MAE 561 Final Project: Two-Dimensional Mixing Chamber

Anushka Subedi

ASU ID: 1225812200

Contents

Abstract

This study delves into fluid flow and mixing dynamics in a two-dimensional rectangular mixing chamber using computational modeling. It investigates the behavior of two chemical species absorbed by rotating disks, employing advanced numerical methods. The Immersed Boundary method, along with fractional step schemes like Adams-Bashforth/Crank Nicolson for momentum equations and WENO5-TVD-RK-3/Crank Nicolson for mass fraction equations, is utilized. Performance metrics such as average kinetic energy and mixing parameters are rigorously assessed. Findings provide detailed insights into flow patterns, species distribution, and mixing efficiency, fostering insightful discussions.

1 Problem Statement/Governing Equations

The problem at hand entails the computational modeling of fluid flow and mixing within a two-dimensional rectangular mixing chamber. This chamber as shown in Figure (1), with dimensions $L_x = 3$ and $L_y = 2$, contains rotating disks that absorb two chemical species. The objective is to analyze the behavior of the fluid flow and the distribution of the chemical species over time. To achieve this, the non-dimensional continuity equation, 2D Navier-Stokes equations, and a convective/diffusive transport equation for mass fraction Y are to be solved numerically. Additionally, the performance of the mixing chamber is to be assessed using metrics such as the average kinetic energy (K) and mixing parameter (R) over the first 10 time units. Numerical methods such as the Immersed Boundary method, fractional step schemes, and specific spatial/temporal methods are employed to solve the governing equations. The study aims to provide comprehensive insights into the flow patterns, species distribution, and mixing efficiency within the mixing chamber, facilitating informed discussions and conclusions.

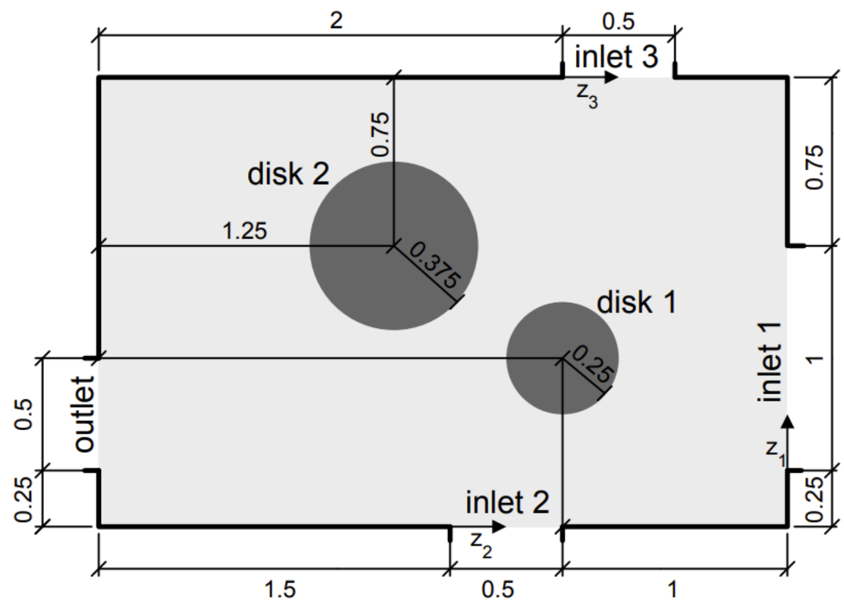

Figure 1: Mixing Chamber Geometry
[taken from assignment statement]

The governing equations to be solved for this problem are the non-dimensional continuity equation, 2D Navier-Stokes equation, and a convective/diffusive transport equation for the mass fraction term. The non-dimensional 2D incompressible continuity equation is:

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

The momentum equations are:

$$\frac{\partial u}{\partial t} + \frac{\partial uu}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$
$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial vv}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

Non-dimensionalizing them:

$$\bar{x} = \frac{x}{L}, \quad \bar{y} = \frac{y}{L}, \quad \bar{u} = \frac{u}{u_{\text{ref}}}, \quad \bar{v} = \frac{v}{v_{\text{ref}}}, \quad \bar{t} = \frac{t}{\frac{L}{u_{\text{ref}}}}, \quad \bar{p} = \frac{p}{\rho \cdot u_{\text{ref}}^2}$$

The equations to solve, now, after dropping the bars are:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{1}$$

$$\frac{\partial u}{\partial t} + \frac{\partial uu}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + Q_u(x,y,t) \tag{2}$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial vv}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + Q_v(x,y,t) \tag{3}$$

where,

$$Re = \frac{u_{ref}L}{\nu}$$

For the mass fraction, the following convective/diffusive transport equation will solved:

$$\frac{\partial Y}{\partial t} + u\frac{\partial Y}{\partial x} + v\frac{\partial Y}{\partial y} = \frac{1}{ReSc}\left(\frac{\partial^2 Y}{\partial x^2} + \frac{\partial^2 Y}{\partial y^2}\right) + Q_Y(x,y,t) \tag{4}$$

where u is the non-dimensional velocity in the x-direction, v is the non-dimensional velocity in the y-direction, Y is the mass fraction of a chemical species, Re is the Reynolds number, and Sc is the Schmidt number. The fluid in the chamber is initially at rest with $Y(x,y,t=0)=0$. At the initial time, inlets 1, 2, and 3 are opened resulting in non-dimensional inflow of

$$\begin{aligned}
\text{inlet1}: u_{\text{in},1}(z_1) &= U_1\cos(\alpha_1)f(z_1), \quad v_{\text{in},1}(z_1) = U_1\sin(\alpha_1)f(z_1), \quad Y_{\text{in},1} = 1\\
\text{inlet2}: u_{\text{in},2}(z_2) &= U_2\cos(\alpha_2)f(z_2), \quad v_{\text{in},2}(z_2) = U_2\sin(\alpha_2)f(z_2), \quad Y_{\text{in},2} = 0\\
\text{inlet3}: u_{\text{in},3}(z_3) &= U_3\cos(\alpha_3)f(z_3), \quad v_{\text{in},3}(z_3) = U_3\sin(\alpha_3)f(z_3), \quad Y_{\text{in},3} = 0
\end{aligned}$$

where $U_1 = \frac{3}{4}$, $U_2 = 2$, $U_3 = 3$, $\alpha_1 = \pi$, $\alpha_2 = \frac{\pi}{3}$, $\alpha_3 = -\frac{\pi}{6}$, and z_i is a non-dimensional coordinate along the inlets as indicated in Fig. 1, i.e., $z_i = 0$ at one edge of the inlet and $z_i = 1$ at the other edge, e.g., $z_1 = \frac{y-0.25}{1}$, $z_2 = \frac{x-1.5}{0.5}$, with

$$f(z) = 6z(1-z) \tag{5}$$

In the outlet:

$$\left.\frac{\partial u}{\partial x}\right|_{\text{out}} = 0, \quad \left.\frac{\partial v}{\partial x}\right|_{\text{out}} = 0, \quad \left.\frac{\partial Y}{\partial x}\right|_{\text{out}} = 0. \tag{6}$$

Treat points belonging to both the outlet and wall as belonging to the outlet only. All mixing chamber walls are no-slip and have zero transport of the chemical species, i.e.,

$$\text{walls}: \quad u_{\text{wall}} = 0, \quad v_{\text{wall}} = 0, \quad \left.\frac{\partial Y}{\partial n}\right|_{\text{wall}} = 0. \tag{7}$$

The performance of the mixing chamber shall be measured by the time-averaged kinetic energy $\bar{K}$ in the chamber:

$$\bar{K} = \frac{1}{10}\int_0^{10} k(t)\,dt, \tag{8}$$

where

$$k(t) = \int_0^{L_y}\int_0^{L_x} \frac{1}{2}(\hat{u}^2 + \hat{v}^2)\,dx\,dy \tag{9}$$

with $\hat{u}$ and $\hat{v}$ velocities at cell centers determined from the staggered velocities u and v by 2nd-order averaging, and the time-averaged scalar mixing parameter $\bar{R}$.

$$\bar{R} = \frac{1}{10}\int_0^{10} S(t)\,dt, \tag{10}$$

where,

$$S(t) = \frac{1}{L_x L_y}\int_0^{L_y}\int_0^{L_x} Y(1-Y)\,dx\,dy \tag{11}$$

2 Numerical Methods

For the Numerical Methods, the Immersed Boundary method for the cylinders (not developed here but supplied as the source term), along with fractional step schemes, Adams-Bashforth/Crank Nicolson for momentum equations, and WENO5-TVD-RK-3/Crank Nicolson for mass fraction equations are developed and employed.

A tabular summary containing the names of all numerical methods used to solve the project and their formal order of accuracy is presented in Figure (2),(3), and (4).

	convective terms/elliptic terms				viscous/diffusive terms				boundary conditions	immersed boundary	variable location
	name time	order time	name space	order space	name time	order time	name space	order space	order space	order space	
u-equation	Adams-Bashforth	2	central	2	Crank-Nicolson	2	Central	2	2	1	u(i+1/2,j)
v-equation	Adams-Bashforth	2	central	2	Crank-Nicolson	2	Central	2	2	1	v(I,j+1/2)
Y-equation	TVD-RK-3	3	WENO-5	5	Crank-Nicolson	2	Central	2	2	1	Y(I,j)

Figure 2: Numerical methods for Convective and Diffusive Terms

		Numerical Method		Poisson Solver	boundary condition of phi, bcGS
	Fractional Step method: Projection Method	Overall Scheme	Technique on each mesh level	order space	order space
Poisson Equation	i. CN for viscous terms ii.Divergence Free Pressure Poisson iii.Projection into solenoidal subspace	Multigrid V-cycle Method	Gauss-Seidel	2	2

Figure 3: Numerical methods for Poisson Equation

Performance Metrics		Numerical Method		
	name space	order space	name time	order time
K_bar	averaging-2nd order	2	trapezoidal	2
S_bar	averaging-2nd order	2	trapezoidal	2

Figure 4: Numerical methods for Poisson Equation

Finite difference equations of the Numerical Methods:
The finite difference equations for solving the u and v-convective predictor terms, u and v-viscous predictor terms, Y-convective terms and Y-viscous terms, along with the u and v projection equations, method of solving the Pressure poisson equation are written in this section.
1. The finite difference equation u for predictor step convective terms are:

$$\begin{aligned}
\left.\frac{\partial uu}{\partial x}\right|_{i+1/2,j} &= \frac{(u_{i+1,j})^2 - (u_{i,j})^2}{\Delta x} + O(\Delta x^2) \\
\left.\frac{\partial uv}{\partial y}\right|_{i+1/2,j} &= \frac{(u_{i+1,j+1/2})^2 - uv_{i+1/2,j-1/2}}{\Delta y} + O(\Delta y^2)
\end{aligned}$$

where,

$$\begin{aligned}
u_{i,j} &= \frac{1}{2}\left(u_{i+1/2,j} + u_{i-1/2,j}\right) \\
u_{i+1/2,j+1/2} &= \frac{1}{2}\left(u_{i+1/2,j} + u_{i+1/2,j+1}\right) \\
v_{i+1/2,j+1/2} &= \frac{1}{2}\left(v_{i,j+1/2} + v_{i+1,j+1/2}\right)
\end{aligned}$$

Use adams's bashforth on this to solve in time i.e.

$$\frac{3}{2}\left[-\left.\frac{\partial uu}{\partial x}\right|^{n}_{i+1/2,j} - \left.\frac{\partial uv}{\partial y}\right|^{n}_{i+1/2,j}\right] - \frac{1}{2}\left[-\left.\frac{\partial uu}{\partial x}\right|^{n-1}_{i+1/2,j} - \left.\frac{\partial uv}{\partial y}\right|^{n-1}_{i+1/2,j}\right]$$

It can then be added to the Qu term.
2. The sub-index finite difference equation u for predictor step viscous terms are:

$$
\begin{aligned}
\left.\frac{\partial^2 u}{\partial x^2}\right|_{i+1/2,j} &= \frac{u_{i+3/2,j} - 2u_{i+1/2,j} + u_{i-1/2,j}}{\Delta x^2} + O(\Delta x^2) \\
\left.\frac{\partial^2 u}{\partial y^2}\right|_{i+1/2,j} &= \frac{u_{i+1/2,j+1} - 2u_{i+1/2,j} + u_{i+1/2,j-1}}{\Delta y^2} + O(\Delta y^2)
\end{aligned}
$$

