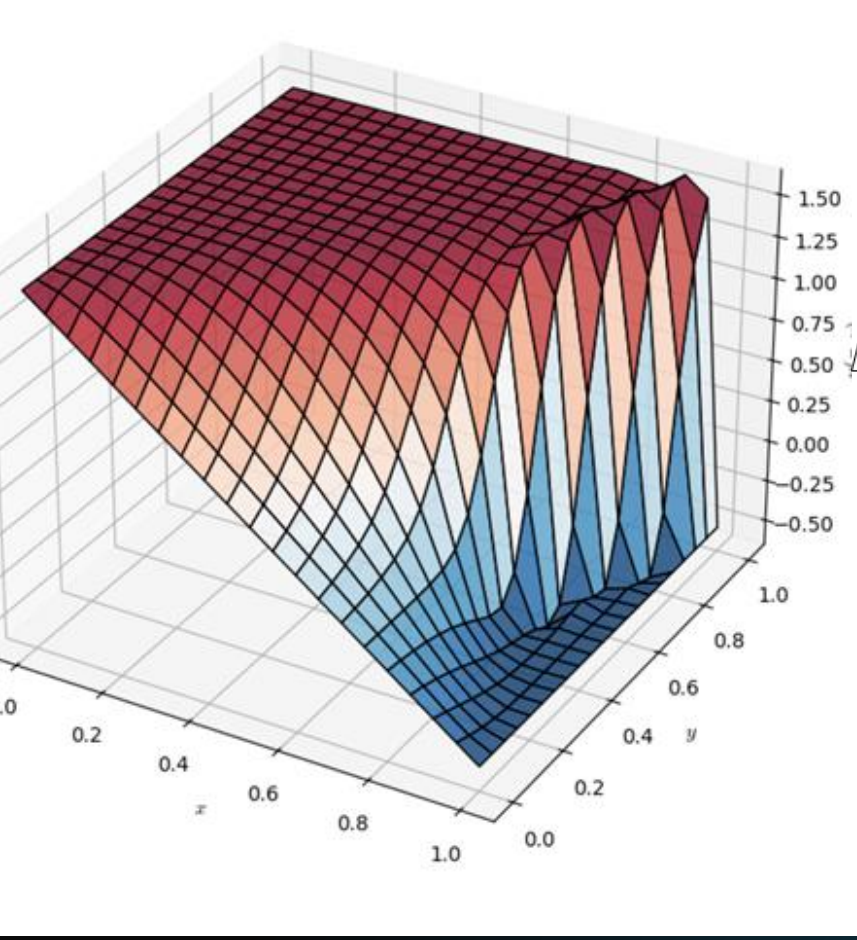
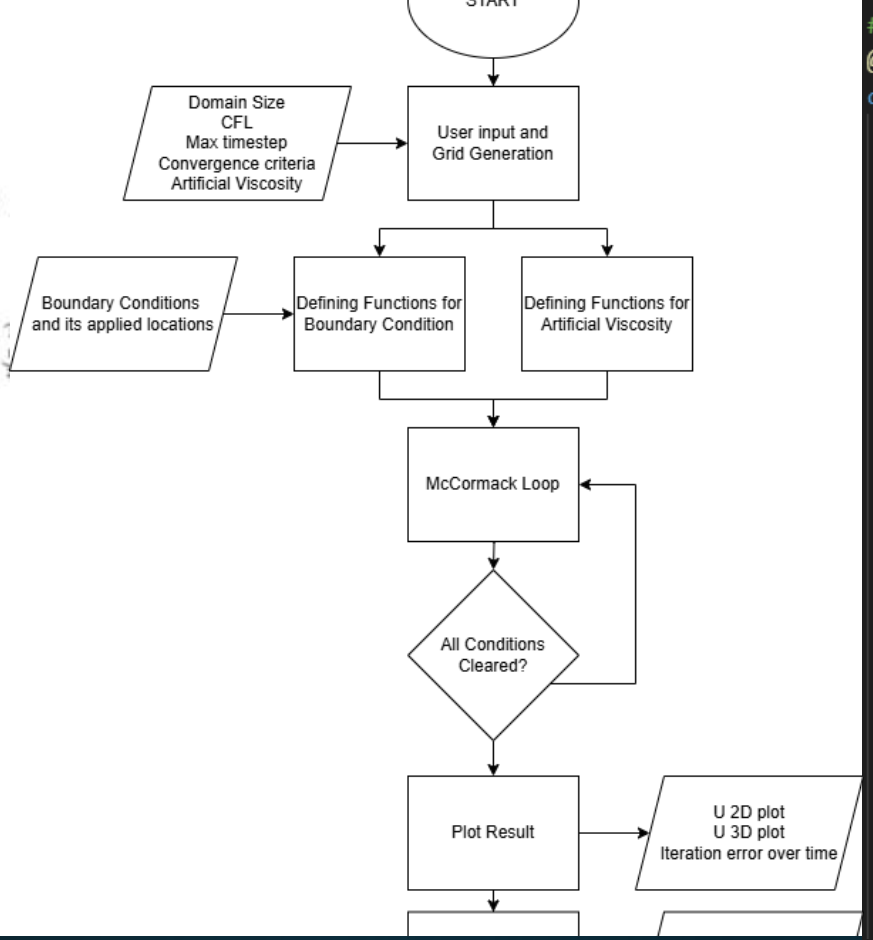





```python
### ------------ FUNCTIONS NEEDED TO DEFINE BOUNDARY AND VIS
@njit(cache = True)
def viscosity(u, nu, stars):
    ''' To calculate artificial viscosity term: S_{i, j} = S
    # Make S_i and S_j array to store each value
    S_i = np.zeros_like(u)
    S_j = np.zeros_like(u)

    # Looping through each index
    for j in range(1, ny):
        for i in range(1, nx-1):
            S_i[j, i] = abs(u[j, i+1] - 2*u[j, i] + u[j, i-1
            S_j[j, i] = abs(u[j  , i] - 2*u[j, i] + u[j-1, i

    # Total viscosity
    S = nu*(S_i + S_j)

    # buat set viscosity di boundary
    for j in range(ny):
        for i in range(nx):
            # At x = 0, left
            S[j, 0] = nu

            # at x = 1, right
            S[j, -1] = nu                    # minus index means

            # at y = 0, bottom
```

Inviscid Burger's Equation in Finite Difference by Using Mc-Cormack Method

Muhammad Lucky Witjaksono - 23623303

Pramudya Vito Agata - 23624020


```python
# Buat nampung error
error_progress = np.zeros(timesteps)

# MacCormack scheme
@njit(cache = True)
def simulate_cormeck (u, timesteps, dx, dy, error_progress, final_t):
    for t in range(timesteps):

        # Update max_u untuk update dt
        max_u = np.max(np.abs(u))
        if max_u > 100:
            print('The solutions Diverged, Please Check the appropriate physiscs settings')
            break
        dt = CFL * min(dx, dy) / (max_u)

        # Calculate the conservative variable
        E = u**2 / 2
        F = u.copy()

        ''' Predictor step '''
        u_star = u.copy()
        for j in range(1, ny):
            for i in range(1, nx - 1):
                u_star[j, i] = u[j, i] - dt/dx * (E[j, i+1] - E[j, i]) - dt/dy * (F[j, i] - F[j-1, i])

        # # Artificial Viscosity
```

Page of content

Problem Definition

Inviscid Burger's Equation In Finite Difference Method

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0$$

Fundamental PDE's equation, with application of:

- Discontinuity
- Non-Linear Phenomena
- Shockwave Formation

Particularly, in this problem we are tasked to:

