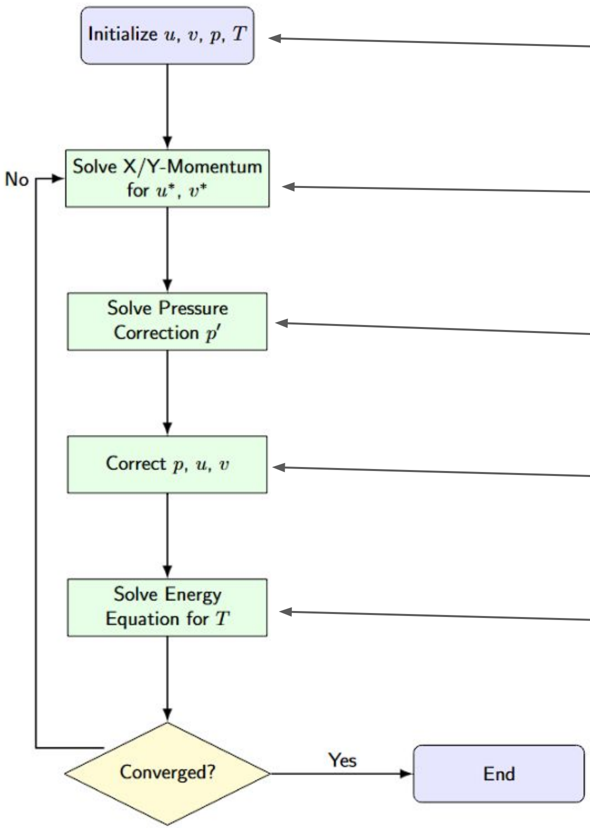
Staggered Grid is used to solve this problem



Term	Equation Part	Involved Nodes	Discretization Scheme
Advection (u)	$\frac{\partial(u^2)}{\partial x}$	$u[i-1,j],\ u[i,j]$	Upwind
Diffusion (u)	$\nu\nabla^2 u$	$u[i-1,j],\ u[i+1,j],\ u[i,j\pm1]$	Central Difference
Pressure Gradient	$\frac{\partial p}{\partial x}$	$p[i-1,j],\ p[i,j]$	Central Difference
Advection (T)	$u\frac{\partial T}{\partial x}$	$T[i-1,j],\ T[i+1,j]$	Central Difference
Diffusion (T)	$\alpha\nabla^2 T$	$T[i\pm1,j],\ T[i,j\pm1]$	Central Difference

- Heated Lid-Driven Cavity Problem

Solution Algorithm Flowchart



Start with initial Guesses of u,v,p,T

Solving X and Y Momentum equation using p and T and find velocity corrections

Solving Pressure correction using velocity corrections $\nabla^2 p' = \frac{1}{\Delta t\, \rho}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right)$

Finding corrected Pressure and velocities

Solving Energy Equation using corrected velocities

- Heated Lid-Driven Cavity Problem

$u^{\text{new}} = u^{\text{old}} + \Delta t \left[-\frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + \nu \nabla^2 u - \frac{\partial p}{\partial x} \right]$	Using Initial value finding velocity correction using this equation
$\nabla^2 p' = \frac{1}{\Delta t\, \rho} \left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y} \right)$	Using velocity correction finding pressure correction
$u^{\text{new}} = u^* - \frac{\Delta t}{\rho} \frac{\partial p'}{\partial x}$, $v^{\text{new}} = v^* - \frac{\Delta t}{\rho} \frac{\partial p'}{\partial y}$, $p^{\text{new}} = p^{\text{old}} + p'$	Using velocity correction and pressure correction finding corrected pressure and velocities
$T^{\text{new}} = T^{\text{old}} + \Delta t \left[-u \frac{\partial T}{\partial x} - v \frac{\partial T}{\partial y} + \alpha \nabla^2 T \right]$	Using this corrected velocities and pressure solving energy equation for temperature
$\text{Residual} = \max \left| \phi^{\text{new}} - \phi^{\text{old}} \right| < \text{tolerance}$	Checking Residual for convergence

