MAE 561

Computational Fluid Dynamics

Final Project

By: Emilio Torres

Introduction:

The purpose of this problem is to solve for the steady state flow inside a 2D mixing chamber. The chamber is a simple square box with a no slip condition on all sides. The left and right wall are at rest (i.e. $u = v = 0$), the top wall moves with a velocity of $A = 1$ in the positive x direction, and the bottom wall moves with a velocity of $B = 1/2$ in the negative x direction. Figure 1. below represents the problem's geometry and the velocities at the boundaries:

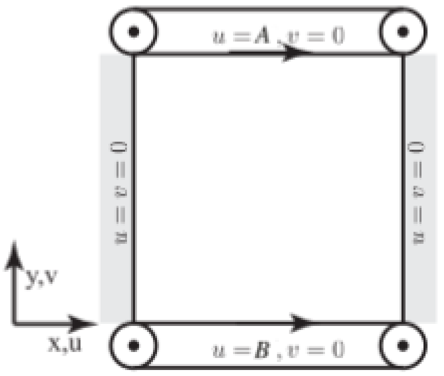


Figure 1: Lid driven cavity.

The flow can be governed by the incompressible 2D Navier Stokes equations below:

$$\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0 \\ \frac{\partial \boldsymbol{v}}{\partial t} + \nabla \cdot (\boldsymbol{v}\boldsymbol{v}) &= -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{v} \end{aligned}$$

Where Re is the Reynolds number. The Fractional Step Method on a staggered grid in conservative form is utilized to solve the problem. The split momentum equation takes the form of:

$$\frac{\partial \vec{v}}{\partial t} = N(\vec{v}) + \frac{1}{Re}L(\vec{v}) \qquad : \qquad \vec{v}^n \to \vec{v}^*$$

$$\frac{\partial \vec{v}}{\partial t} = -\nabla\varphi \qquad : \qquad \vec{v}^* \to \vec{v}^{n+1}$$

where

$$N(\vec{v}) = -\nabla \cdot (\vec{v}\vec{v})$$

$$L(\vec{v}) = \nabla^2 \vec{v}$$

The first equation represent the intermediate velocity term which will be denoted with a star symbol. The second equation uses the pressure and star velocities to solve for the velocity at the next time step. Splitting the momentum equation though may cause the velocity solved for at the next time step to violate the continuity equations. To account possibility, the velocities solved for are projected into the desired subspace. Because of this the pressure term is no longer determined using diffusion equations. Therefore the term is not truly a pressure but a Lagrange multiplier, and is denoted with φ. The problem is broken down into three steps:

1. Solve the first part of momentum equation (viscous Burger's equation) to determine the velocities at the intermediate time step using a time stepping method.
2. Calculate the Lagrange multiplier by solving the pressure Poisson equation using an implicit iterative method.
3. Use the final Lagrange multiplier to project the intermediate velocities and solve for the velocity at the next time step.

To solve the steady state equation velocity value a convergence is set using the change in velocity over time.

Before any of the above steps can be performed the grid has to be defined. Using the cell as a control volume the velocities are defined at the cell faces. The pressure or Lagrange multiplier are defined at the cell centers. In order to use to the most compressed grid the vorticity is defined at cell centers. The following figure below is used to represent the grid used:

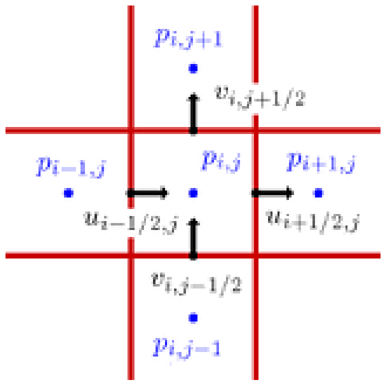


Figure 2: Staggered Grid Cell

Due to the grid the velocity index have to be redefined from the usual cell centered to cell faces. The following notation was used in the code of the solver:

$$u(i,j) = u_{i+\frac{1}{2},j}$$

$$v(i,j) = v_{i,j+\frac{1}{2}}$$

Using the staggered grid leads to different size *u* and *v* vectors because the values of velocity are located at cell face. The Lagrange multiplier term however remains at the cell center and utilizes the standard notation in the solver. Using the staggered grid shown above the three steps outlined above can be performed.