Use Crank Nicolson-ADI to solve these first in step one in x-direction and then in y-direction.
Step 1:

$$
d_1 u^{n+1/2}_{i+3/2,j} + (1+2d_1)u^{n+1/2}_{i+1/2,j} - d_1 u^{n+1/2}_{i-1/2,j} = d_2 u^n_{i+1/2,j+1} + (1-2d_2)u^n_{i+1/2,j} + d_2 u^n_{i+1/2,j-1} + \frac{\Delta t}{2} Qu(i+1/2,j)
$$

Step 2:

$$
d_2 u^{n+1}_{i+1/2,j+1} + (1+2d_2)u^{n+1}_{i+1/2,j} - d_2 u^{n+1}_{i+1/2,j-1} = d_1 u^{n+1/2}_{i+3/2,j} + (1-2d_1)u^{n+1/2}_{i+1/2,j} + d_1 u^{n+1/2}_{i-1/2,j} + \frac{\Delta t}{2} Qu(i,j+1/2)
$$

where,

$$
d_1 = \frac{d_x}{2}, \quad d_2 = \frac{d_y}{2}, \quad d_x = \frac{\Delta t}{Re\Delta x^2}, \quad d_y = \frac{\Delta t}{Re\Delta y^2}
$$

3. The finite difference equation v for predictor step convective terms are:

$$
\begin{aligned}
\left.\frac{\partial uv}{\partial x}\right|_{i,j+1/2} &= \frac{(uv)_{i+1/2,j+1/2} - (uv)_{i-1/2,j+1/2}}{\Delta x} + O(\Delta x^2) \\
\left.\frac{\partial vv}{\partial y}\right|_{i,j+1/2} &= \frac{(v_{i,j+1})^2 - (v_{i,j})^2}{\Delta y} + O(\Delta y^2)
\end{aligned}
$$

where,

$$
\begin{aligned}
v_{i,j} &= \frac{1}{2}\left(v_{i,j+1/2} + v_{i,j-1/2}\right) \\
u_{i+1/2,j+1/2} &= \frac{1}{2}\left(u_{i+1/2,j} + u_{i+1/2,j+1}\right) \\
v_{i+1/2,j+1/2} &= \frac{1}{2}\left(v_{i,j+1/2} + v_{i+1,j+1/2}\right)
\end{aligned}
$$

Use adams's bashforth on this to solve in time i.e.

$$
\frac{3}{2}\left[-\left.\frac{\partial uv}{\partial x}\right|^n_{i,j+1/2} - \left.\frac{\partial vv}{\partial y}\right|^n_{i,j+1/2}\right] - \frac{1}{2}\left[\left.\frac{\partial uv}{\partial x}\right|^{n-1}_{i,j+1/2} - \left.\frac{\partial vv}{\partial y}\right|^{n-1}_{i,j+1/2}\right]
$$

It can then be added to the Qv term.
4. The sub-index finite difference equation v for predictor step viscous terms is:

$$
\begin{aligned}
\left.\frac{\partial^2 v}{\partial x^2}\right|_{i,j+1/2} &= \frac{v_{i+1,j+1/2} - 2v_{i,j+1/2} + v_{i-1,j+1/2}}{\Delta x^2} + O(\Delta x^2) \\
\left.\frac{\partial^2 v}{\partial y^2}\right|_{i,j+1/2} &= \frac{v_{i,j+3/2} - 2v_{i,j+1/2} + v_{i,j-1/2}}{\Delta y^2} + O(\Delta y^2)
\end{aligned}
$$

Use Crank Nicolson-ADI to solve these first in step one in x-direction and then in y-direction.
Step 1:

$$d_1 v_{i,j+3/2}^{n+1/2} + (1+2d_1)v_{i,j+1/2}^{n+1/2} - d_1 v_{i,j-1/2}^{n+1/2} = d_2 v_{i+1,j+1/2}^{n} + (1-2d_2)v_{i,j+1/2}^{n} + d_2 v_{i-1,j+1/2}^{n} + \frac{\Delta t}{2} Qv(i+1/2,j)$$

Step 2:

$$d_2 v_{i+1,j+1/2}^{n+1} + (1+2d_2)v_{i,j+1/2}^{n+1} - d_2 v_{i-1,j+1/2}^{n+1} = d_1 v_{i,j+3/2}^{n+1/2} + (1-2d_1)v_{i,j+1/2}^{n+1/2} + d_1 v_{i,j-1/2}^{n+1/2} + frac\Delta t2Qv(i,j+1/2)$$

where,

$$d_1 = \frac{d_x}{2}, \quad d_2 = \frac{d_y}{2}, \quad d_x = \frac{\Delta t}{Re\Delta x^2}, \quad d_y = \frac{\Delta t}{Re\Delta y^2}$$

5. The sub-index finite difference equation Y convective is:
Note: here a and b in the equations are u and v respectively in the Y convective terms. The TVD-RK-3 method for time is:
Step 1:

$$Y_{i,j}^{(1)} = Y_{i,j}^{(n)} - \Delta t \left(a_{i,j}^n \frac{\partial Y^n}{\partial x}\bigg|_{i,j}^{\pm} + b_{i,j}^n \frac{\partial Y^n}{\partial y}\bigg|_{i,j}^{\pm} \right)$$

Apply BC to $Y_{i,j}^{(1)}$ at $t^n + \Delta t$.
Step 2:

$$Y_{i,j}^{(2)} = Y_{i,j}^{(1)} + \frac{3}{4}\Delta t \left(a_{i,j}^n \frac{\partial Y^n}{\partial x}\bigg|_{i,j}^{\pm} + b_{i,j}^n \frac{\partial Y^n}{\partial y}\bigg|_{i,j}^{\pm} \right) - \frac{1}{4}\Delta t \left(a_{i,j}^{n+1} \frac{\partial Y^{(1)}}{\partial x}\bigg|_{i,j}^{\pm} + b_{i,j}^{n+1} \frac{\partial Y^{(1)}}{\partial y}\bigg|_{i,j}^{\pm} \right)$$

Apply BC to $Y_{i,j}^{(2)}$ at $t^n + \frac{1}{2}\Delta t$.
Step 3:

$$\begin{aligned} Y_{i,j}^{n+1} = Y_{i,j}^{(2)} &+ \frac{1}{12}\Delta t \left(a_{i,j}^n \frac{\partial Y^n}{\partial x}\bigg|_{i,j}^{\pm} + b_{i,j}^n \frac{\partial Y^n}{\partial y}\bigg|_{i,j}^{\pm} \right) + \frac{1}{12}\Delta t \left(a_{i,j}^{n+1} \frac{\partial Y^{(1)}}{\partial x}\bigg|_{i,j}^{\pm} + b_{i,j}^{n+1} \frac{\partial Y^{(1)}}{\partial y}\bigg|_{i,j}^{\pm} \right) \\ &- \frac{2}{3}\Delta t \left(a_{i,j}^{n+1/2} \frac{\partial Y^{(2)}}{\partial x}\bigg|_{i,j}^{\pm} + b_{i,j}^{n+1/2} \frac{\partial Y^{(2)}}{\partial y}\bigg|_{i,j}^{\pm} \right) \end{aligned}$$

Apply BC to $Y_{i,j}^{(n+1)}$ at $t^n + \Delta t$.
The $\frac{\partial Y}{x}$ and $\frac{\partial Y}{y}$ parts are calculated using WENO-5:

$$\begin{aligned} \frac{\partial Y}{\partial x}\bigg|_{i,j}^{-} &= \frac{1}{12\Delta x}(-\Delta^+ Y_{i-2,j} + 7\Delta^+ Y_{i-1,j} + 7\Delta^+ Y_{i,j} - \Delta^+ Y_{i+1,j}) - \\ &\psi_{WENO}\left(\frac{\Delta^-\Delta^+ Y_{i-2,j}}{\Delta x}, \frac{\Delta^-\Delta^+ Y_{i-1,j}}{\Delta x}, \frac{\Delta^-\Delta^+ Y_{i,j}}{\Delta x}, \frac{\Delta^-\Delta^+ Y_{i+1,j}}{\Delta x} \right) \end{aligned}$$

with

$$\psi_{WENO}(a,b,c,d) = \frac{1}{3}\omega_0(a - 2b + c) + \frac{1}{6}\left(\omega_2 - \frac{1}{2}\right)(b - 2c + d)$$

with

$$\omega_0 = \frac{\alpha_0}{\alpha_0 + \alpha_1 + \alpha_2}, \quad \omega_2 = \frac{\alpha_2}{\alpha_0 + \alpha_1 + \alpha_2}$$

with

$$\alpha_0 = \frac{1}{(\epsilon + IS_0)^2}, \quad \alpha_1 = \frac{6}{(\epsilon + IS_1)^2}, \quad \alpha_2 = \frac{3}{(\epsilon + IS_2)^2} \qquad \epsilon = 10^{-6}$$

with

$$IS_0 = 13(a-b)^2 + 3(a-3b)^2 \quad IS_1 = 13(b-c)^2 + 3(b+c)^2 \quad IS_2 = 13(c-d)^2 + 3(3c-d)^2$$

For $a < 0$,

$$\begin{aligned}\left.\frac{\partial Y}{\partial x}\right|_{i,j}^{+} &= \frac{1}{12\Delta x}(-\Delta^+ Y_{i-2,j} + 7\Delta^+ Y_{i-1,j} + 7\Delta^+ Y_{i,j} - \Delta^+ Y_{i+1,j}) + \\ &\psi_{WENO}\left(\frac{\Delta^-\Delta^+ Y_{i+2,j}}{\Delta x}, \frac{\Delta^-\Delta^+ Y_{i+1,j}}{\Delta x}, \frac{\Delta^-\Delta^+ Y_{i,j}}{\Delta x}, \frac{\Delta^-\Delta^+ Y_{i-1,j}}{\Delta x}\right)\end{aligned}$$

Similarly, for the $\frac{\partial Y}{\partial y}$,

$$\begin{aligned}\left.\frac{\partial Y}{\partial y}\right|_{i,j}^{-} &= \frac{1}{12\Delta y}(-\Delta^+ Y_{i,j-2} + 7\Delta^+ Y_{i,j-1} + 7\Delta^+ Y_{i,j} - \Delta^+ Y_{i,j+1}) - \\ &\psi_{WENO}\left(\frac{\Delta^-\Delta^+ Y_{i,j-2}}{\Delta y}, \frac{\Delta^-\Delta^+ Y_{i,j-1}}{\Delta y}, \frac{\Delta^-\Delta^+ Y_{i,j}}{\Delta y}, \frac{\Delta^-\Delta^+ Y_{i,j+1}}{\Delta y}\right)\end{aligned}$$

with

$$\psi_{WENO}(a,b,c,d) = \frac{1}{3}\omega_0(a - 2b + c) + \frac{1}{6}\left(\omega_2 - \frac{1}{2}\right)(b - 2c + d)$$

with

$$\omega_0 = \frac{\alpha_0}{\alpha_0 + \alpha_1 + \alpha_2}, \quad \omega_2 = \frac{\alpha_2}{\alpha_0 + \alpha_1 + \alpha_2}$$

with

$$\alpha_0 = \frac{1}{(\epsilon + IS_0)^2}, \quad \alpha_1 = \frac{6}{(\epsilon + IS_1)^2}, \quad \alpha_2 = \frac{3}{(\epsilon + IS_2)^2} \qquad \epsilon = 10^{-6}$$

with

$$IS_0 = 13(a-b)^2 + 3(a-3b)^2 \quad IS_1 = 13(b-c)^2 + 3(b+c)^2 \quad IS_2 = 13(c-d)^2 + 3(3c-d)^2$$

For $b < 0$,

$$\begin{aligned}\left.\frac{\partial Y}{\partial y}\right|_{i,j}^{+} &= \frac{1}{12\Delta y}(-\Delta^+ Y_{i,j-2} + 7\Delta^+ Y_{i,j-1} + 7\Delta^+ Y_{i,j} - \Delta^+ Y_{i,j+1}) + \\ &\psi_{WENO}\left(\frac{\Delta^-\Delta^+ Y_{i,j+2}}{\Delta y}, \frac{\Delta^-\Delta^+ Y_{i,j+1}}{\Delta y}, \frac{\Delta^-\Delta^+ Y_{i,j}}{\Delta y}, \frac{\Delta^-\Delta^+ Y_{i,j-1}}{\Delta y}\right)\end{aligned}$$