- Generate Grid Based on TFI
- Create FDM with Mc-Cormack Scheme
- Analyze the effect of artificial viscosity

$$u(0, y) = 1.5$$

$$u(1, y) = -0.5$$

$$u(x, 0) = 1.5 - 2x$$

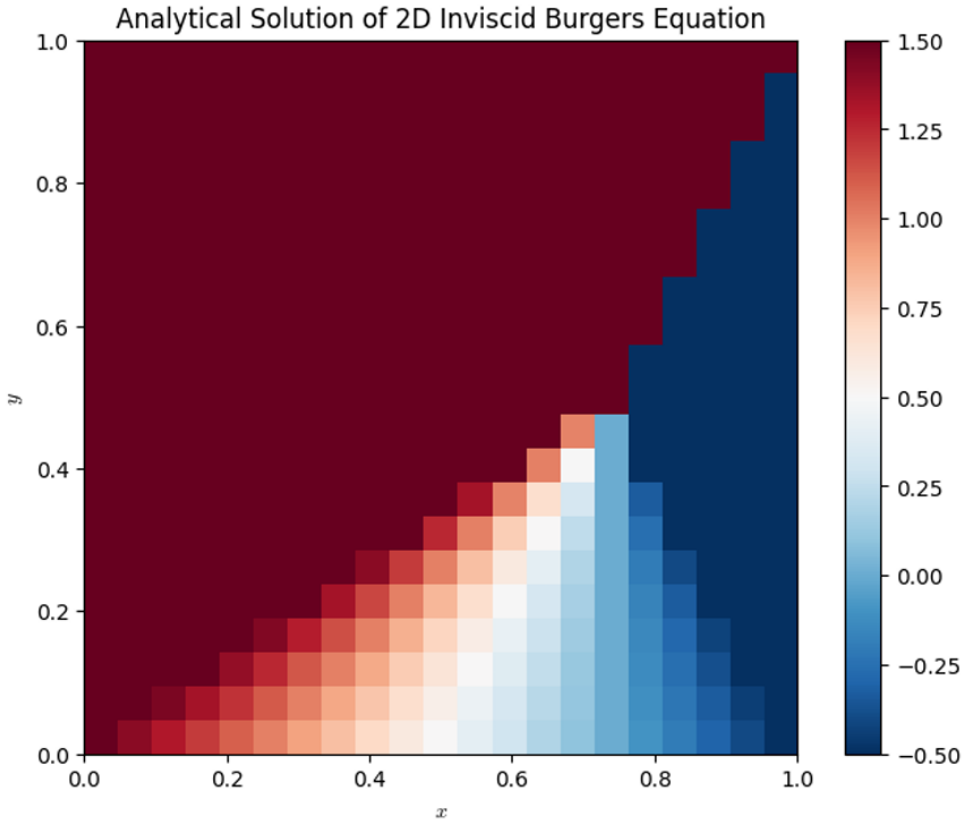
Analytical Solution Given

for $y \leq 0.5$

$$u(x,y) = \begin{cases} 1.5, & \text{if } x \leq 1.5y \\ \frac{1.5-2x}{1-2y}, & \text{if } 1.5y \leq x \leq (1-0.5y) \\ -0.5, & \text{if } x \geq (1-0.5y) \end{cases}$$

for $y \geq 0.5$

$$u(x,y) = \begin{cases} 1.5, & \text{if } x \leq (0.5+0.5y) \\ -0.5, & \text{if } x > (0.5+0.5y) \end{cases}$$



Fundamental Theory

Mc-Cormack Method

$$\frac{\partial u}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} = 0$$

$$E = f(u) = \frac{u^2}{2}, F = g(u) = u$$

$$u_i^* = u_i^n - \frac{\Delta t}{\Delta x}\left(E_{i+1}^n - E_i^n\right)$$

$$u_i^{**} = u_i^n - \alpha\frac{\Delta t}{\Delta x}\left(E_i^* - E_{i-1}^*\right)$$

$$u_i^{n+1} = \frac{1}{2}\left[u_i^{**} + u_i^*\right]$$

Space and Time Discretization

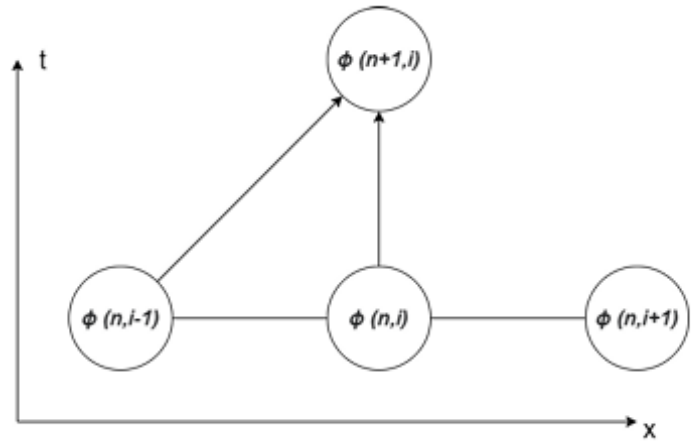


Figure 2.1 FTBS

$$U_i^{n+1} = U_i^n - \alpha \frac{\Delta t}{\Delta x}\left(U_i^n - U_{i-1}^n\right) \quad (2.8)$$

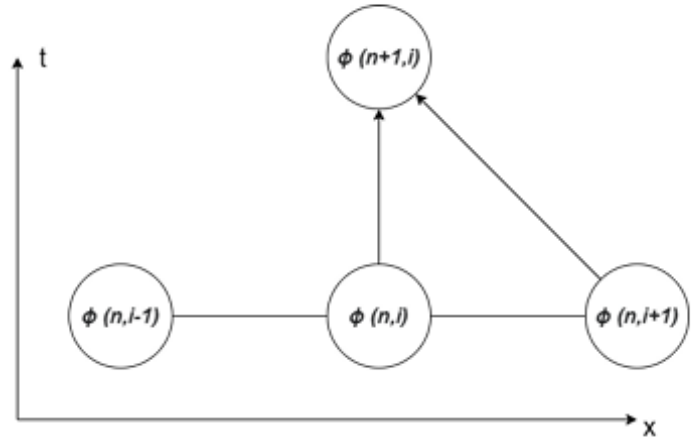


Figure 2.2 FTFS

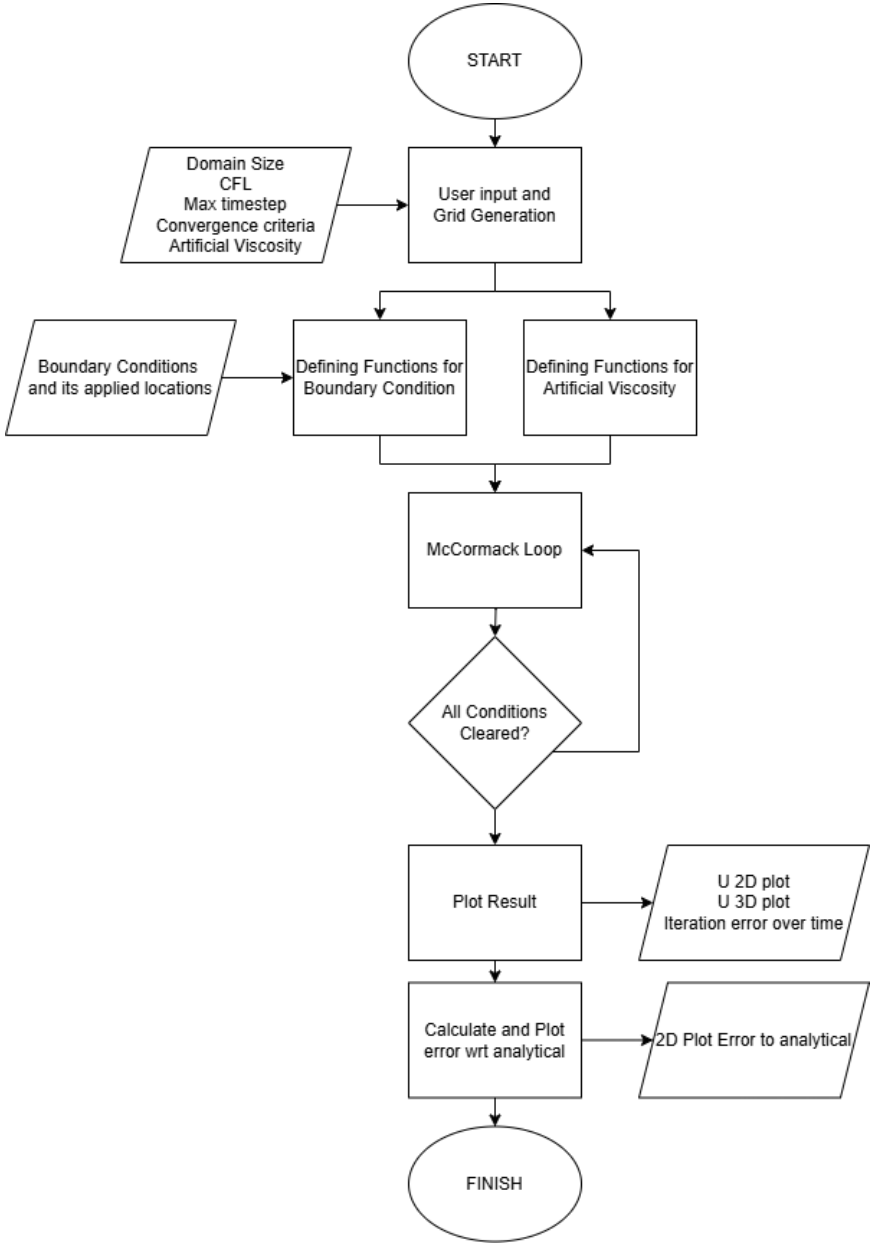
$$U_i^{n+1} = U_i^n - \alpha \frac{\Delta t}{\Delta x}\left(U_{i+1}^n - U_i^n\right) \quad (2.9)$$

$$
\begin{aligned}
S^t_{(i,j)} =& C_x \frac{|p^t_{(i+1,j)} - 2p^t_{(i,j)} + p^t_{(i-1,j)}|}{p^t_{(i+1,j)} + 2p^t_{(i,j)} + p^t_{(i-1,j)}} \left(U^t_{(i+1,j)} - 2U^t_{(i,j)} + U^t_{(i-1,j)}\right) \\
&+ C_y \frac{|p^t_{(i,j+1)} - 2p^t_{(i,j)} + p^t_{(i,j-1)}|}{p^t_{(i,j+1)} + 2p^t_{(i,j)} + p^t_{(i,j-1)}} \left(U^t_{(i,j+1)} - 2U^t_{(i,j)} + U^t_{(i,j-1)}\right)
\end{aligned}
$$