- Heated Lid-Driven Cavity Problem

Coding of Solution algorithm

```python
#------------------------------
# Solve X-Momentum Equation
#------------------------------
for i in range(1, N-1):
    for j in range(1, N-1):
        # Advection terms (upwind)
        ue = 0.5 * (u[j, i+1] + u[j, i])
        uw = 0.5 * (u[j, i-1] + u[j, i])
        un = 0.5 * (u[j+1, i] + u[j, i])
        us = 0.5 * (u[j-1, i] + u[j, i])

        # Diffusion term
        diffusion = (u[j, i+1] - 2*u[j, i] + u[j, i-1]) / dx**2 \
                  + (u[j+1, i] - 2*u[j, i] + u[j-1, i]) / dy**2

        # Pressure gradient
        dpdx = (p[j, i+1] - p[j, i-1]) / (2*dx)

        # Update u
        u[j, i] = u_old[j, i] + dt * (
            - (ue**2 - uw**2)/dx - (un*v[j, i] - us*v[j, i-1])/dy
            + (diffusion)/Re
            - dpdx
        ) * alpha_u
```

X-momentum equation (similarly for Y)

```python
#------------------------------
# Solve Pressure Poisson Equation (SIMPLE Algorithm)
#------------------------------
for _ in range(20):
    p_old = p.copy()
    for i in range(1, N-1):
        for j in range(1, N-1):
            p[j, i] = ((p_old[j, i+1] + p_old[j, i-1]) * dy**2 +
                      (p_old[j+1, i] + p_old[j-1, i]) * dx**2 -
                      (u[j, i+1] - u[j, i-1]) * dy**2 * dx / dt -
                      (v[j+1, i] - v[j-1, i]) * dx**2 * dy / dt
                    ) / (2*(dx**2 + dy**2))
    # Apply pressure BCs (dp/dn = 0)
    p[:, 0] = p[:, 1]          # Left wall
    p[:, -1] = p[:, -2]        # Right wall
    p[0, :] = p[1, :]          # Bottom wall
    p[-1, :] = p[-2, :]        # Top wall
```

Pressure Correction

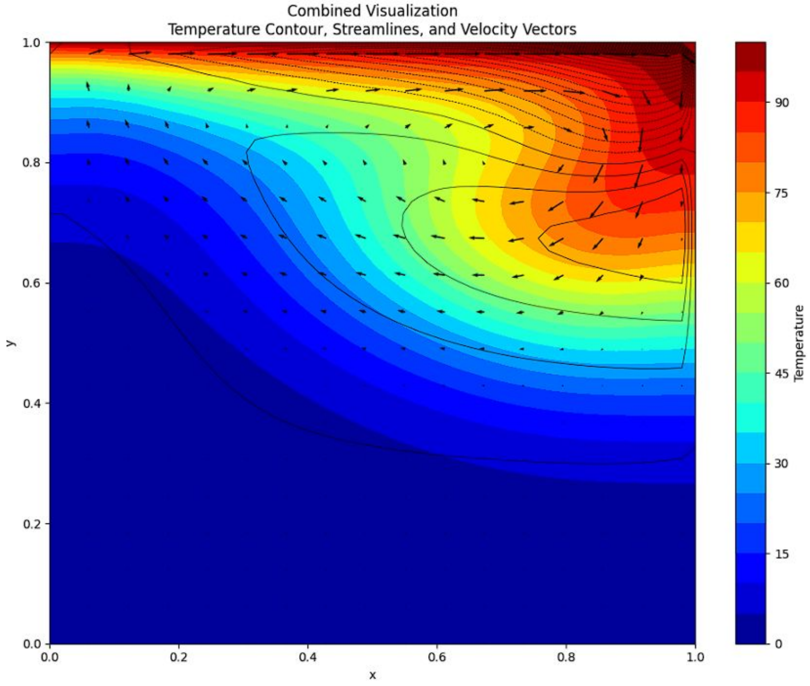
```python
#------------------------------
# Correct Velocities
#------------------------------
u[1:-1, 1:-1] -= alpha_p * (p[1:-1, 2:] - p[1:-1, :-2]) / (2*dx) * dt
v[1:-1, 1:-1] -= alpha_p * (p[2:, 1:-1] - p[:-2, 1:-1]) / (2*dy) * dt

#------------------------------
# Solve Energy Equation
#------------------------------
for i in range(1, N-1):
    for j in range(1, N-1):
        # Advection terms (upwind)
        dTdx = (T[j, i+1] - T[j, i-1]) / (2*dx)
        dTdy = (T[j+1, i] - T[j-1, i]) / (2*dy)
        # Diffusion term
        diffusion = (T[j, i+1] - 2*T[j, i] + T[j, i-1]) / dx**2 \
                  + (T[j+1, i] - 2*T[j, i] + T[j-1, i]) / dy**2
        # Update T
        T[j, i] = T_old[j, i] + dt * (
            -u[j, i] * dTdx - v[j, i] * dTdy
            + alpha * diffusion
        ) * alpha_T
```