6. The sub-index finite difference equation Y for viscous terms is:

$$\begin{aligned}\left.\frac{\partial^2 Y}{\partial x^2}\right|_{i,j} &= \frac{Y_{i+1,j} - 2Y_{i,j} + Y_{i-1,j}}{\Delta x^2} + O(\Delta x^2) \\ \left.\frac{\partial^2 Y}{\partial y^2}\right|_{i,j} &= \frac{Y_{i,j+1} - 2Y_{i,j} + Y_{i,j-1}}{\Delta y^2} + O(\Delta y^2)\end{aligned}$$

Use Crank Nicolson-ADI to solve these first in step one in x-direction and then in y-direction.
Step 1:

$$d_1 Y_{i+1,j}^{n+1/2} + (1+2d_1)Y_{i,j}^{n+1/2} - d_1 Y_{i-1,j}^{n+1/2} = d_2 Y_{i,j+1}^{n} + (1-2d_2)Y_{i,j}^{n} + d_2 Y_{i,j-1}^{n} + \frac{\Delta t}{2} QY(i,j)$$

Step 2:

$$d_2 Y_{i,j+1}^{n+1} + (1+2d_2)Y_{i,j}^{n+1} - d_2 Y_{i,j-1}^{n+1} = d_1 Y_{i+1,j}^{n+1/2} + (1-2d_1)Y_{i,j}^{n+1/2} + d_1 Y_{i-1,j}^{n+1/2} + \frac{\Delta t}{2} QY(i,j)$$

where,

$$d_1 = \frac{d_x}{2}, \quad d_2 = \frac{d_y}{2}, \quad d_x = \frac{\Delta t}{2ReSc\Delta x^2}, \quad d_y = \frac{\Delta t}{2ReSc\Delta y^2}$$

7. The Poisson equation is:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f(x,y)$$

in the finite difference form,

$$\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{i,j} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + O(\Delta x^2)$$
$$\left.\frac{\partial^2 \phi}{\partial y^2}\right|_{i,j} = \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta x^2} + O(\Delta y^2)$$

Therefore, the sub-index finite difference equation for Poisson equation is:

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta x^2} = f_{i,j}$$

8. The sub-index finite difference equation for the velocity divergence at the cell center or the RHS of the Poisson equation is:

$$= \left.\frac{\partial u}{\partial x}\right|_{i,j} + \left.\frac{\partial u}{\partial y}\right|_{i,j}$$
$$= \frac{u_{i+1/2,j} - u_{i-1/2,j} + v_{i,j+1/2} - v_{i,j-1/2}}{h}$$

9. The sub-index finite difference equation for u projection step is:

$$u_{i+1/2,j}^{n+1} = u_{i+1/2,j}^{*} - \Delta t \left(\frac{\phi_{i+1,j}^{n+1} - \phi_{i,j}^{n+1}}{h} \right)$$

10. The sub-index finite difference equation for v projection step is:

$$v_{i,j+1/2}^{n+1} = v_{i,j+1/2}^{*} - \Delta t \left(\frac{\phi_{i,j+1}^{n+1} - \phi_{i,j}^{n+1}}{h} \right)$$

These were the finite difference equations for all the numerical methods used to solve the problem. The finite difference equations for the boundary conditions have not been added here, since they were not the part of the grading rubric. The stability time constraint of solving these equations is mentioned here with the maximum stable time step for the Navier-Stokes equation.

Use product rule / chain rule to bring hyperbolic parts of the Navier-Stokes equation into this form:

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

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

$$\frac{\partial Y}{\partial t} + u\frac{\partial Y}{\partial x} + v\frac{\partial Y}{\partial y} = ...$$

Since all $a, c, e = u$ and b,d,f=v i.e. all three equations' hyperbolic terms have the same maximum stable time, making,

$$\Delta t_{\text{hyperbolic}} = \text{CFL} \cdot \frac{h}{(\max(|u_i|) + \max(|v_i|))}$$

For Crank-Nicolson method, the maximum stable time step ends up becoming:

$$\Delta t_{\text{parabolic}} = \text{CFL} \cdot \frac{h^2}{2}$$

Overall: $\Delta t_{max} = \min(t_{\text{hyperbolic}}, t_{\text{parabolic}})$

The cell number Reynolds number condition can usually be violated and hence, isn't mentioned here.

3 Results

3.1 Simulation Parameters

Before presenting the results, a quick summary of the solution parameters is mentioned in this section. The problem has been solved for the following solution parameters:
Mesh grid in x-direction, **M=96**, and
Mesh grid in y-direction, **N=64**.
These Mesh sizes were determined by doing Grid Convergence Index analysis. The **CFL** number for the problem has been set to **0.9**. The **Poisson** equation error of **convergence** has been set to **10e-3**.

3.2 Solution Verification

The grid convergence index analysis has been run for solution verification of the problem. The problem has been solved for three different grids of:
1. Grid 1: M=96, N=64
2. Grid 2: M=48, N=32, and
3. Grid 3: M=24, N=16

Note: had to use this small grid because M=96*2 ran for over 3 days and I had to force stop.
The $\bar{K}$ and $\bar{S}$ were calculated for each of the grids and the GCI analysis was performed.

i. GCI analysis for $\bar{K}$

Grid No.	Grid	K_bar	Observed order, p	Richardon Extrap.	GCI_12	GCI_23	Asymptotic Range of Convergence
1	M=96,N=64	98.6825	0.2733	107.0038	0.11%	0.1286%	1.0257%
2	M=48,N=32	96.947	_	_	_	_	_
3	M=24,N=16	94.8494	_	_	_	_	_

Figure 5: GCI analysis table for

- Determine order of convergence, p:

$$
\begin{aligned}
p &= \ln\left(\frac{|f_3 - f_2|}{|f_2 - f_1|}\right) / \ln(r) \\
&= \ln\left(\frac{|94.8494 - 96.947|}{|96.947 - 98.6825|}\right) / \ln(2) \\
&= \ln(1.2086)/\ln(2) \\
&= 0.2733
\end{aligned}
$$

- Richardson Extrapolation with two finest grids:

$$
\begin{aligned}
f_{h=0} &= f_1 + \frac{f_1 - f_2}{r^p - 1} \\
&= 98.6825 + \frac{98.6825 - 96.947}{2^{0.2733} - 1} \\
&= 98.6825 + \frac{1.7355}{0.20856} \\
&= 98.6825 + 8.321 \\
&= 107.0038
\end{aligned}
$$

- Grid Convergence Index 12:

$$
\begin{aligned}
GCI_{12} &= F_{sec}\frac{|\epsilon|}{r^p - 1} \\
&= 1.25\frac{|\frac{98.6825 - 96.947}{98.6825}|}{2^{0.2733} - 1} \\
&= 0.11\%
\end{aligned}
$$

$\therefore$ $\bar{K} = 107.0038$ **with the error band of 0.11 %.**

- Grid Convergence Index 23:

$$
\begin{aligned}
GCI_{23} &= F_{sec}\frac{|\epsilon|}{r^p - 1} \\
&= 1.25\frac{|\frac{96.947 - 94.8494}{96.947}|}{2^{0.2733} - 1} \\
&= 0.1296\%
\end{aligned}
$$

- Asymptotic Range of Convergence

$$
\begin{aligned}
&= \frac{GCI_{12}}{GCI_{23}} r^p \\
&= \frac{0.11}{0.1296} 2^{0.2733} \\
&= 1.0257\%
\end{aligned}
$$

$\therefore$ The result lies with the asymptotic range of convergence of 0.95% to 1.05%.

ii. GCI analysis for $\bar{R}$

- Determine order of convergence, p:

$$
\begin{aligned}
p &= \ln\left(\frac{|f_3 - f_2|}{|f_2 - f_1|}\right) / \ln(r) \\
&= \ln\left(\frac{|0.5281 - 0.54379|}{|0.54379 - 0.55691|}\right) / \ln(2) \\
&= \ln(1.196796)/\ln(2) \\
&= 0.2592
\end{aligned}
$$

Grid No.	Grid	R_bar	Observed order, p	Richardon Extrap.	GCI_12	GCI_23	Asymptotic Range of Convergence
1	M=96,N=64	0.5569	0.2592	0.62351	0.15%	0.1839%	0.9762%
2	M=48,N=32	0.54379	_	_	_	_	_
3	M=24,N=16	0.5281	_	_	_	_	_

Figure 6: GCI analysis table for

- Richardson Extrapolation with two finest grids:

$$
\begin{aligned}
f_{h=0} &= f_1 + \frac{f_1 - f_2}{r^p - 1} \\
&= 0.5569 + \frac{0.5569 - 0.54379}{2^{0.2592} - 1} \\
&= 0.5569 + \frac{0.01311}{0.19681} \\
&= 0.5569 + 0.06661 \\
&= 0.62351
\end{aligned}
$$

- Grid Convergence Index 12:

$$
\begin{aligned}
GCI_{12} &= F_{sec} \frac{|\epsilon|}{r^p - 1} \\
&= 1.25 \frac{|\frac{0.5569 - 0.54379}{0.5569}|}{2^{0.2592} - 1} \\
&= 0.15\%
\end{aligned}
$$

$\therefore$ $\bar{R} = 0.62351$ **with the error band of 0.15 %.**

- Grid Convergence Index 23:

$$
\begin{aligned}
GCI_{23} &= F_{sec} \frac{|\epsilon|}{r^p - 1} \\
&= 1.25 \frac{|\frac{0.54379 - 0.5281}{0.54379}|}{2^{0.2592} - 1} \\
&= 0.18389\%
\end{aligned}
$$

- Asymptotic Range of Convergence

$$
\begin{aligned}
&= \frac{GCI_{12}}{GCI_{23}} r^p \\
&= \frac{0.15}{0.18389} 2^{0.2592} \\
&= 0.9762\%
\end{aligned}
$$

$\therefore$ The result lies with the asymptotic range of convergence of 0.95% to 1.05%.
These GCI analysis results conclude that the mesh size of M=96 and N=64 are good enough to go forward with.

3.3 Result Figures

The contour plots of the velocity component-u at time levels of t = 4, 4.25, 4.5, 4.75, 5, 5.25, 5.5, 5.75, and 6 is as shown in Figure (7). These time points show one cycle of the rotation of the two disks starting at t=4 and ending at t=6.

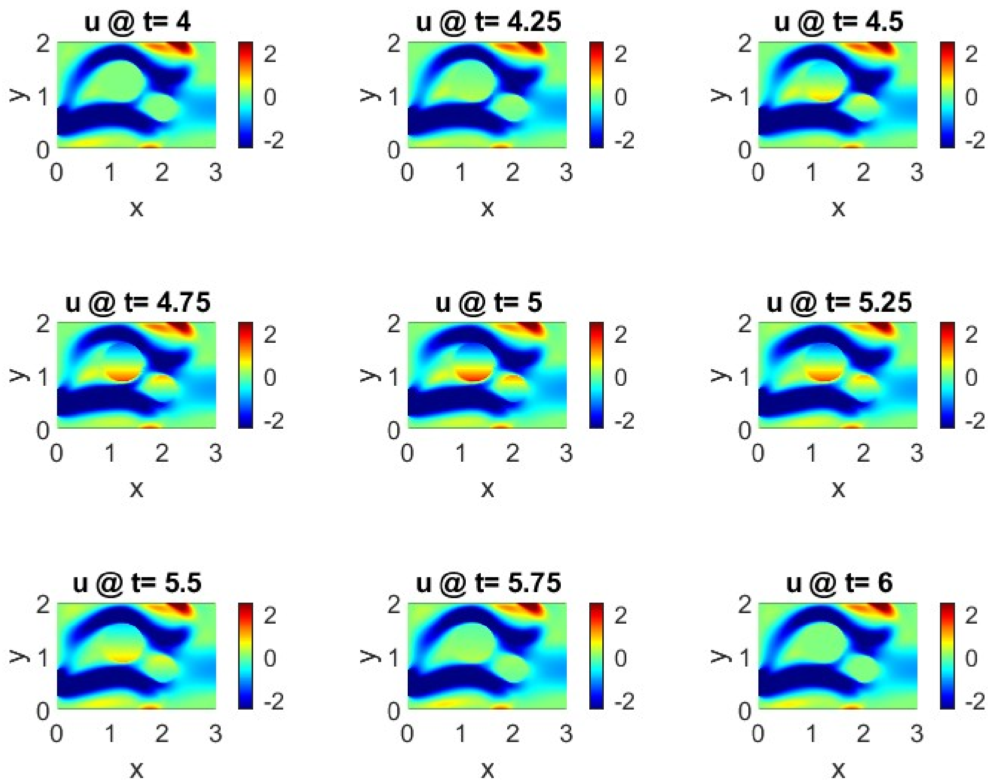

Figure 7: u-velocity at different time levels

The contour plots of the velocity component-v at time levels of t = 4, 4.25, 4.5, 4.75, 5, 5.25, 5.5, 5.75, and 6 is as shown in Figure (8). These time points show one cycle of the rotation of the two disks starting at t=4 and ending at t=6.