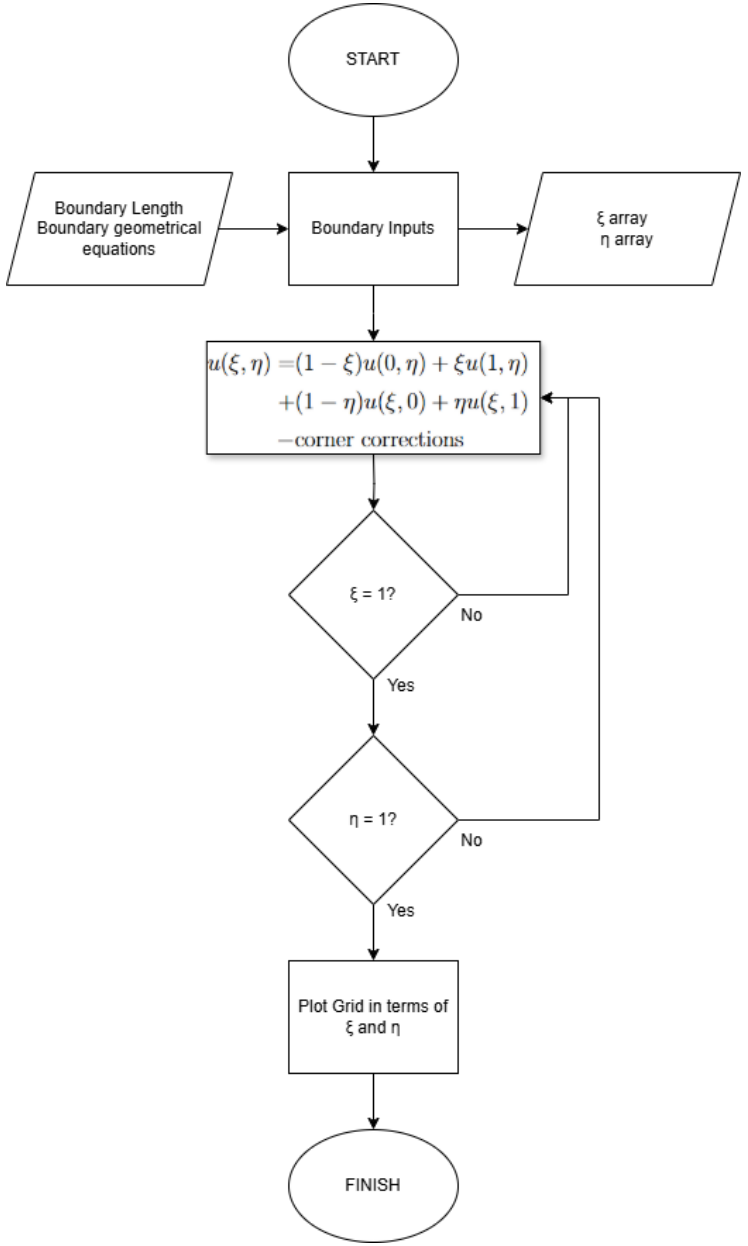
Artificial Viscosity

Program Workflow and Sub-routine

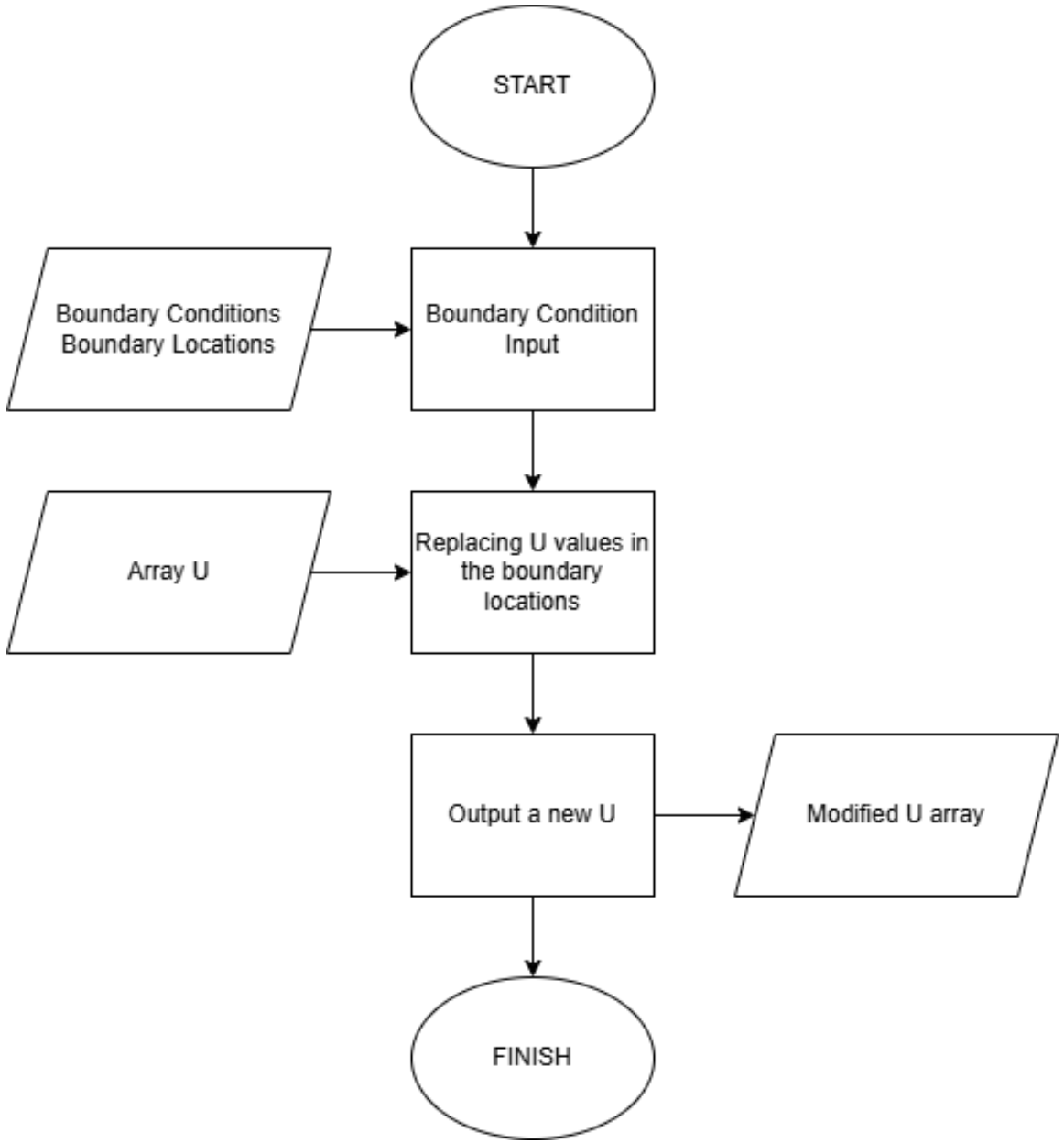
Main Loop


START
Domain Size
CFL
Max timestep
Convergence criteria
Artificial Viscosity
User input and
Grid Generation
Boundary Conditions
and its applied locations
Defining Functions for
Boundary Condition
Defining Functions for
Artificial Viscosity
McCormack Loop
All Conditions
Cleared?
Plot Result
U 2D plot
U 3D plot
Iteration error over time
Calculate and Plot
error wrt analytical
2D Plot Error to analytical
FINISH