Method:

Step 1: Solving v^* using the FTCS Method

The first step in solving for steady state is solving for the intermediate velocity using the FTCS Method. The vicious Burger's equation is solved to determine v^*. The following equation represents the conservative form of the momentum equation split into vector components:

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

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

Taking a finite difference of the above equation yields the following equations:

$$u^*_{i+\frac{1}{2},j} = u^n_{i+\frac{1}{2},j} + \Delta t\Bigg[-\left(\frac{\left(u^n_{i+1,j}\right)^2 - \left(u^n_{i,j}\right)^2}{\Delta x} + \frac{(uv)^n_{i+\frac{1}{2},j+\frac{1}{2}} - (uv)^n_{i+\frac{1}{2},j-\frac{1}{2}}}{\Delta y}\right) + \frac{1}{Re}\left(\frac{u^n_{i+\frac{3}{2},j} - 2u^n_{i+\frac{1}{2},j} + u^n_{i-\frac{1}{2},j}}{\Delta x^2} + \frac{u^n_{i+\frac{1}{2},j+1} - 2u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j-1}}{\Delta y^2}\right)\Bigg]$$

$$v^*_{i,j+\frac{1}{2}} = v^n_{i,j+\frac{1}{2}} + \Delta t\Bigg[-\left(\frac{(uv)^n_{i+\frac{1}{2},j+\frac{1}{2}} - (uv)^n_{i-\frac{1}{2},j+\frac{1}{2}}}{\Delta x} + \frac{\left(v^n_{i,j+1}\right)^2 - \left(v^n_{i,j}\right)^2}{\Delta y}\right) + \frac{1}{Re}\left(\frac{v^n_{i+1,j+\frac{1}{2}} - 2v^n_{i,j+\frac{1}{2}} + v^n_{i-1,j+\frac{1}{2}}}{\Delta x^2} + \frac{v^n_{i,j+\frac{3}{2}} - 2v^n_{i,j+\frac{1}{2}} + v^n_{i,j-\frac{1}{2}}}{\Delta y^2}\right)\Bigg]$$

Because the velocities are located at the cell faces on the staggered grid several of the values in the finite difference formula are determined using linear interpolation from known surrounding velocities. The index formulas used to solve for the velocities not located at the cell faces are as follows:

$$u^n_{i+1,j} = \frac{1}{2}\left(u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{3}{2},j}\right)$$

$$u^n_{i,j} = \frac{1}{2}\left(u^n_{i+\frac{1}{2},j} + u^n_{i-\frac{1}{2},j}\right)$$

$$u^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}\left(u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j+1}\right)$$

$$u^n_{i+\frac{1}{2},j-\frac{1}{2}} = \frac{1}{2}\left(u^n_{i+\frac{1}{2},j} + u^n_{i+\frac{1}{2},j-1}\right)$$

$$u^n_{i-\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}\left(u^n_{i-\frac{1}{2},j} + u^n_{i-\frac{1}{2},j+1}\right)$$

$$v^n_{i,j+1} = \frac{1}{2}\left(v^n_{i,j+\frac{1}{2}} + v^n_{i,j+\frac{3}{2}}\right)$$

$$v^n_{i,j} = \frac{1}{2}\left(v^n_{i,j+\frac{1}{2}} + v^n_{i,j-\frac{1}{2}}\right)$$

$$v^n_{i-\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}\left(v^n_{i-1,j+\frac{1}{2}} + v^n_{i,j+\frac{1}{2}}\right)$$

$$v^n_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}\left(v^n_{i,j+\frac{1}{2}} + v^n_{i+1,j+\frac{1}{2}}\right)$$

$$v^n_{i+\frac{1}{2},j-\frac{1}{2}} = \frac{1}{2}\left(v^n_{i,j-\frac{1}{2}} + v^n_{i+1,j-\frac{1}{2}}\right)$$

Step 2: Solving the Lagrange Multiplier

The Lagrange multiplier is solved from the pressure Poisson equation using the Gauss Seidel implicit iterative method. The following represents the pressure Poisson equation:

$$\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} = \frac{1}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right)$$

Taking a finite difference from the above equation and solving for $\varphi^{n+1}_{i,j}$ yields the following:

$$\varphi^{n+1}_{i,j} = \frac{1}{4}\left(\varphi^{n+1}_{i-1,j} + \varphi^n_{i+1,j} + \varphi^{n+1}_{i,j-1} + \varphi^n_{i,j+1}\right) - \frac{1}{4}\frac{1}{\Delta t}\frac{\left(u^*_{i+\frac{1}{2},j} - u^*_{i-\frac{1}{2},j}\right)}{\Delta x} + \frac{\left(v^*_{i,j+\frac{1}{2}} - v^*_{i,j-\frac{1}{2}}\right)}{\Delta y}$$

The convergence of the Gauss Seidel is solved using the residual. The residual is equal to difference of the Poisson equation from the right hand side. The following represents the finite difference equation of the residual:

$$Residual = \frac{\left(\varphi^n_{i+1,j} - 2\varphi^n_{i,j} + \varphi^n_{i-1,j}\right)}{\Delta x^2} + \frac{\left(\varphi^n_{i,j+1} - 2\varphi^n_{i,j} + \varphi^n_{i,j-1}\right)}{\Delta y^2} - \frac{1}{\Delta t}\left(\frac{\left(u^*_{i+\frac{1}{2},j} - u^*_{i-\frac{1}{2},j}\right)}{\Delta x} + \frac{\left(v^*_{i,j+\frac{1}{2}} - v^*_{i,j-\frac{1}{2}}\right)}{\Delta y}\right)$$