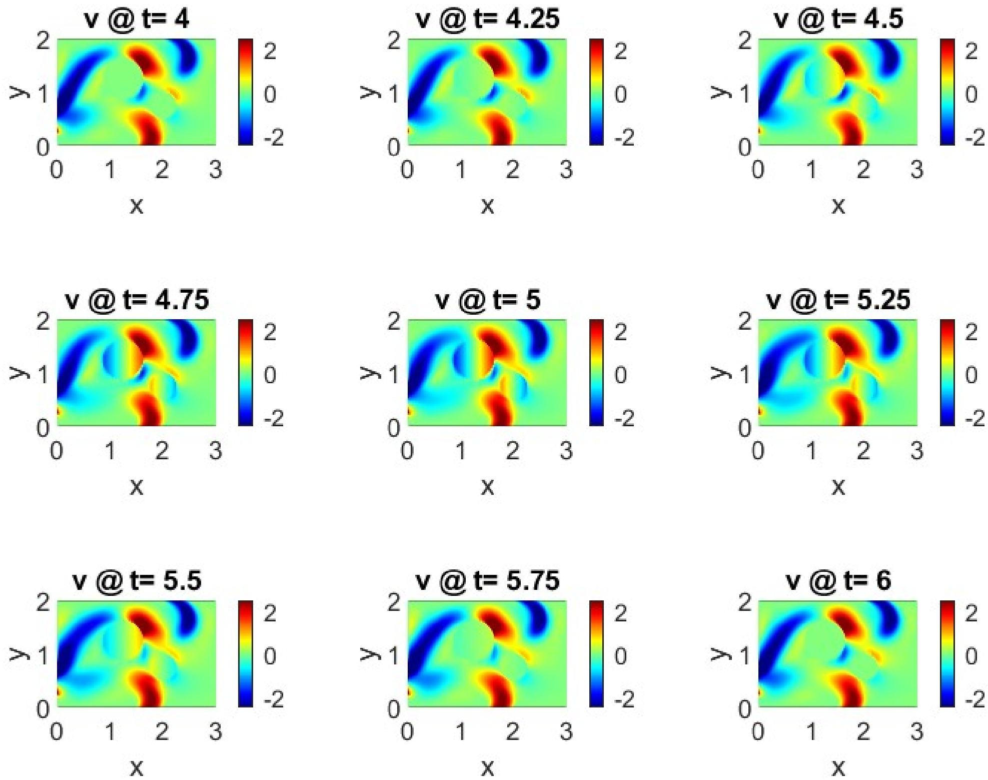

Figure 8: v-velocity at different time levels

The contour plots of the Mass Fraction-Y at time levels of t = 4, 4.25, 4.5, 4.75, 5, 5.25, 5.5, 5.75, and 6 is as shown in Figure (9). These time points show one cycle of the rotation of the two disks starting at t=4 and ending at t=6.

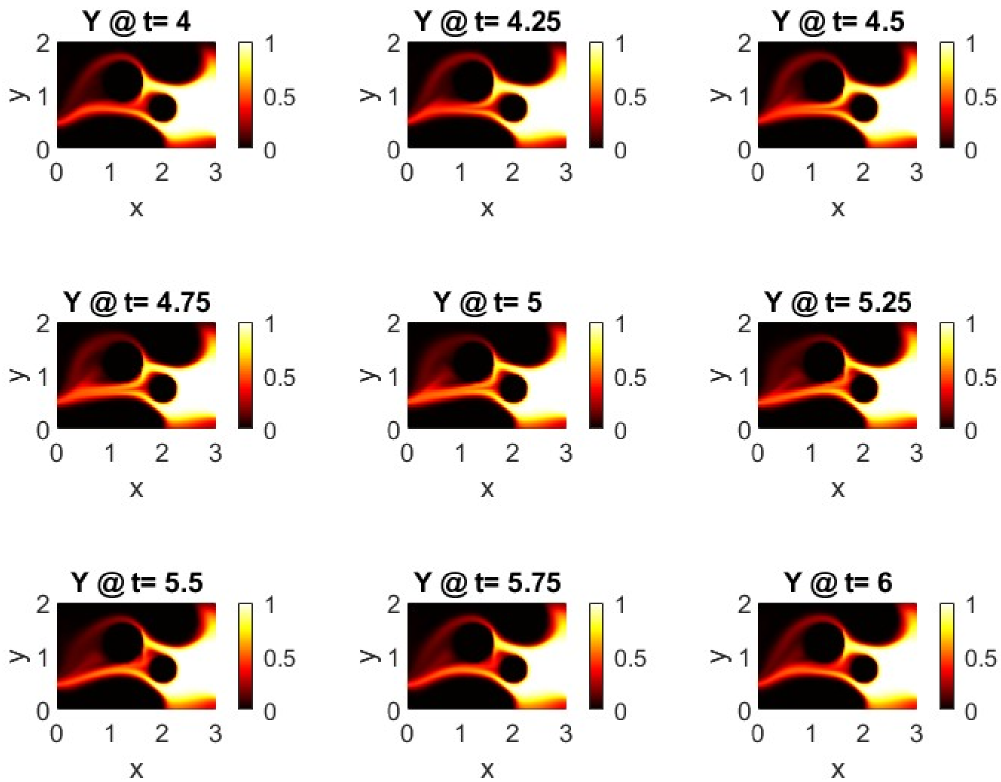

Figure 9: Mass Fraction-Y at different time levels

The plot in Figure (10) shows k(t) with time, the time goes from 0 to 10.

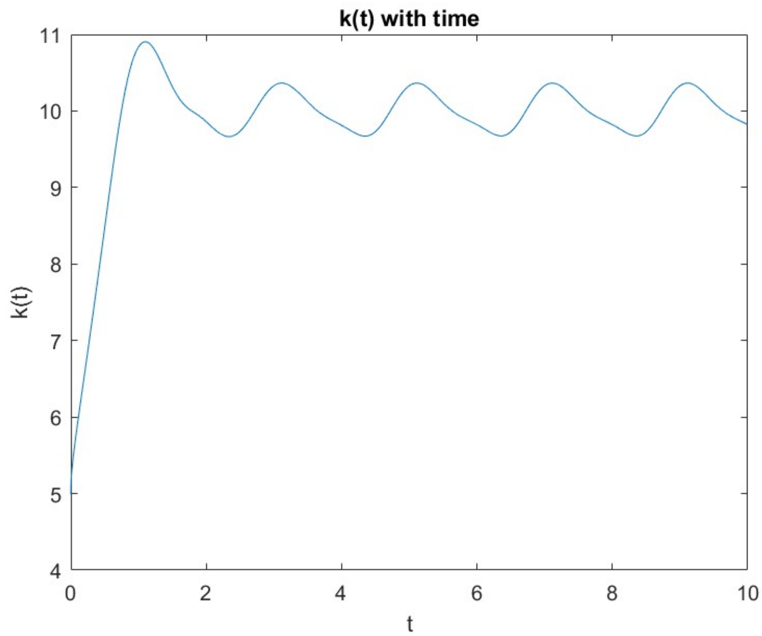

Figure 10: k(t) with time

The plot in Figure (11) shows S(t) with time, the time goes from 0 to 10.

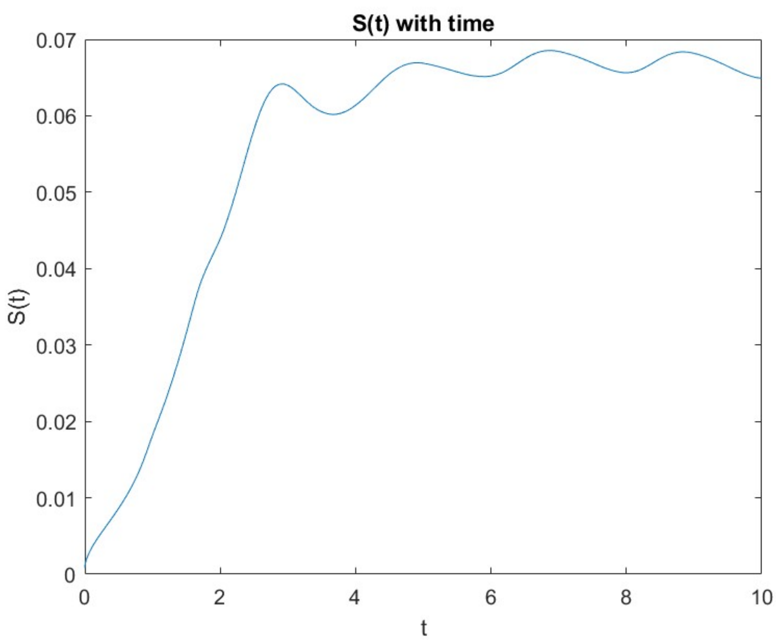

Figure 11: S(t) with time

3.4 Result Discussion

At the beginning, the chamber is still, with no chemical presence in the fluid. As we start the simulation, inlets 1, 2, and 3 open up, each introducing fluid with fully developed laminar velocity profiles and specific compositions. Looking at the u-velocity contours from t=4 to t=6, we witness a full rotation cycle of the disks. The fluid near the disks appears blue, indicating its slow movement around the disk surfaces. Around the midpoint, roughly at t=5, the flow between the disks nearly splits, with u-velocity values dropping close to zero. However, due to our low Reynolds number, we don't observe clear separations or swirling eddies. Instead, by t=6, the flow reintegrates, and the disks come to a stop. Inlet 1's fluid has the most noticeable impact, while inlets 2 and 3 contribute minimally, mainly affecting their immediate surroundings. They do not travel far enough to reach the disks. Similarly, examining the v-velocity contours reveals the consistent rotation of the disks from t=4 to t=6. Though we can't precisely identify the fluid from inlet 1, inlets 2 and 3, and the outlet are easily distinguishable within the visual representation.

At the simulation's outset, mixing starts with all parameters starting at zero. Inlet one introduces a species with a mass fraction of 1 into the chamber, while inlets two and three remain inactive, and the outlet serves as the exit point for the mass. As the rotating disks complete a full cycle between t=4 and t=6, the chemical species must navigate around them. Near the disks' surfaces, the mass fraction diminishes, reaching its lowest point, while inside the disks, it remains at zero, indicating the disks' presence. Around t=5, midway through the cycle, the flow separates, though the formation of eddies is isn't really seen due to the low Reynold's number in this project.

In our solution verification, we found that $\bar{K}$ is 107.0038 with an error band of 0.11%, and $\bar{R}$ is 0.62351 with an error band of 0.15%. Although these values fall within the expected range of 0.95% to 1.05%, they are at the lower end of the spectrum. This is evident from the order of convergence, which is approximately 0.2733 for $\bar{K}$ and 0.2592 for $\bar{R}$. In reality, the convergence order for this problem should be 2, so these values will likely approach 2 with further grid refinement. However, for our project's scope, these results are satisfactory.

4 Conclusion

In conclusion, this study employed computational modeling to investigate fluid flow and mixing dynamics within a two-dimensional rectangular mixing chamber. By numerically solving the non-dimensional continuity equation, 2D Navier-Stokes equations, and a convective/diffusive transport equation for mass fraction, we obtained detailed insights into flow patterns, species distribution, and mixing efficiency over time.

The results revealed a cyclical behavior in flow patterns, with rotating disks exerting a significant influence on fluid movement and species distribution. The introduction of chemical species through inlets led to effective mixing, with distinct flow behaviors observed near the disks and inlet/outlet regions. Performance metrics were rigorously assessed, with the average kinetic energy ($\bar{K}$) calculated to be 107.0038 with an error band of 0.11%, and the time-averaged scalar mixing parameter ($\bar{R}$) determined to be 0.62351 with an error band of 0.15%. These values fell within the expected range, indicating satisfactory convergence and solution reliability.

Solution verification through grid convergence index analysis demonstrated the robustness of the numerical approach, albeit with room for further refinement. The convergence orders for kinetic energy ($\bar{K}$) and mixing parameter ($\bar{R}$) were approximately 0.2733 and 0.2592 respectively, suggesting potential improvements with finer grids.