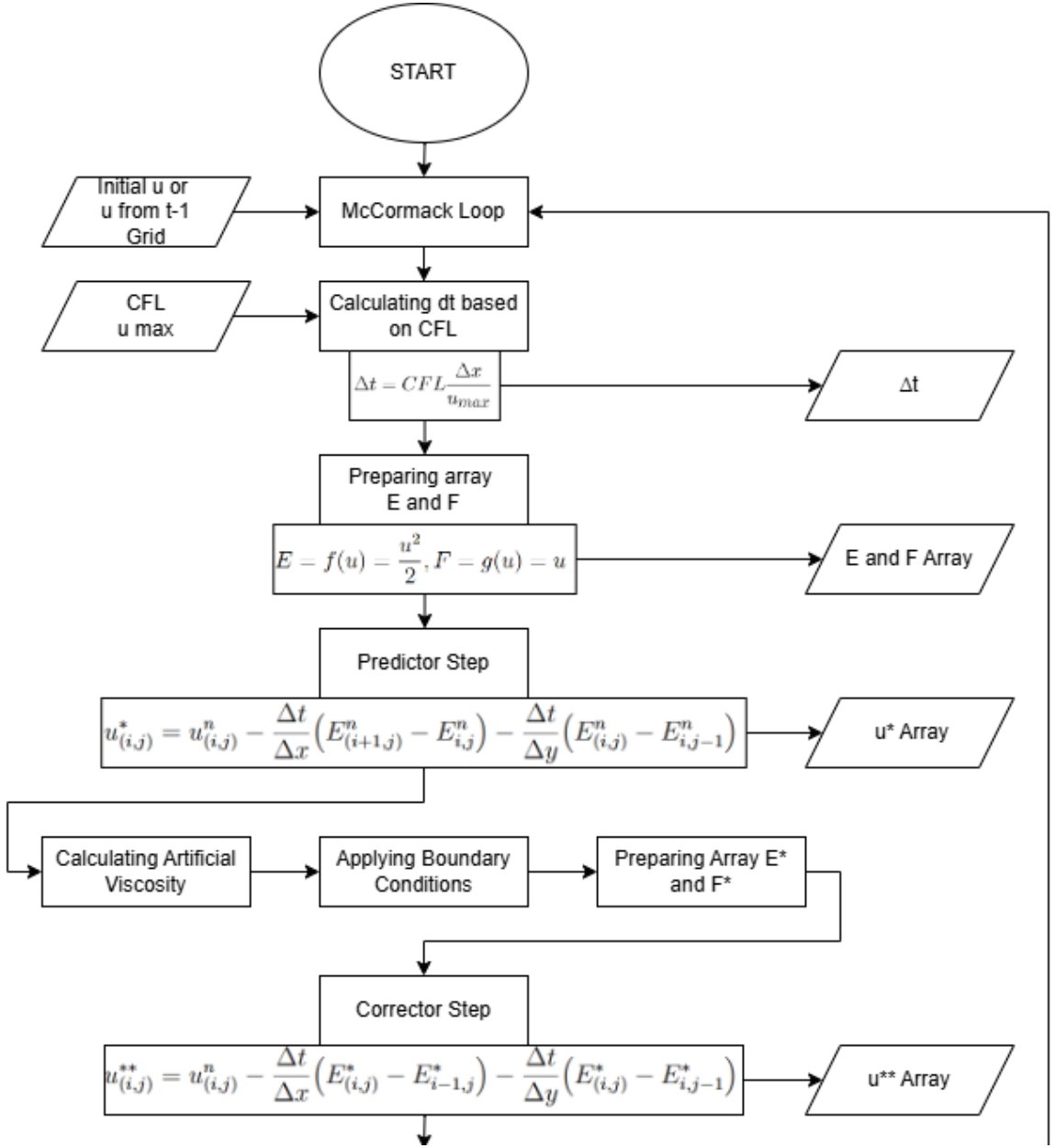
Grid Generation With TFI



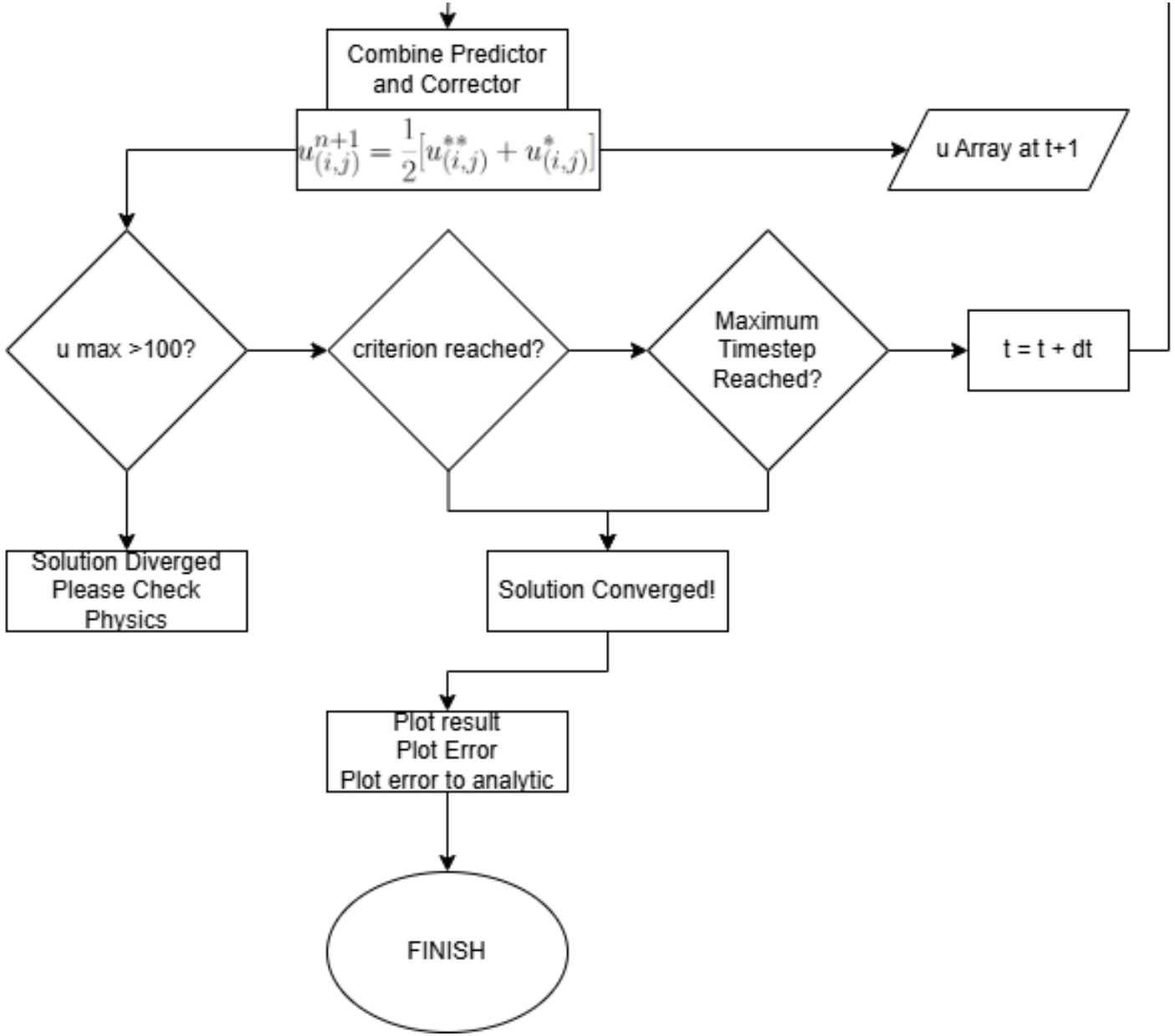
Boundary Condition



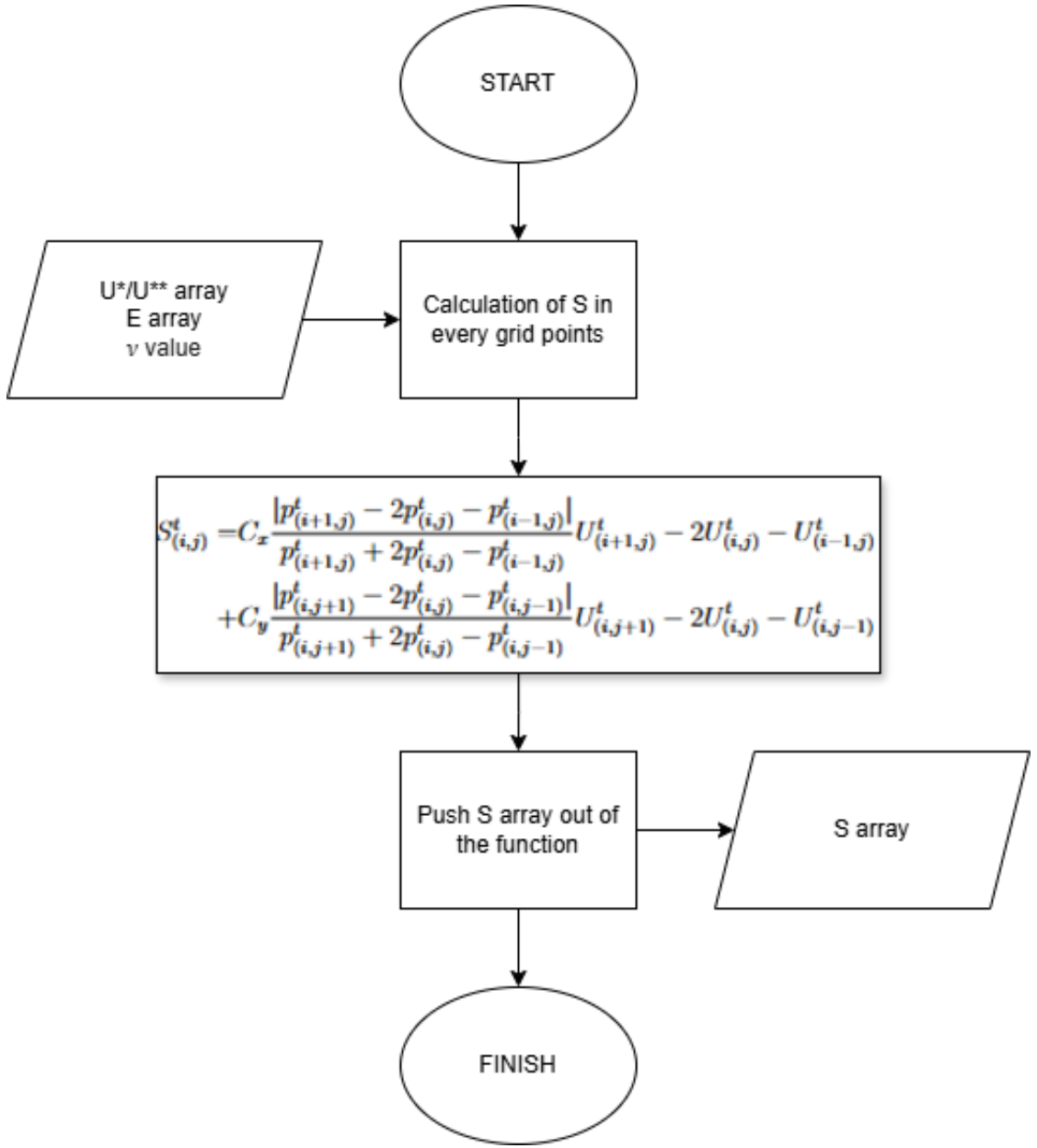
Mc-Cormack Loop [1/2]



Mc-Cormack Loop [2/2]



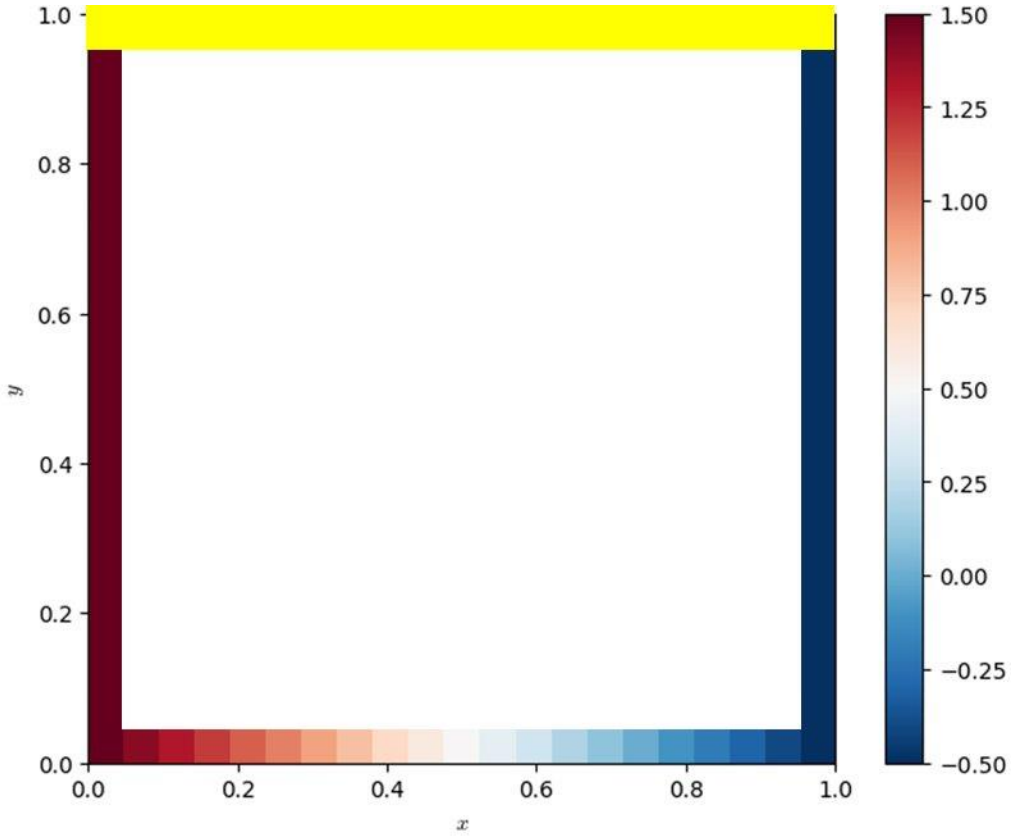
Artificial Viscosity



Modified Mc-Cormack Method

$$u^{*}_{(i,j)} = u^{n}_{(i,j)} - \frac{\Delta t}{\Delta x}\left(E^{n}_{(i+1,j)} - E^{n}_{i,j}\right) - \frac{\Delta t}{\Delta y}\left(F^{n}_{(i,j)} - F^{n}_{i,j-1}\right)$$

$$u^{**}_{(i,j)} = u^{n}_{(i,j)} - \frac{\Delta t}{\Delta x}\left(E^{*}_{(i,j)} - E^{*}_{i-1,j}\right) - \frac{\Delta t}{\Delta y}\left(F^{*}_{(i,j)} - F^{*}_{i,j-1}\right)$$