Corrected Velocities , Pressure and Energy equation

- Heated Lid-Driven Cavity Problem

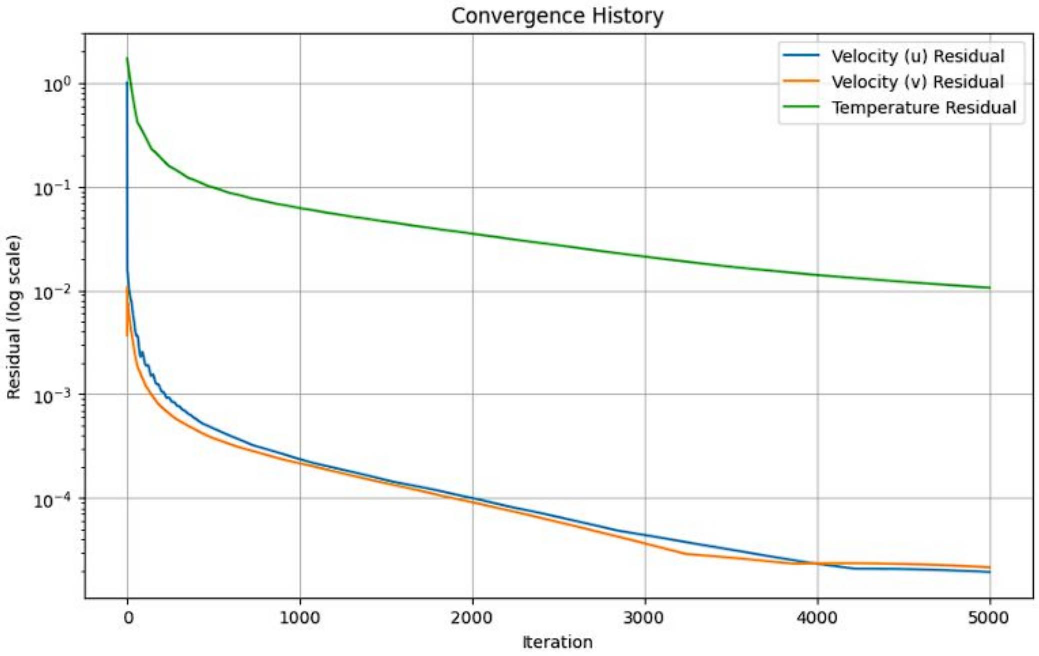
Results



- Here combined Visualization of temperature contour , streamlines and velocity vectors is shown.
- Here contour plot shows change in temperature , and black arrows shows velocity vectors and black line shows streamlines.
- As we can see as time progress temperature and velocity in top right region starts increasing because of higher temperature on top wall and velocity of top wall in x direction.

Heated Lid-Driven Cavity Problem

Convergence and stability



- As we can see from the graph as we performs the iterations Value of residual starts decreasing for velocity u, v and for temperature.
- Which shows solutions is converging towards actual solution and which also shows stability of method that as we go further in time value of residual decrease.
- So from this we can say that our solution is accurate and method we have used is accurate.

Thank You