Overall, this study provides valuable quantitative insights into the complex dynamics of fluid flow and mixing in two-dimensional chambers. Further research avenues could explore higher Reynolds numbers, refine numerical methods, and investigate additional performance metrics for a more comprehensive understanding of mixing efficiency.

Appendices

All the functions and code used to solve this project has been attached here, except for the source term. All these files are also uploaded to the upload code in Gradescope.

```
function [u]=bc_u(u,t)

global xf yc;

[M,N]=size(u);
M=M-1;
N=N-2;

U1=3/4;
U2=2;
U3=3;
alpha1=pi;
alpha2=pi/3;
alpha3= pi/6;


%wall no-slip bc are dirichlet boundary
% applied directly in node
%not in cell centered based

%top and bottom are cell centered based
%top
u(2:M,N+2)=-u(2:M,N+1);

%bottom
u(2:M,1)=-u(2:M, 2);

%left and right are node center based
%left
u(1,2:N+1)=0;
%right
u(M+1,2:N+1)=0;


%set inlet1
%right
for j =2:N+1
        if yc(j) > 0.25 &&  yc(j) < 1.25
            u(M+1,j)= U1 * cos(alpha1)* (6*(yc(j) - 0.25) / 1)*(1-((yc(j) -
                0.25) / 1));
        end
end

%set inlet2
%bottom
for i =2:M
    if xf(i) > 1.5 && xf(i) < 2
        u(i,1)= 2*U2 * cos(alpha2)* (6*(xf(i) - 1.5) / 0.5)*(1-((xf(i) - 1.5)
            / 0.5))-u(i,2);
```

```
    end
end

%set inlet3
%top
for i =2:M
    if xf(i) > 2 && xf(i) < 2.5
        u(i,N+2)= 2*U3 * cos(alpha3)* (6*(xf(i) - 2) / 0.5)*(1-((xf(i) - 2) /
            0.5))-u(i,N+1);;
    end
end

%set outlet
%left boundary %IS NEUMANN %check %CALCULATE  %second order node based, use
    that from last hoemwork
for j =2:N+1
        if yc(j) > 0.25 &&  yc(j) < 0.75
            u(1,j)=    -(1/3)*u(3,j)+(4/3)*u(2,j);
        end
end

end
```

```
function [v]=bc_v(v,t)

global xc yf;

[M,N]=size(v);
M=M-2;
N=N-1;

U1=3/4;
U2=2;
U3=3;
alpha1=pi;
alpha2=pi/3;
alpha3= -pi/6;


%wall no-slip bc are dirichlet boundary
% applied directly in node
%not in cell centered based

%top and bottom are node based
%top
v(2:M+1,N+1)=0;

%bottom
v(2:M+1,1)=0;

%left and right are cell centered based
%left
v(1,2:N)=-v(2,2:N);
%right
```

```
v(M+2,2:N)=-v(M+1,2:N);


%set inlet1
%right
for j =2:N
        if yf(j) > 0.25 &&  yf(j) < 1.25
            v(M+2,j)= 2*U1 * sin(alpha1)* (6*(yf(j) - 0.25) / 1)*(1-((yf(j) -
                0.25) / 1))-v(M+1,j);
        end
end

%set inlet2
%bottom
for i =2:M+1
    if xc(i) > 1.5 && xc(i) < 2
        v(i,1)= U2 * sin(alpha2)* (6*(xc(i) - 1.5) / 0.5)*(1-((xc(i) - 1.5) /
            0.5));
    end
end

%set inlet3
%top
for i =2:M+1
    if xc(i) > 2 && xc(i) < 2.5
        v(i,N+1)= U3 * sin(alpha3)* (6*(xc(i) - 2) / 0.5)*(1-((xc(i) - 2) /
            0.5));
    end
end

%set outlet
%left boundary
for j =2:N
        if (yf(j) >= 0.25) &&  (yf(j) <= 0.75)
            v(1,j)= v(2,j);
        end
end

end
```

```
function [Y]=bc_Y3(Y,t)

global xc3 yc3;

[M,N]=size(Y);
M=M-6;
N=N-6;

%top and bottom are cell-centered based
%top
Y(4:M+3,N+6)=Y(4:M+3,N+1);
Y(4:M+3,N+5)=Y(4:M+3,N+2);
Y(4:M+3,N+4)=Y(4:M+3,N+3);