Numerical Result

Grid Comparisons 21 by 21

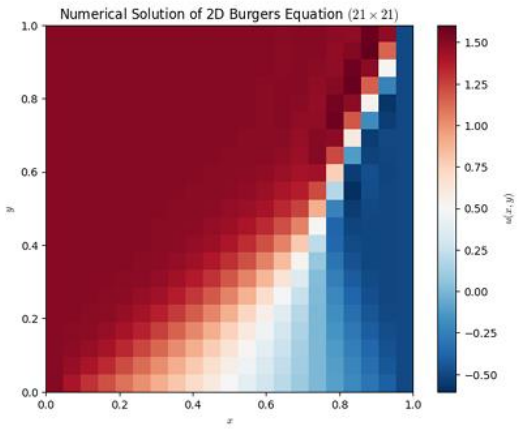


Figure 4.9 2D contour plot of numerical solution for 21 by 21 grid

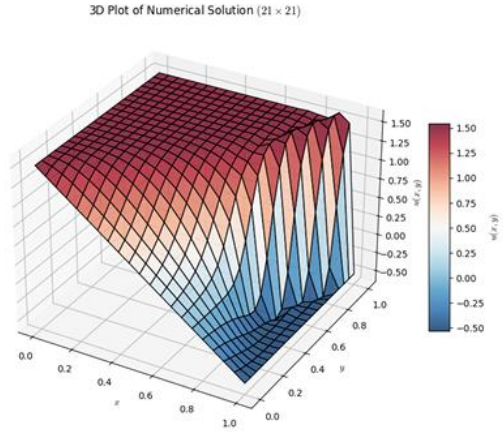


Figure 4.10 3D plot of numerical solution for 21 by 21 grid

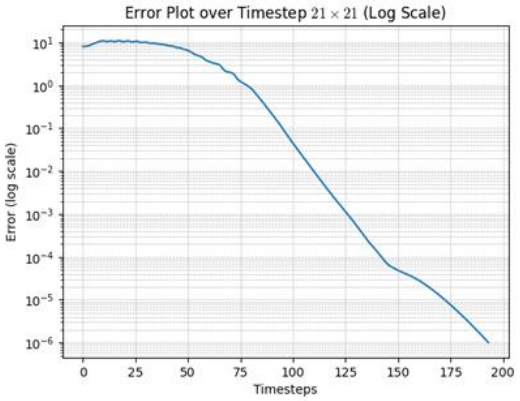


Figure 4.11 Error plot of numerical solution for 21 by 21 grid

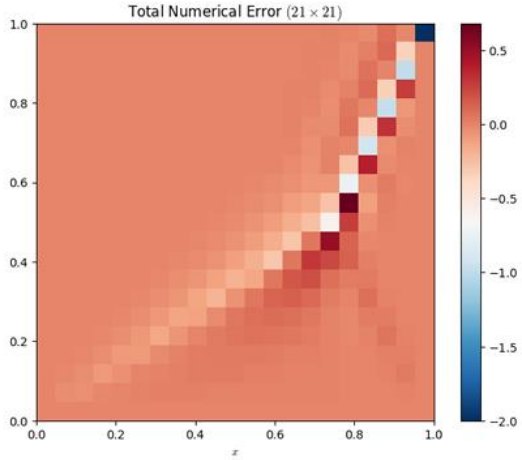


Figure 4.12 Total numerical error for 21 by 21 grid

Grid Comparisons 41 by 41

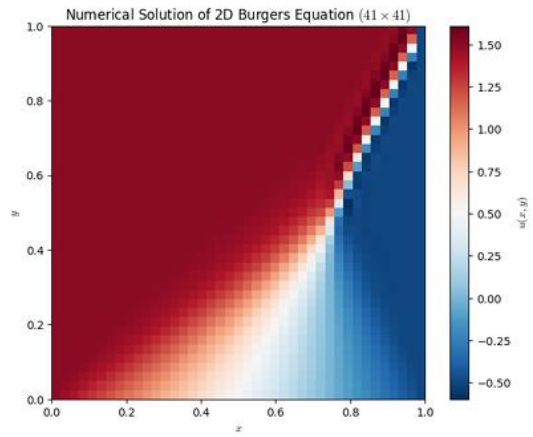


Figure 4.13 2D contour plot of numerical solution for 41 by 41 grid

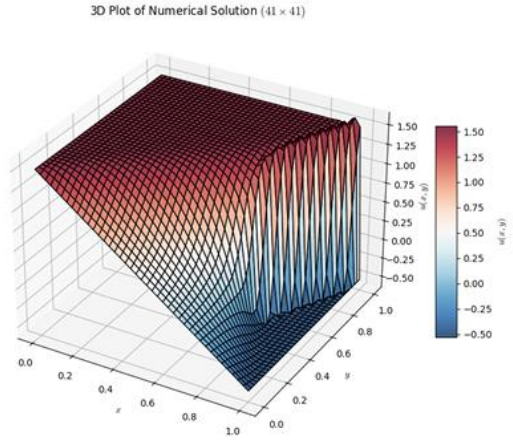


Figure 4.14 3D plot of numerical solution for 41 by 41 grid

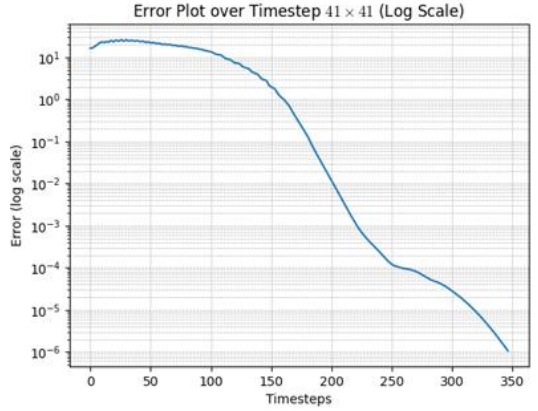


Figure 4.15 Error plot of numerical solution for 41 by 41 grid

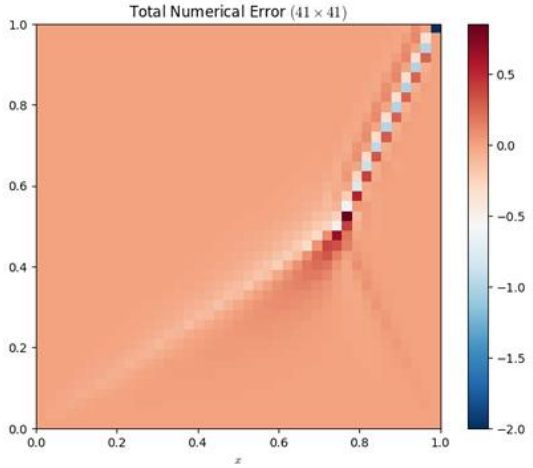


Figure 4.16 Total numerical error for 41 by 41 grid

Artificial Viscosity Study [1/2]

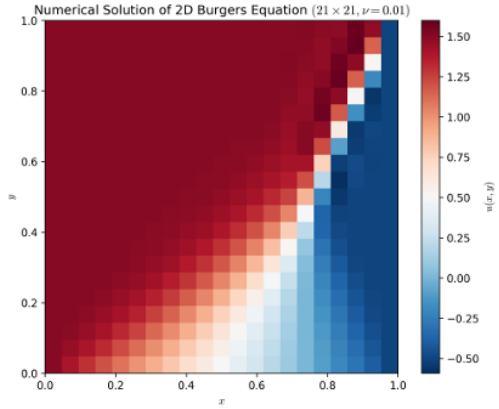


Figure 4.17 2D contour plot of numerical solution with $\nu = 0.01$

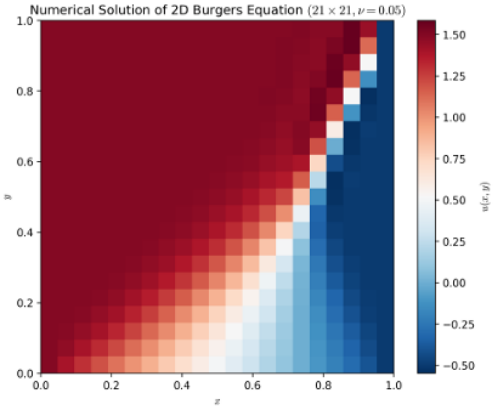


Figure 4.18 2D contour plot of numerical solution with $\nu = 0.05$

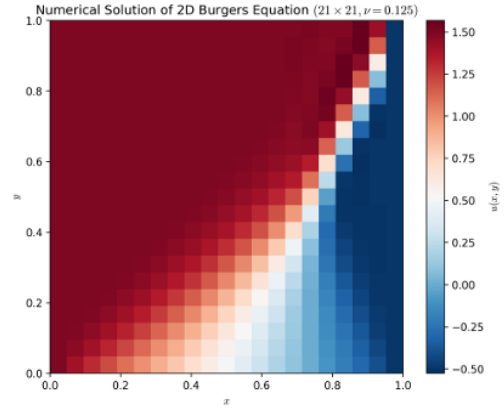


Figure 4.19 2D contour plot of numerical solution with $\nu = 0.125$

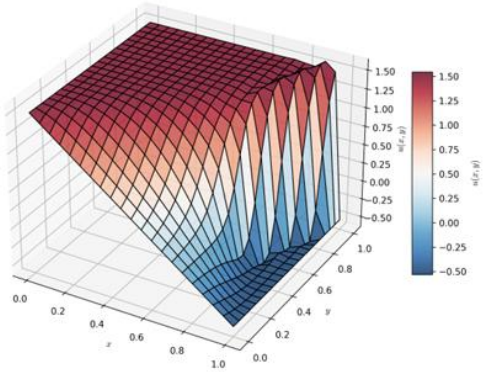


Figure 4.20 3D contour plot of numerical solution with $\nu = 0.01$

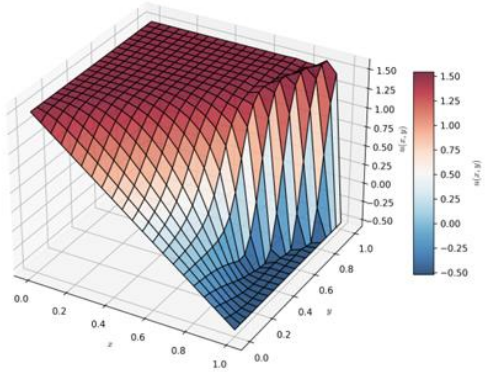


Figure 4.21 3D contour plot of numerical solution with $\nu = 0.05$

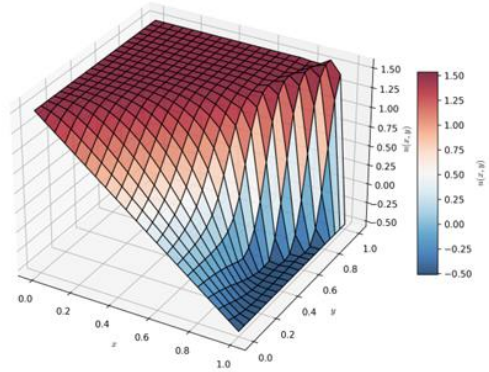


Figure 4.22 3D contour plot of numerical solution with $\nu = 0.125$

Grid	$\nu = 0$	$\nu = 0.01$	$\nu = 0.05$	$\nu = 0.125$
21×21	19.84826	19.89419	20.31916	21.33440
41×41	42.22017	42.30640	42.86701	43.54519