```

```
%bottom
Y(4:M+3,1)=Y(4:M+3,6);
Y(4:M+3,2)=Y(4:M+3,5);
Y(4:M+3,3)=Y(4:M+3,4);


%left and right are cell centered based
%left
Y(1,4:N+3)=Y(6,4:N+3);
Y(2,4:N+3)=Y(5,4:N+3);
Y(3,4:N+3)=Y(4,4:N+3);

%right
Y(M+6,4:N+3)=Y(M+1,4:N+3);
Y(M+5,4:N+3)=Y(M+2,4:N+3);
Y(M+4,4:N+3)=Y(M+3,4:N+3);


%set inlet2
%bottom
for i =4:M+3
    if xc3(i) > 1.5 && xc3(i) < 2
        Y(i,1)= 2*0-Y(i,6);
        Y(i,2)= 2*0-Y(i,5);
        Y(i,3)= 2*0-Y(i,4);
    end
end

%set inlet3
%top
for i =4:M+3
    if xc3(i) > 2 && xc3(i) < 2.5
        Y(i,N+6)= 2*0-Y(i,N+1);
        Y(i,N+5)= 2*0-Y(i,N+2);
        Y(i,N+4)= 2*0-Y(i,N+3);
    end
end


%set inlet1
%right
for j =4:N+3
        if yc3(j) > 0.25 &&  yc3(j) < 1.25
            Y(M+6,j)= 2*1-Y(M+1,j);
            Y(M+5,j)= 2*1-Y(M+2,j);
            Y(M+4,j)= 2*1-Y(M+3,j);
        end
end


%set outlet
%left boundary
for j =4:N+3
        if (yc3(j) >= 0.25) &&  (yc3(j) <= 0.75)
```

```
                Y(1,j)= Y(6,j);
                Y(2,j)= Y(5,j);
                Y(3,j)= Y(4,j);
            end
end

end
```

```
function [a,b,c,d] = bcCN1_u(a,b,c,d,t)

global yc;

[M,N]=size(a);
M=M-1;
N=N-2;


U1=3/4;
alpha1=pi;


%left and right are node-based

%left boundary: all wall points are same
%left boundary outlet changes:
%set outlet values: only b and c changed
for j =2:N+1
        if yc(j) >= 0.25 &&  yc(j) <= 0.75
            b(2,j)=4/3*a(2,j)+b(2,j);
            c(2,j)=-a(2,j)/3+c(2,j);
        end
end
a(2,2:N+1)= 0;



%right boundary
%inlet_1
for j =2:N+1
        if yc(j) > 0.25 &&  yc(j) < 1.25

          d(M,j)= d(M,j)-c(M,j)*U1 * cos(alpha1)* (6*(yc(j) - 0.25) / 1)*(1-((
              yc(j) - 0.25) / 1));
      end
end

%right wall
c(M,2:N+1)=0;

end
```

```
function [a,b,c,d] = bcCN1_v(a,b,c,d,t)

global yf;
```

```

[M,N]=size(a);
M=M-2;
N=N-1;
U1=3/4;
alpha1=pi;

%left and right are cell center-based

%left boundary: wall


%left boundary outlet changes:
for j =2:N
        if (yf(j) >= 0.25) && (yf(j) <= 0.75)
            b(2,j)=b(2,j)+a(2,j);
        else
            b(2,j)=b(2,j)-a(2,j);
        end
end

%right boundary: wall points
b(M+1,2:N)=b(M+1,2:N)-c(M+1,2:N);

%right boundary inlet changes:
for j=2:N
 if yf(j) > 0.25 && yf(j) < 1.25
    d(M+1,j)=d(M+1,j)-2*c(M+1,j)*U1 * sin(alpha1)* (6*(yf(j) - 0.25) / 1)
        *(1-((yf(j) - 0.25) / 1));
 end
end


end
```

```
function [a,b,c,d] = bcCN1_Y3(a,b,c,d,t)

global yc3;

[M,N]=size(a);
M=M-6;
N=N-6;


%left and right are cell centered based
%left

for j =4:N+3
    b(4,j)=a(4,j)+b(4,j);
end

%right
for j =4:N+3
        if yc3(j) > 0.25 && yc3(j) < 1.25
```

```
            b(M+3,j)=b(M+3,j)-c(M+3,j);
            d(M+3,j)=d(M+3,j)-2*c(M+3,j);
        else
            b(M+3,j)=b(M+3,j)+c(M+3,j);
        end
end

end
```

```
function [a,b,c,d] = bcCN2_u(a,b,c,d,t)

global xf;

[M,N]=size(a);
M=M-1;
N=N-2;

U1=3/4;
U2=2;
U3=3;
alpha1=pi;
alpha2=pi/3;
alpha3= pi/6;

%top and bottom are cell center-based

%top boundary
%top
b(2:M,N+1)=b(2:M,N+1)-c(2:M,N+1);
%top boundary inlet changes:
%set inlet values: only b and d changed
%inlet3
for i =2:M
    if xf(i) > 2 && xf(i) < 2.5
%        b(i,N+1)= b(i,N+1)-c(i,N+1);
        d(i,N+1)=d(i,N+1)-2*c(i,N+1)*U3 * cos(alpha3)* (6*(xf(i) - 2) / 0.5)
            *(1-((xf(i) - 2) / 0.5));
    end
end


%bottom boundary
%bottom wall
b(2:M,2)= b(2:M,2)-a(2:M,2);

%bottom boundary inlet changes:
%inlet2
for i =2:M
    if xf(i) > 1.5 && xf(i) < 2
%         b(i,2)= b(i,2)-a(i,2);
         d(i,2)= d(i,2)-2*a(i,2)*U2 * cos(alpha2)* (6*(xf(i) - 1.5) / 0.5)
             *(1-((xf(i) - 1.5) / 0.5));
    end
end
```

```
end
```

```
function [a,b,c,d] = bcCN2_v(a,b,c,d,t)

global xc;

[M,N]=size(a);
M=M-2;
N=N-1;
U2=2;
U3=3;
alpha2=pi/3;
alpha3=-pi/6;


%top and bottom are node-based
%top wall
%same

%top inlet: node based dirichlet
for i =2:M+1
    if xc(i) > 2 && xc(i) < 2.5
        d(i,N)= d(i,N)-c(i,N)*U3 * sin(alpha3)* (6*(xc(i) - 2) / 0.5)*(1-((xc(i) - 2) / 0.5));
    end
end

%bottom
%same

%bottom inlet: node based dirichlet
for i =2:M+1
    if xc(i) > 1.5 && xc(i) < 2
        d(i,2)= d(i,2)-a(i,2)*U2 * sin(alpha2)* (6*(xc(i) - 1.5) / 0.5)*(1-((xc(i) - 1.5) / 0.5));
    end
end




end
```

```
function [a,b,c,d] = bcCN2_Y3(a,b,c,d,t)

global xc3;

[M,N]=size(a);
M=M-6;
N=N-6;
```

```
%top and bottom are cell centered based
%top

for i =4:M+3
    if xc3(i) > 2 && xc3(i) < 2.5
       b(i,N+3)=b(i,N+3)-c(i,N+3);
    else
        b(i,N+3)=b(i,N+3)+c(i,N+3);
    end
end

%bottom
for i =4:M+3
    if xc3(i) > 1.5 && xc3(i) < 2
       b(i,4)=b(i,4)-a(i,4);
    else
       b(i,4)=b(i,4)+a(i,4);
    end
end

end
```

```
function [u]=bcGhost_u(u,t)

global xf;

[M,N]=size(u);
M=M-1;
N=N-2;



U2=2;
U3=3;

alpha2=pi/3;
alpha3= pi/6;


%wall no-slip bc are dirichlet boundary
% applied directly in node
%not in cell centered based

%top and bottom are cell centered based
%top
u(1:M+1,N+2)=-u(1:M+1,N+1);



%set inlet3
%top
for i =1:M+1
    if xf(i) > 2 && xf(i) < 2.5
        u(i,N+2)= 2*U3 * cos(alpha3)* (6*(xf(i) - 2) / 0.5)*(1-((xf(i) - 2) /
```

```
            0.5))-u(i,N+1);;
    end
end


%bottom
u(1:M+1,1)=-u(1:M+1,2);

%set inlet2
%bottom
for i =1:M+1
    if xf(i) > 1.5 && xf(i) < 2
        u(i,1)= 2*U2 * cos(alpha2)* (6*(xf(i) - 1.5) / 0.5)*(1-((xf(i) - 1.5)
            / 0.5))-u(i,2);
    end
end

end
```

```
function [v]=bcGhost_v(v,t)

global yf;

[M,N]=size(v);
M=M-2;
N=N-1;

U1=3/4;
U2=2;
U3=3;
alpha1=pi;
alpha2=pi/3;
alpha3= -pi/6;


%left and right are cell centered based
%left
v(1,1:N+1)=-v(2,1:N+1);
%right
v(M+2,1:N+1)=-v(M+1,1:N+1);


%set outlet
%left boundary
for j =1:N+1
        if (yf(j) >= 0.25) &&  (yf(j) <= 0.75)
            v(1,j)= v(2,j);
        end
end

%set inlet1
%right
for j =1:N+1
        if yf(j) > 0.25 &&  yf(j) < 1.25
```

```
            v(M+2,j)= 2*U1 * sin(alpha1)* (6*(yf(j) - 0.25) / 1)*(1-((yf(j) -
                0.25) / 1))-v(M+1,j);
        end
end


end
```

```
function phi= bcGS(phi)
    [M,N] = size(phi);
    M=M-2;
    N=N-2;

    %Neumann boundary conditions
    phi(1, 2:N+1) = phi(2, 2:N+1);        %left boundary
    phi(M+2, 2:N+1) = phi(M+1, 2:N+1);    %right boundary
    phi(2:M+1, 1) = phi(2:M+1, 2);        %bottom boundary
    phi(2:M+1, N+2)=phi(2:M+1, N+1);      %top boundary

end
```

```
%let j/N be as it is

function f = calc_adYdx_WENO_2D(Y, a)
    global h;

    [M, N] = size(Y);
    M = M - 6;
    N = N - 6;
    f = zeros(M + 6, N + 6);

    % Define local functions for dp and dmdp
    function dp = dpf(Y, i, j)
        dp = Y(i+1, j) - Y(i, j);
    end

    function dmdp = dmdpf(Y, i, j)
        dmdp = Y(i+1, j) - 2 * Y(i, j) + Y(i-1, j);
    end

    % Implement WENO-5 method
    for j = 4:N+3
        for i = 4:M+3
                if a(i, j) >= 0
                    f(i, j) = (1/(12 * h)) * (-dpf(Y, i-2, j) + 7 * dpf(Y, i
                        -1, j) + 7 * dpf(Y, i, j) - dpf(Y, i+1, j)) - psiWENO(
                        dmdpf(Y, i-2, j)/h, dmdpf(Y, i-1, j)/h, dmdpf(Y, i, j)
                        /h, dmdpf(Y, i+1, j)/h);
                else
                    f(i, j) = (1/(12 * h)) * (-dpf(Y, i-2, j) + 7 * dpf(Y, i
                        -1, j) + 7 * dpf(Y, i, j) - dpf(Y, i+1, j)) + psiWENO(
                        dmdpf(Y, i+2, j)/h, dmdpf(Y, i+1, j)/h, dmdpf(Y, i, j)
                        /h, dmdpf(Y, i-1, j)/h);
                end
```

```
                f(i, j) = a(i, j) * f(i, j);
            end
        end
    end
```

```
function f = calc_bdYdy_WENO_2D(Y, b)
    global h dp dmdp;

    [M, N] = size(Y);
    M = M - 6;
    N = N - 6;
    f = zeros(M + 6, N + 6);

    % Define local functions for dp and dmdp
    function dp = dpf(Y, i, j)
        dp = Y(i, j+1) - Y(i, j);
    end

    function dmdp = dmdpf(Y, i, j)
        dmdp = Y(i, j+1) - 2 * Y(i, j) + Y(i, j-1);
    end

    % Implement WENO-5 method
    for j = 4:N+3
        for i = 4:M+3
            if b(i, j) >= 0
                f(i, j) = (1/(12 * h)) * (-dpf(Y, i, j-2) + 7 * dpf(Y, i,
                    j-1) + 7 * dpf(Y, i, j) - dpf(Y, i, j+1)) - psiWENO(
                    dmdpf(Y, i, j-2)/h, dmdpf(Y, i, j-1)/h, dmdpf(Y, i, j)
                    /h, dmdpf(Y, i, j+1)/h);
            else
                f(i, j) = (1/(12 * h)) * (-dpf(Y, i, j-2) + 7 * dpf(Y, i,
                    j-1) + 7 * dpf(Y, i, j) - dpf(Y, i, j+1)) + psiWENO(
                    dmdpf(Y, i, j+2)/h, dmdpf(Y, i, j+1)/h, dmdpf(Y, i, j)
                    /h, dmdpf(Y, i, j-1)/h);
            end
            f(i, j) = b(i, j) * f(i, j);
        end
    end
end
```

```
function divV = calcDivV(u,v)
    global h;

    % Initialize du with zeros
    [M,N]=size(u);
    M=M-1;
    N=N-2;
    du = zeros(M+2,N+2);
    dv = zeros(M+2,N+2);
    divV=zeros(M+2,N+2);

    % Calculate du, dv and divV in the interior cells
    for j =2:N+1
```

```
        for i = 2:M+1
            du(i, j) = (u(i, j) - u(i-1, j)) / (h);
            dv(i, j) = (v(i, j) - v(i, j-1)) / (h);
            divV(i,j)= du(i,j)+dv(i,j);
        end
    end

end
```

```
function [dt, outputFlag] = calcDt(t, outputTime, u,v)
    global h;
    global CFL;

    % Initialize output flag
    outputFlag = 0;

    %Hyperbolic: from end of module 7 LOOK AT THE MODULESSS DONT GUESS!!!
    %ends of becoming the same for both 1-2
    dt_hyper=CFL * h/ (max(abs(u),[],"all")+ max(abs(v),[],"all"));

    %parabolic
    %assumin nu to be 1
    dt_para=CFL *h^2;    %this is for crank nicolson you had it for FTCS!!!

    dt=min(dt_hyper,dt_para);

    % Check if the next time step exceeds the output time
        if (t < outputTime) && (t + dt >= outputTime)
            % Adjust time step for last step to match output time
            dt = outputTime - t;

            % Set output flag to indicate reaching output time
            outputFlag = 1;
        end

end
```

```
function kx = calck(u,v)
    global h;

    % Determine the size of the velocity array
    [M, N] = size(u);
    M=M-1;
    N=N-2;


    %dx=Lx/h;
    %dy=Ly/h;
    % Initialize kx
    kx = 0;

    % Calculate kx using composite 2D midpoint integration rule
    for j=2:N+1
        for i = 2:M+1
```

```
            % Calculate velocity at cell center (staggered)
            u_center=0.5 * (u(i,j) + u(i-1,j));
            v_center=0.5 * (v(i,j) + v(i,j-1));
            % Calculate kinetic energy for this time step using midpoint rule
            kx = kx+0.5 * (u_center.^2+v_center.^2) * h * h;
        end
    end

end
```

```
function S= calcS3(Y)
    global h Lx Ly;

    % Determine the size of the velocity array
    [M, N] = size(Y);
    M=M-6;
    N=N-6;


    %dx=Lx/h;
    %dy=Ly/h;
    % Initialize S
    S = 0;

    % Calculate S using composite 2D midpoint integration rule
    for j=4:N+3
        for i = 4:M+3
            %it is all already cell centered values
            % Calculate kinetic energy for this time step using midpoint rule
            S = S+(Y(i,j)*(1-Y(i,j)) * h * h);
        end
    end
    S=S/(Lx*Ly);

end
```

```
function [u, v]= correctOutlet(u,v)
    global h yc ucorr;

    L_out=0.5;

    [M1,N1]=size(u);
    M1=M1-1;
    N1=N1-2;

    [M2,N2]=size(v);
    M2=M2-2;
    N2=N2-1;

    u_asterisk1=0;
    u_asterisk2=0;
    v_asterisk1=0;
    v_asterisk2=0;
    for j =2:N1+1
```

```
        u_asterisk1=u_asterisk1+u(1,j)*h;
        u_asterisk2=u_asterisk2+u(M1+1,j)*h;
    end

    for i =2:M2+1
        v_asterisk1=v_asterisk1+v(i,1)*h;
        v_asterisk2=v_asterisk2+v(i,N2+1)*h;
    end
    q_dot_asterisk=u_asterisk1-u_asterisk2+v_asterisk1-v_asterisk2;
    q_corr=q_dot_asterisk;
    ucorr=q_corr/L_out;


   for j = 2:N1+1
        if yc(j)>=0.25 && yc(j) <=0.75
              u(1,j)=u(1,j)-ucorr;
        end
   end



end
```

```
clear all;
%grid is varied


% Define global parameters

global Re Sc t h CFL xf yc xc yf xc3 yc3 Lx Ly;


% Parameters

Lx = 3;
Ly = 2;
M = 48;      %the grid is varied
N= 32;       %the grid is varied
Re = 100;
Sc = 2;
h = Lx / M; %is the same for both x and y
CFL = 0.9;
t = 0;


% Initialize grid

xf = linspace(0, Lx, M + 1)';
yc = linspace(0-h/2,Ly+h/2, N+2)';

xc = linspace(0-h/2,Lx+h/2,M+2)';
yf = linspace(0,2,N+1)';

xc3 = linspace(0-2.5*h,Lx+2.5*h,M+6)';
```

```
yc3 = linspace(0-2.5*h,Ly+2.5*h,N+6)';



% Initialize solution arrays
phi=ones(M+2,N+2);
u_ini = zeros(M+1, N+2);
Hu= zeros(M+1, N+2);

v_ini = zeros(M+2, N+1);
Hv= zeros(M+2, N+1);

Y_ini = zeros(M+6, N+6);

%BCs
phi=bcGS(phi);
u = bc_u(u_ini, t);
v = bc_v(v_ini, t);
Y = bc_Y3(Y_ini, t);

%Ghost BCs
u= bcGhost_u(u,t);
v= bcGhost_v(v,t);

%Correct Outlet
[u, v]= correctOutlet(u,v);

%Poisson Equation
f=calcDivV(u,v);
phi=myPoisson(phi,f,h,10,10e-3);

%Project velocities
[u,v] = projectV(u,v, phi, 1);

%Ghost BCs
u= bcGhost_u(u,t);
v= bcGhost_v(v,t);

counter=1;

%vid=VideoWriter('video','MPEG-4');
%open(vid);
time_points = 0:0.0049505:10;
for i = 1:length(time_points)
    t = time_points(i);
    outputTime = t + 0.0049505;

   while (t < outputTime)
   [dt, outputFlag] = calcDt(t, outputTime,u,v);


  %for the next loop/for the Y for first loop

    u1=u;
```

```
    v1=v;

    [Hu, Hv]=hyperbolic_uv_2D(u,v);


    %for the first loop only

    if t==0
    Hu1=Hu;
    Hv1=Hv;
    end


    Hu_loop=1.5*Hu-0.5*Hu1;
    Hv_loop=1.5*Hv-0.5*Hv1;


    %heat source
    [Qu, Qv, QY] = calcSourceIBFinal(u, v, Y, t, dt);

     %added H to the first ADI step Q
     Qu=Qu+Hu_loop;
     Qv=Qv+Hv_loop;


     %parabolic part
     u_CN1 = parabolic_CN1_2D_u(u,Qu,dt);
     v_CN1 = parabolic_CN1_2D_v(v,Qv,dt);

     %for the second ADI
     %calculated at the right time
     [Qu, Qv, QY] = calcSourceIBFinal(u_CN1, v_CN1, Y, t+dt/2, dt);

     %Added H values to the new Qs again
     Qu=Qu+Hu_loop;
     Qv=Qv+Hv_loop;

     u = parabolic_CN2_2D_u(u_CN1,Qu,dt);
     v = parabolic_CN2_2D_v(v_CN1,Qv,dt);


     %hyperbolic and parabolic part for Y

     [Qu, Qv, QY] = calcSourceIBFinal(u1, v1, Y, t, dt); %repeated for u and v but isnt used
     HY=1.5*hyperbolic_Y_WENO_2D(Y,u1,v1,u,v,dt)-0.5*hyperbolic_Y_WENO_2D(Y,u1,v1,u,v,dt);
     QY=QY+HY;
     Y = parabolic_CN1_2D_Y3(Y,QY,dt);
     [Qu, Qv, QY] = calcSourceIBFinal(u_CN1, v_CN1, Y, t+dt/2, dt); %repeated for u and v but isnt used
     QY=QY+HY;
     Y = parabolic_CN2_2D_Y3(Y,QY,dt);

```

```

    %for the next loop making current hyperbolic t to t-1
    Hu1=Hu;
    Hv1=Hv;

    %time-stepping
    t=t+dt;

    %Ghost BCs
    u= bcGhost_u(u,t);
    v= bcGhost_v(v,t);

     %Correct Outlet
     [u, v]= correctOutlet(u,v);

     %HAVENT CALCULATED func, rhs of poisson equation
     %Poisson Equation
      f=calcDivV(u,v);
      phi=myPoisson(phi,f,h,10,10e-3);

      %Project velocities
      [u,v] = projectV(u,v, phi, 1);

      %Ghost BCs
      u= bcGhost_u(u,t);
      v= bcGhost_v(v,t);

end

%calc k

k(i) = calck(u,v);

%calc S

S(i) = calcS3(Y);

if round(t,3)==4 || round(t,3)==4.252 || round(t,3)==4.5 || round(t,3)==4.752 || round(t,3)==5 || round(t,3)==5.252 || round(t,3)==5.50 || round(t,3)==5.752 || round(t,3)==6
examFig1 = figure(1);
subplot(3,3,counter);
pcolor(xf, yc, u');
shading interp;
colormap(jet);
colorbar;
xlim([0 Lx]);
ylim([0 Ly]);
clim([-2.5, 2.5]);
title(['u @ t= ', num2str(round(t,2))]);
xlabel('x');
ylabel('y');
% Adjusting aspect ratio to make the figure rectangular
pbaspect([3 2 1]); % Set the aspect ratio to 3:2 (width:height)
```

```

  examFig2 = figure(2);
  subplot(3,3,counter);
  pcolor(xc, yf, v');
  shading interp;
  colormap(jet);
  colorbar;
  xlim([0 Lx]);
  ylim([0 Ly]);
  clim([-2.5, 2.5]);
  title(['v @ t= ', num2str(round(t,2))]);
  xlabel('x');
  ylabel('y');
 % Adjusting aspect ratio to make the figure rectangular
  pbaspect([3 2 1]); % Set the aspect ratio to 3:2 (width:height)


  examFig3 = figure(3);
  subplot(3,3,counter);
  pcolor(xc3, yc3, Y');
  shading interp;
  colormap(hot);
  colorbar;
  xlim([0 Lx]);
  ylim([0 Ly]);
  clim([0, 1]);
  title(['Y @ t= ', num2str(round(t,2))]);
  xlabel('x');
  ylabel('y');
  counter=counter+1;
   % Adjusting aspect ratio to make the figure rectangular
  pbaspect([3 2 1]); % Set the aspect ratio to 3:2 (width:height)

end

%FOR PLOTTING VIDEO BONUS PROBLEM varied for each variable
% figure(6)
% pcolor(xf, yc, u');
%  shading interp;
%  colormap(jet);
%  colorbar;
%  xlim([0 Lx]);
%  ylim([0 Ly]);
%  clim([-2.5, 2.5]);
%  title(['u @ t= ', num2str(round(t,2))]);
%  xlabel('x');
%  ylabel('y');
%  % Adjusting aspect ratio to make the figure rectangular
%  pbaspect([3 2 1]); % Set the aspect ratio to 3:2 (width:height)

   % F=getframe(figure(6));
    %writeVideo(vid, F);
end
%close(vid);
```

```
%plotting k
examFig4 = figure(4);
plot(time_points,k);
title('k(t) with time')
xlabel('t');
ylabel('k(t)');


%plotting S
examFig5 = figure(5);
plot(time_points,S);
title('S(t) with time')
xlabel('t');
ylabel('S(t)');


k_bar= myTrapezoidal(time_points,k);
r_bar= myTrapezoidal(time_points,S);
```

```
function [Hu Hv]= hyperbolic_uv_2D(u,v)

    global h;
    [M,N]=size(u);
    M=M-1;
    N=N-2;
    Hu=zeros(M+1,N+2);

    %for u
    for j=2:N+1
        for i=2:M
            u_ij=0.5*(u(i,j)+u(i-1,j)); %chnaged
            u_ihalfjhalf=0.5*(u(i,j)+u(i,j+1));
            v_ihalfjhalf=0.5*(v(i,j)+v(i+1,j));


            u_ihalfjhalfneg=0.5*(u(i,j)+u(i,j-1));
            v_ihalfjhalfneg=0.5*(v(i,j-1)+v(i+1,j-1));


            duu_dx(i,j)=((0.5*(u(i+1,j)+u(i,j))).^2-u_ij.^2)/h;
            duv_dy(i,j)=(u_ihalfjhalf*v_ihalfjhalf-u_ihalfjhalfneg*
                v_ihalfjhalfneg)/h;

            Hu(i,j)=-duu_dx(i,j)-duv_dy(i,j); %MISSED THE TAKE TO RHS SIDE

        end
    end

    %for v
    [M,N]=size(v);
    M=M-2;
    N=N-1;
    Hv=zeros(M+2,N+1);
```

```
    for j=2:N
        for i=2:M+1
            v_ij(i,j)=0.5*(v(i,j)+v(i,j-1));

            u_ihalfjhalf_V(i,j)=0.5*(u(i,j)+u(i,j+1));
            v_ihalfjhalf_V(i,j)= 0.5*(v(i,j)+v(i+1,j));

            u_ihalfjhalfneg_V(i,j)=0.5*(u(i-1,j)+u(i-1,j+1));
            v_ihalfjhalfneg_V(i,j)=0.5*(v(i,j)+v(i-1,j));


            dvv_dy(i,j)=((0.5*(v(i,j+1)+v(i,j))).^2-v_ij(i,j).^2)/h;
            duv_dx(i,j)=(u_ihalfjhalf_V(i,j)*v_ihalfjhalf_V(i,j)-
                u_ihalfjhalfneg_V(i,j)*v_ihalfjhalfneg_V(i,j))/h;

            Hv(i,j)=-dvv_dy(i,j)-duv_dx(i,j);

        end
    end


end
```

```
%ONLY CHECKING FOR a0 YESSSSSSS!!!!!


function [HY] =hyperbolic_Y_WENO_2D(Y,u,v,u1,v1,dt)
    global t a0 a1 a2 b0 b1 b2 Y1 Y2 Ys;

    [M,N]=size(Y);
    M=M-6;
    N=N-6;
   % Determine the size of the velocity array
    [M1, N1] = size(u);
    M1=M1-1;
    N1=N1-2;

    for j=2:N1+1
        for i = 2:M1+1
            % Calculate velocity at cell center (staggered)
            u_center(i,j)=0.5 * (u(i,j) + u(i-1,j));
            v_center(i,j)=0.5 * (v(i,j) + v(i,j-1));

            u1_center(i,j)=0.5 * (u1(i,j) + u1(i-1,j));
            v1_center(i,j)=0.5 * (v1(i,j) + v1(i,j-1));
        end
    end

    a0=[zeros(M+1,2) u_center zeros(M+1,3)];
    a0=[zeros(2,N+6); a0; zeros(3,N+6)];
    a1=[zeros(M+1,2) u1_center zeros(M+1,3)];
    a1=[zeros(2,N+6); a1; zeros(3,N+6)];
    a2=(a0+a1)*0.5;
```

```

    b0=[zeros(M+1,2) v_center zeros(M+1,3)];
    b0=[zeros(2,N+6); b0; zeros(3,N+6)];
    b1=[zeros(M+1,2) v1_center zeros(M+1,3)];
    b1=[zeros(2,N+6); b1; zeros(3,N+6)];
    b2=(b0+b1)*0.5;



    Y1=zeros(M+6,N+6);
    Y2=zeros(M+6,N+6);
    Ys=zeros(M+6,N+6);

    adydx0=calc_adYdx_WENO_2D(Y,a0);
    bdydx0=calc_bdYdy_WENO_2D(Y,b0);


    % step 1
    for j =4:N+3
        for i =4:M+3
         Y1(i,j)=Y(i,j)-dt*(adydx0(i,j)+bdydx0(i,j));
        end
    end
    Y1=bc_Y3(Y1,t+dt);

    adydx1=calc_adYdx_WENO_2D(Y1,a1);
    bdydx1=calc_bdYdy_WENO_2D(Y1,b1);
    %step 2
    for j=4:N+3
        for i =4:M+3
            Y2(i,j)=Y1(i,j)+(3/4)*dt*(adydx0(i,j)+bdydx0(i,j))+-(1/4)*dt*(adydx1(i,j)+bdydx1(i,j));
        end
    end
    Y2=bc_Y3(Y2,t+dt/2);

    adydx2=calc_adYdx_WENO_2D(Y2,a2);
    bdydx2=calc_bdYdy_WENO_2D(Y2,b2);
    %step 3
    for j=4:N+3
        for i =4:M+3
         Ys(i,j)=Y2(i,j)+(1/12)*dt*(adydx0(i,j)+bdydx0(i,j))+(1/12)*dt*(adydx1(i,j)+bdydx1(i,j))-(2/3)*dt*(adydx2(i,j)+bdydx2(i,j));
        end
    end
    Ys=bc_Y3(Ys,t+dt);

    HY=(Ys-Y)/dt;

end
```

```
function phi=myGaussSeidel(phi,f,h,niter)
[M,N] = size(phi);
M=M-2;
```

```
N=N-2;
for k=1:niter
    for j =2:N+1
        for i=2:M+1
            phi(i,j)=(phi(i-1,j)+phi(i+1,j)+phi(i,j-1)+phi(i,j+1))/4-h^2*f(i,j)
                /4; %copied exact
        end
    end
    phi=bcGS(phi);
end
end
```

```
function phi=myMultigrid(phi,f,h)
%global Mfine;
M = size(phi,1)-2; N = size(phi,2)-2;
phi = myGaussSeidel(phi,f,h,1);

if mod(M,2)==0 && mod(N,2)==0
  rh = myResidual(phi,f,h);
  r2h = myRestrict(rh);
  e2h = zeros(M/2+2,N/2+2); %2D cell centered
  e2h = myMultigrid(e2h,r2h,2*h);
  eh = myProlong(e2h);
  phi = phi + eh;
  phi = bcGS(phi);
  phi = myGaussSeidel(phi,f,h,1);
else
  phi = myGaussSeidel(phi,f,h,3);
end
```

```
function [phi,Linf,iter] = myPoisson(phi, f, h, nIterMax, epsilon)
[M,N]=size(phi);
M=M-2;
N=N-2;
residual_norm(1) = myRelResNorm(phi,f,h);
phi=myMultigrid(phi,f,h);
iter=1;
    for i=2:nIterMax
        phi=myMultigrid(phi,f,h);
        residual_norm(i) = myRelResNorm(phi,f,h);
        if residual_norm(i)<epsilon
            break;
        end
        iter=iter+1;
    end
    Linf=myRelResNorm(phi,f,h);
end
```

```
function eh=myProlong(e2h)
[M,N] = size(e2h);
M=M-2;                                  %size of coarse values, x
N=N-2;                                  %size of coare values, y

eh=zeros(2*M+2,2*N+2);
```

```
for j=2:N+1                          %loop over size of coarse mesh mid values
    for i =2:M+1
        eh(2*i-2,2*j-2)=e2h(i,j);
        eh(2*i-1,2*j-2)=e2h(i,j);
        eh(2*i-2,2*j-1)=e2h(i,j);
        eh(2*i-1,2*j-1)=e2h(i,j);
    end
end
eh = bcGS(eh);
end
```

```
function rr= myRelResNorm(phi,f,h)
[M,N] = size(phi);
M=M-2;
N=N-2;

%calling myResidual for residual
r=myResidual(phi,f,h);

%residual inf norm
r_inf= max(max(abs(r(2:M+1,2:N+1))));

%function inf norm
f_inf= max(max(abs(f(2:M+1,2:N+1))));

%relative residual norm
rr=r_inf/f_inf;

end
```

```
function r = myResidual(phi,f,h)
[M,N] = size(phi);
M=M-2;
N=N-2;
%residual for exterior points/initialization
r=zeros(M+2,N+2);
    for j = 2:N+1
        for i = 2:M+1
            r(i,j) = f(i,j)-(1/h^2)*(phi(i+1,j)-4*phi(i,j)+phi(i-1,j)+phi(i,j
                +1)+phi(i,j-1));
        end
    end
end
```

```
function r2h=myRestrict(rh)
[M,N] = size(rh);
M=M-2;                                   %size of fine values, x
N=N-2;                                   %size of fine values, y

r2h=zeros(M/2+2,N/2+2);
for j=2:N/2+1                            %loop over size of coarse mesh mid values
    for i =2:M/2+1