Artificial Viscosity Study [2/2]

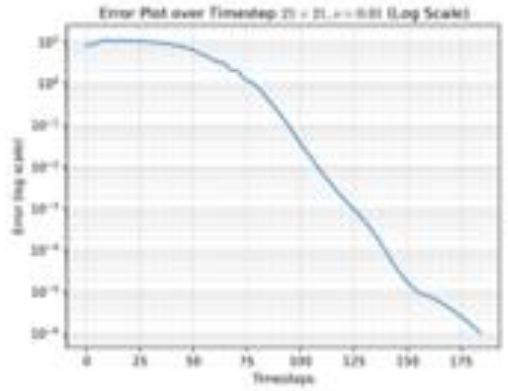


Figure 4.23 Numerical errors w.r.t iterations with $\nu = 0.01$

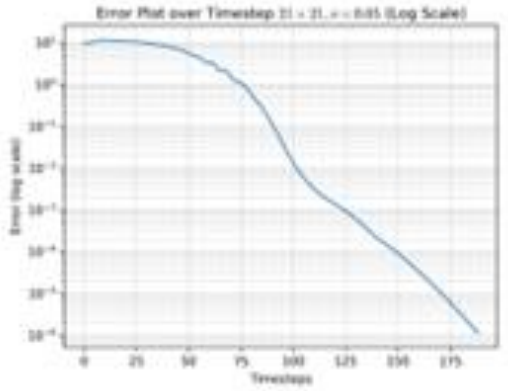


Figure 4.24 Numerical errors w.r.t iterations with $\nu = 0.05$

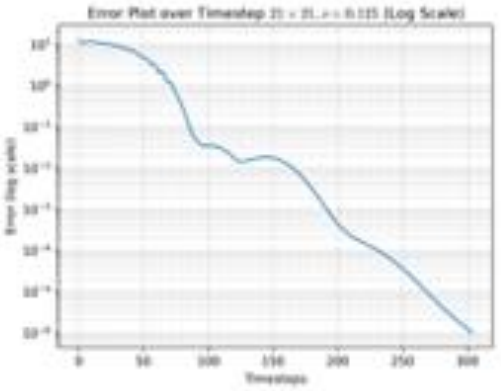


Figure 4.25 Numerical errors w.r.t iterations with $\nu = 0.125$

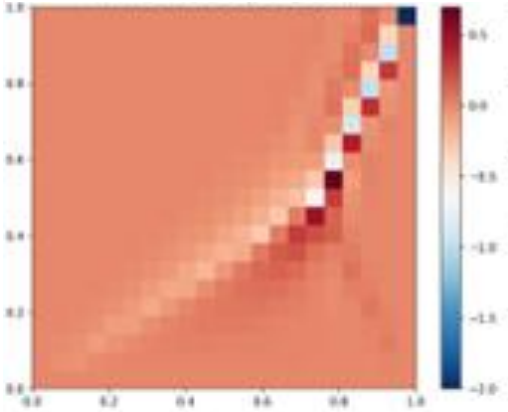


Figure 4.26 2D contour plot of error wrt analytical with $\nu = 0.01$

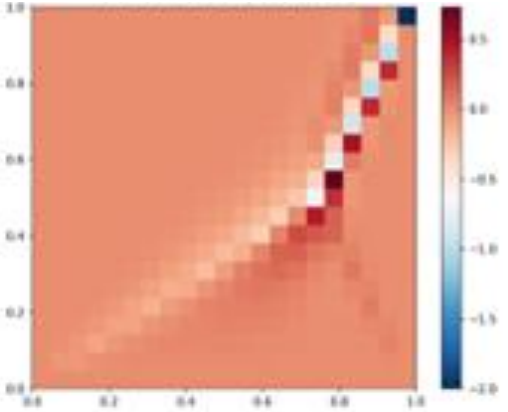


Figure 4.27 2D contour plot of error wrt analytical with $\nu = 0.05$

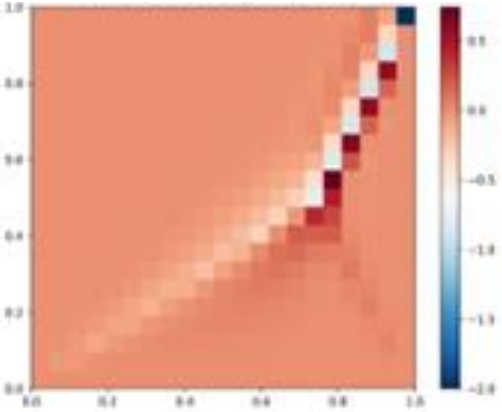
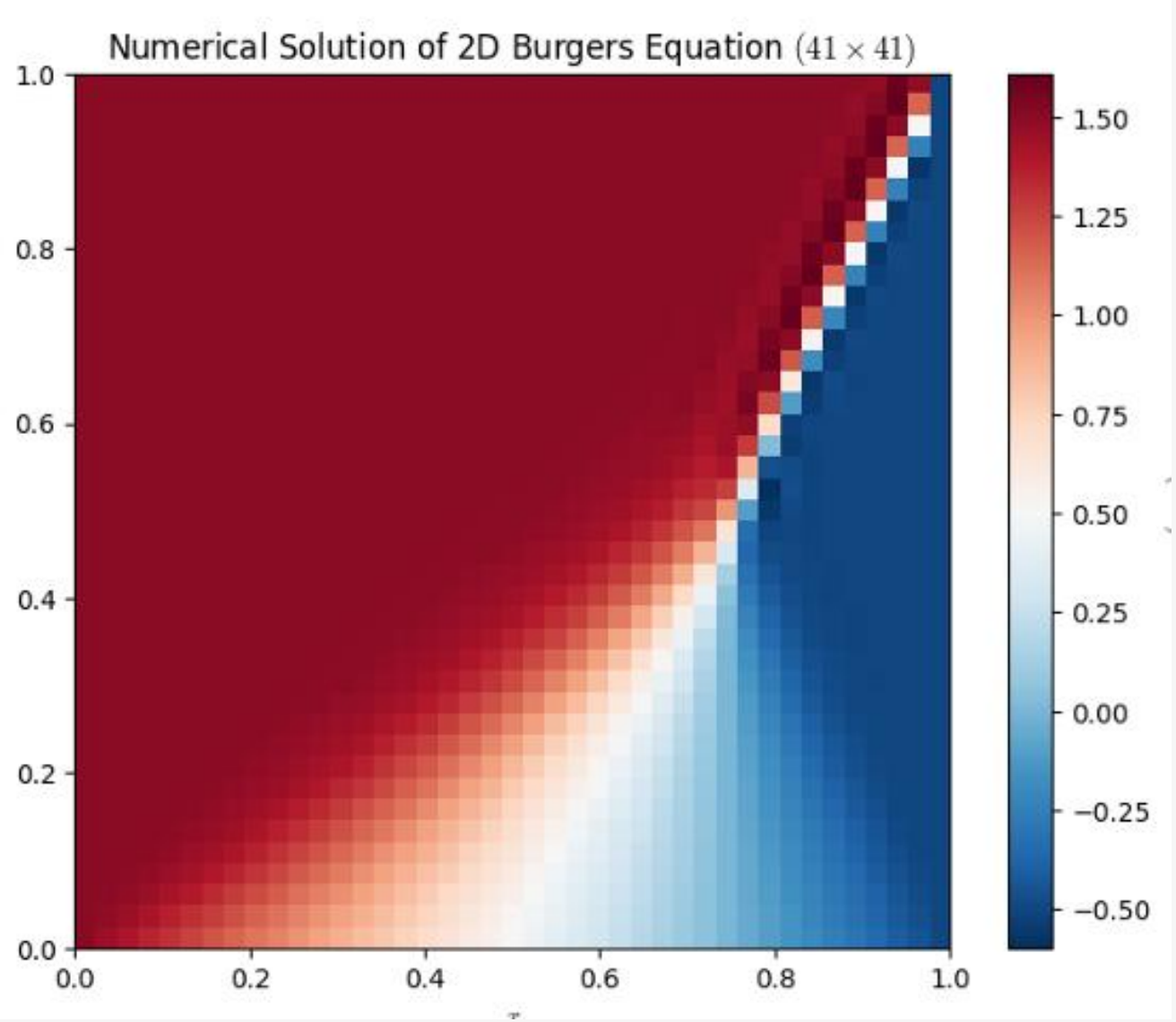