        r2h(i,j)=0.25*(rh(2*i-2,2*j-2)+rh(2*i-1,2*j-2)+rh(2*i-2,2*j-1)+rh(2*i
            -1,2*j-1));
```

```
    end
end

end
```

```
function x = mySolveTriDiag(a,b,c,d)
%format long; format compact;
if (length(a)==length(b) && length(b)==length(c) && length(c)==length(d))
    P =length(a);

    %forward elimination
    for i = 2:P
        b(i)=b(i)-(c(i-1).*a(i)./b(i-1));
        d(i)=d(i)-(d(i-1).*a(i)./b(i-1));
    end

    %back-substitution
    d(P)=d(P)./b(P);
    for i=P-1:-1:1
        d(i)=(d(i)-c(i).*d(i+1))./b(i);
    end
    x=d;
else
    error ERROR Make all vectors of same length!
end
```

```
function I = myTrapezoidal(x, y)

    % Number of data points
    N = numel(x);

    % Initialize integral
    I = 0;

    % Calculate integral using composite trapezoidal rule
    for i = 1:N-1
        % Interval width
        h = x(i+1) - x(i);

        % Trapezoidal rule for this interval
        I = I + 0.5 * h * (y(i) + y(i+1));
    end
end
```

```
function [u] = parabolic_CN1_2D_u(u, Qu, dt)
    global Re;
    global t;
    global h;

    global a b c d;


    [M,N]=size(u);
    M=M-1;
```

```
    N=N-2;

    % Initialize coefficient vectors
    a = zeros(M+1, N+2);
    b = zeros(M+1, N+2);
    c = zeros(M+1, N+2);
    d = zeros(M+1, N+2);

    % Calculate coefficients for tridiagonal matrix
    a(2:M, 2:N+1) = - dt / (2 * (h^2)*Re);
    b(2:M, 2:N+1) = 1 + (dt / ((h^2)*Re));
    c(2:M, 2:N+1) = - dt / (2 * (h^2)*Re);
    % Calculate d coefficients using the parabolic CN method
    for j =2:N+1
        for i = 2:M
            d(i, j) = (dt / (2 * (h^2)*Re)) * u(i, j+1) + (1 - ( dt /(Re * (h
                ^2)))) * u(i, j) + (dt / (2 * (h^2)*Re)) * u(i, j-1) + (dt/2)
                * Qu(i, j);
        end
    end

    [a, b, c, d] = bcCN1_u(a, b, c, d, t + dt/2);

    % Solve the tridiagonal system of equations
    for j=2:N+1
        u(2:M,j) = mySolveTriDiag(a(2:M,j), b(2:M,j), c(2:M,j), d(2:M,j));
    end
    u = bc_u(u, t + dt/2);
end
```

```
function [v] = parabolic_CN1_2D_v(v, Qv, dt)
    global Re;
    global t;
    global h;

    global a b c d;


    [M,N]=size(v);
    M=M-2;
    N=N-1;

    % Initialize coefficient vectors
    a = zeros(M+2, N+1);
    b = zeros(M+2, N+1);
    c = zeros(M+2, N+1);
    d = zeros(M+2, N+1);

    % Calculate coefficients for tridiagonal matrix
    a(2:M+1, 2:N) = - dt / (2 * (h^2)*Re);
    b(2:M+1, 2:N) = 1 + (dt / ((h^2)*Re));
    c(2:M+1, 2:N) = - dt / (2 * (h^2)*Re);
    % Calculate d coefficients using the parabolic CN method
    for j =2:N
```

```
        for i = 2:M+1
            d(i, j) = (dt / (2 * (h^2)*Re)) * v(i, j+1) + (1 - ( dt /(Re * (h^2)))) * v(i, j) + (dt / (2 * (h^2)*Re)) * v(i, j-1) + (dt/2) * Qv(i, j);
        end
    end

    [a, b, c, d] = bcCN1_v(a, b, c, d, t + dt/2);

    % Solve the tridiagonal system of equations
    for j=2:N
        v(2:M+1,j) = mySolveTriDiag(a(2:M+1,j), b(2:M+1,j), c(2:M+1,j), d(2:M+1,j));
    end
    v = bc_v(v, t + dt/2);
end
```

```
function [Y] = parabolic_CN1_2D_Y3(Y, QY, dt)
    global Re;
    global Sc;
    global t;
    global h;

    global a b c d;


    [M,N]=size(Y);
    M=M-6;
    N=N-6;

    % Initialize coefficient vectors
    a = zeros(M+6, N+6);
    b = zeros(M+6, N+6);
    c = zeros(M+6, N+6);
    d = zeros(M+6, N+6);

    % Calculate coefficients for tridiagonal matrix
    a(4:M+3, 4:N+3) = - dt / (2 * (h^2)*Re*Sc);
    b(4:M+3, 4:N+3) = 1 + (dt / ((h^2)*Re*Sc));
    c(4:M+3, 4:N+3) = - dt / (2 * (h^2)*Re*Sc);
    % Calculate d coefficients using the parabolic CN method
    for j =4:N+3
        for i = 4:M+3
            d(i, j) = (dt / (2 * (h^2)*Re*Sc)) * Y(i, j+1) + (1 - ( dt /(Re * Sc* (h^2)))) * Y(i, j) + (dt / (2 * (h^2)*Re*Sc)) * Y(i, j-1) + (dt/2) * QY(i, j);
        end
    end

    [a, b, c, d] = bcCN1_Y3(a, b, c, d, t + dt/2);

    % Solve the tridiagonal system of equations
    for j=4:N+3
        Y(4:M+3,j) = mySolveTriDiag(a(4:M+3,j), b(4:M+3,j), c(4:M+3,j), d(4:M
```

```
            +3,j));
    end
    Y = bc_Y3(Y, t + dt/2);
end
```

```
function [u] = parabolic_CN2_2D_u(u, Qu, dt)
    global Re;
    global t;
    global h;

    global a b c d;

    [M,N]=size(u);
    M=M-1;
    N=N-2;

    % Initialize coefficient vectors
    a = zeros(M+1, N+2);
    b = zeros(M+1, N+2);
    c = zeros(M+1, N+2);
    d = zeros(M+1, N+2);

    % Calculate coefficients for tridiagonal matrix
    a(2:M, 2:N+1) = - dt / (2 * (h^2)*Re);
    b(2:M, 2:N+1) = 1 + (dt / ((h^2)*Re));
    c(2:M, 2:N+1) = - dt / (2 * (h^2)*Re);
    % Calculate d coefficients using the parabolic CN method
    for i =2:M
        for j = 2:N+1
            d(i, j) = (dt / (2 * (h^2)*Re)) * u(i+1, j) + (1 - ( dt /(Re * (h
                ^2)))) * u(i, j) + (dt / (2 * (h^2)*Re)) * u(i-1, j) + (dt/2)
                * Qu(i, j);
        end
    end

    [a, b, c, d] = bcCN2_u(a, b, c, d, t + dt);

    % Solve the tridiagonal system of equations
    for i=2:M
        u(i,2:N+1) = mySolveTriDiag(a(i,2:N+1), b(i,2:N+1), c(i,2:N+1), d(i,2:
            N+1));
    end
    u = bc_u(u, t + dt);
end
```

```
function [v] = parabolic_CN2_2D_v(v, Qv, dt)
    global Re;
    global t;
    global h;

    global a b c d;

    [M,N]=size(v);
    M=M-2;
```

```
    N=N-1;


    % Initialize coefficient vectors
    a = zeros(M+2, N+1);
    b = zeros(M+2, N+1);
    c = zeros(M+2, N+1);
    d = zeros(M+2, N+1);

    % Calculate coefficients for tridiagonal matrix
    a(2:M+1, 2:N) = - dt / (2 * (h^2)*Re);
    b(2:M+1, 2:N) = 1 + (dt / ((h^2)*Re));
    c(2:M+1, 2:N) = - dt / (2 * (h^2)*Re);
    % Calculate d coefficients using the parabolic CN method
    for i =2:M+1
        for j = 2:N
            d(i, j) = (dt / (2 * (h^2)*Re)) * v(i+1, j) + (1 - ( dt /(Re * (h
                ^2)))) * v(i, j) + (dt / (2 * (h^2)*Re)) * v(i-1, j) + (dt/2)
                * Qv(i, j);
        end
    end
    [a, b, c, d] = bcCN2_v(a, b, c, d, t+dt);

    % Solve the tridiagonal system of equations
    for i=2:M+1
        v(i,2:N) = mySolveTriDiag(a(i,2:N), b(i,2:N), c(i,2:N), d(i,2:N));
    end
    v = bc_v(v, t+dt);
end
```

```
function [Y] = parabolic_CN2_2D_Y3(Y, QY, dt)
    global Re;
    global Sc;
    global t;
    global h;

    global a b c d;


    [M,N]=size(Y);
    M=M-6;
    N=N-6;

    % Initialize coefficient vectors
    a = zeros(M+6, N+6);
    b = zeros(M+6, N+6);
    c = zeros(M+6, N+6);
    d = zeros(M+6, N+6);

    % Calculate coefficients for tridiagonal matrix
    a(4:M+3, 4:N+3) = - dt / (2 * (h^2)*Re*Sc);
    b(4:M+3, 4:N+3) = 1 + (dt / ((h^2)*Re*Sc));
    c(4:M+3, 4:N+3) = - dt / (2 * (h^2)*Re*Sc);
    % Calculate d coefficients using the parabolic CN method
```

```
    for i =4:M+3
        for j = 4:N+3
            d(i, j) = (dt / (2 * (h^2)*Re*Sc)) * Y(i+1, j) + (1 - ( dt /(Re *
                Sc* (h^2)))) * Y(i, j) + (dt / (2 * (h^2)*Re*Sc)) * Y(i-1, j)
                + (dt/2) * QY(i, j);
        end
    end

    [a, b, c, d] = bcCN2_Y3(a, b, c, d, t + dt);

    % Solve the tridiagonal system of equations
    for i=4:M+3
        Y(i,4:N+3) = mySolveTriDiag(a(i,4:N+3), b(i,4:N+3), c(i,4:N+3), d(i,4:
            N+3));
    end
    Y = bc_Y3(Y, t + dt);
end
```

```
function [u,v] = projectV(u,v, phi, dt)
    global h t;

    % Calculate grid dimensions
    [M1, N1] = size(u);
    M1 = M1 - 1;
    N1 = N1 - 2;

    [M2, N2] = size(v);
    M2 = M2 - 2;
    N2 = N2 - 1;



    % Project u onto the subspace of divergence-free fields
    % Find grad phi at the nodes i.e. 1 to M+1, it is currently cell centered
        values
    gradphi1=zeros(M1+1,N1+2);
    gradphi2=zeros(M2+2,N2+1);


    % grad of phi1
    for j =2:N1+1
        for i = 2:M1
            gradphi1(i, j) = (phi(i+1, j) - phi(i, j)) / (h);
        end
    end

     % grad of phi2
    for i =2:M2+1
        for j = 2:N2
            gradphi2(i, j) = (phi(i, j+1) - phi(i, j)) / (h);
        end
    end


```

```
    u = u - dt * gradphi1;
    v = v - dt * gradphi2;


    for i = 2:M1
        u=bcGhost_u(u,t);
    end

    for i = 2:N2
        v=bcGhost_v(v,t);
    end





end
```

```
function psi=psiWENO(a,b,c,d)
    global IS0 IS1 IS2 alpha0 alpha1 alpha2 omega0 omega2

    ep=10^-6;

    IS0=13*(a-b)^2+3*(a-3*b)^2;
    IS1=13*(b-c)^2+3*(b+c)^2;
    IS2=13*(c-d)^2+3*(3*c-d)^2;

    alpha0=1/(ep+IS0)^2;
    alpha1=6/(ep+IS1)^2;
    alpha2=3/(ep+IS2)^2;

    omega0=alpha0/(alpha0+alpha1+alpha2);
    omega2=alpha2/(alpha0+alpha1+alpha2);

    psi=1/3*omega0*(a-2*b+c)+1/6*(omega2-0.5)*(b-2*c+d);

end
```