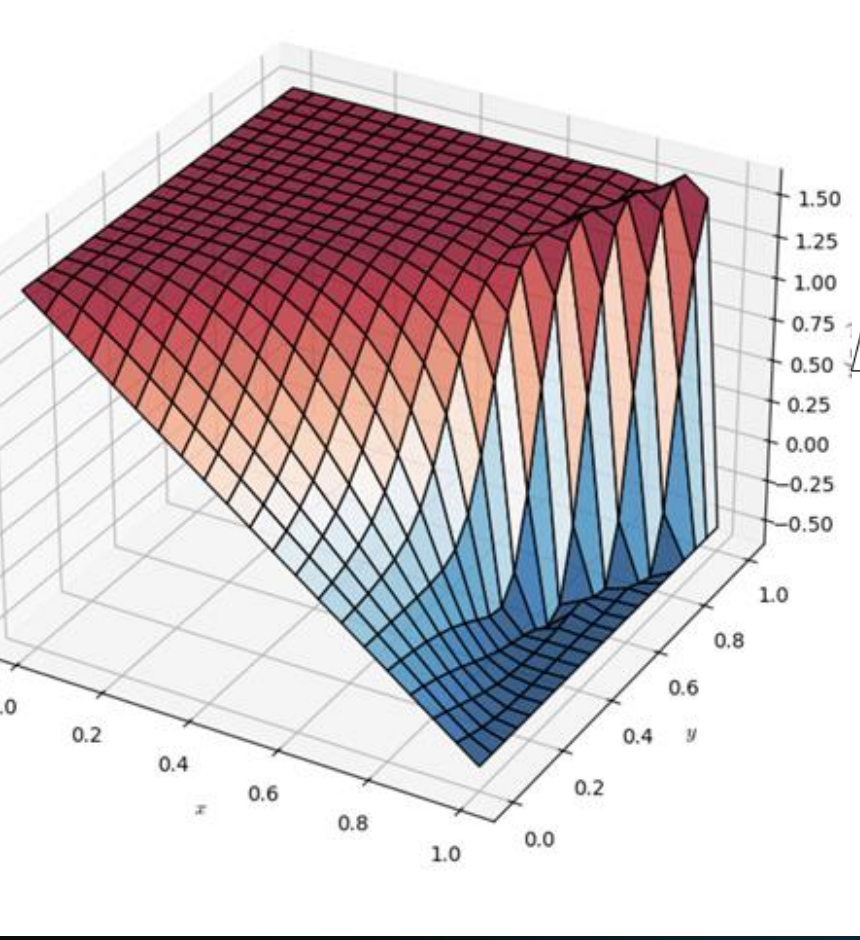
Figure 4.28 2D contour plot of error wrt analytical with $\nu = 0.125$

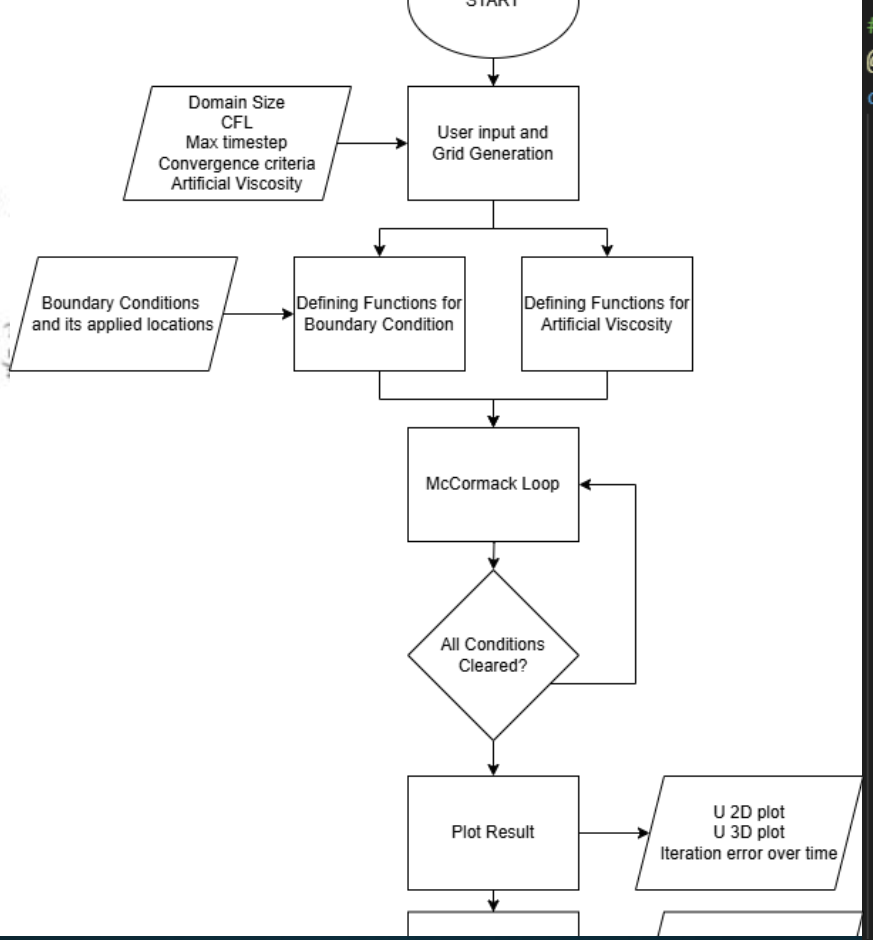
Conclusion



Conclusion

1. The derivation of 2-dimensional Burger's equation using McCormack has been performed with its appropriate scheme,
2. The grid is generated by using a transfinite interpolation method as stated in chapter 1.1,
3. The numerical and analytic solutions are calculated based on the grid of N =21 and N =41, with satisfying result in numerical convergence wise,
4. The solution of characteristic equation are also performed and approached by mathematical derivation,
5. The effects on artificial viscosity, while the solution is affected by it. It is only affecting it in adding additional errors and iterations, while having a little effects on solution stability. Further studies would need to be required to have a great dissipation terms in a burger's equation.






```python
### ------------ FUNCTIONS NEEDED TO DEFINE BOUNDARY AND VIS
@njit(cache = True)
def viscosity(u, nu, stars):
    ''' To calculate artificial viscosity term: S_{i, j} = S
    # Make S_i and S_j array to store each value
    S_i = np.zeros_like(u)
    S_j = np.zeros_like(u)

    # Looping through each index
    for j in range(1, ny):
        for i in range(1, nx-1):
            S_i[j, i] = abs(u[j, i+1] - 2*u[j, i] + u[j, i-1
            S_j[j, i] = abs(u[j  , i] - 2*u[j, i] + u[j-1, i

    # Total viscosity
    S = nu*(S_i + S_j)

    # buat set viscosity di boundary
    for j in range(ny):
        for i in range(nx):
            # At x = 0, left
            S[j, 0] = nu

            # at x = 1, right
            S[j, -1] = nu                    # minus index means

            # at y = 0, bottom
```

Any Inquires?

```python
# Buat nampung error
error_progress = np.zeros(timesteps)

# MacCormack scheme
@njit(cache = True)
def simulate_cormeck (u, timesteps, dx, dy, error_progress, final_t):
    for t in range(timesteps):

        # Update max_u untuk update dt
        max_u = np.max(np.abs(u))
        if max_u > 100:
            print('The solutions Diverged, Please Check the appropriate physiscs settings')
            break
        dt = CFL * min(dx, dy) / (max_u)

        # Calculate the conservative variable
        E = u**2 / 2
        F = u.copy()

        ''' Predictor step '''
        u_star = u.copy()
        for j in range(1, ny):
            for i in range(1, nx - 1):
                u_star[j, i] = u[j, i] - dt/dx * (E[j, i+1] - E[j, i]) - dt/dy * (F[j, i] - F[j-1, i])

        # # Artificial Viscosity
```