Step 3: Solving for v^{n+1}

The final step in solving for velocity is to project the intermediate velocities, solved for in step one, into the subspace of the solenoidal using the Lagrange multiplier solved for in step tow. The following relationship is used to solve for the velocity at the next time step:

$$v^{n+1} = v^* - \Delta grad(\varphi^{n+1})$$

Writing the above equation in finite difference form produces the following equation:

$$u^{n+1}_{i+\frac{1}{2},j} = u^{*}_{i+\frac{1}{2},j} - \frac{\Delta t}{\Delta x}\left(\varphi^{n}_{i+1,j} - \varphi^{n}_{i,j}\right)$$

$$v^{n+1}_{i,j+\frac{1}{2}} = v^{*}_{i,j+\frac{1}{2}} - \frac{\Delta t}{\Delta y}\left(\varphi^{n}_{i,j+1} - \varphi^{n}_{i,j}\right)$$

The last operation that has to be performed is determining when the flow reaches steady state. To determine when the flow reaches steady state when the difference between $\boldsymbol{v}^{n+1}$ and $\boldsymbol{v}^{n}$ are dived by the change in time drop under a set convergence criteria. The following represent the index formula of the change in velocity over time:

$$Con_u = \frac{u^{n}_{i+\frac{1}{2},j} - u^{n-1}_{i+\frac{1}{2},j}}{\Delta t}$$

$$con_v = \frac{v^{n}_{i,j+\frac{1}{2}} - v^{n-1}_{i,j+\frac{1}{2}}}{\Delta t}$$

<u>Vorticity:</u>

The vorticity can defined with the following equations:

$$\Omega = \nabla \text{ x } \vec{v}$$

$$\Omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

A result of using the staggered mesh is that the vorticity has to be calculated at the corners of every cell in order to have the most compact stencil. Taking a finite difference of the above equations provides the following index equations for vorticity:

$$\Omega_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{v_{i+1,j+\frac{1}{2}} - v_{i,j+\frac{1}{2}}}{\Delta x} - \frac{u_{i+\frac{1}{2},j+1} - u_{i+\frac{1}{2},j}}{\Delta y}$$

It should be noted that the velocity used to calculate the vorticity is the steady state value. Since the vorticity is calculated at the corners and all the corners occur inside the spatial domain the boundary conditions can be calculated during the looping of the interior domain.

<u>Stream Function:</u>

The following equation defines the stream function:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = -\Omega$$

The index formula to the stream function can be determined by taking a finite difference of the above equation. The following represents the index formula of the stream line located at the corners of the cells:

$$\frac{\psi_{i+\frac{3}{2},j+\frac{1}{2}} - 2\psi_{i+\frac{1}{2},j+\frac{1}{2}} + \psi_{i-\frac{1}{2},j+\frac{1}{2}}}{\Delta x^2} + \frac{\psi_{i+\frac{1}{2},j+\frac{3}{2}} - 2\psi_{i+\frac{1}{2},j+\frac{1}{2}} + \psi_{i+\frac{1}{2},j-\frac{1}{2}}}{\Delta y^2} = -\Omega_{i+\frac{1}{2},j+\frac{1}{2}}$$

It should be noted that the boundary conditions of the stream line are also zero.

Boundary Conditions:

One of the benefits of using the staggered grid mesh is that the velocities perpendicular to wall are known, unfortunately the though the velocities tangential to the walls are unknown. Because the tangential velocities are unknown, linear interpolation alongside some algebra have to be used in order to solve for the ghost cell boundary condition. The following figure represents the staggered mesh with the gray cell representing the ghost cell:

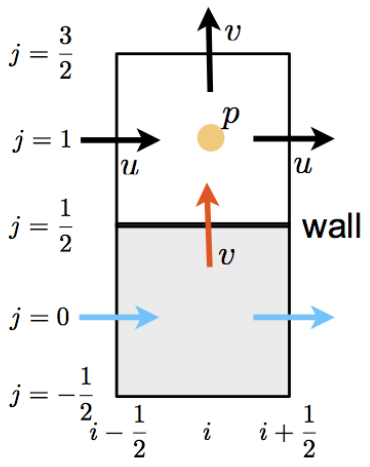


Figure 3: Ghost Cell Boundary Conditions

The figure represents the lower boundary layer of the wall where the tangential ghost cell velocity is unknown. The given value for velocity on the wall is used below to solve for the ghost cell value:

$$u_{wall} = \frac{1}{2}\left(u_{i+\frac{1}{2},1} + u_{i+\frac{1}{2};0}\right)$$

Solving the above equation for $u_{i+1/2}$ yields the following:

$$u_{i+\frac{1}{2};0} = 2u_{wall} - u_{i+\frac{1}{2};1}$$

It should be noted that staggered mesh results in the in different dimensions for the *u velocity* and *v velocity*. Using the outline method above the boundary conditions can be defined as followed:

$$u^n_{i+\frac{1}{2},N+1} = 2A - u^n_{i+\frac{1}{2},N}$$

$$u^n_{i+\frac{1}{2},0} = 2B = u^n_{i+\frac{1}{2},1}$$

$$u^n_{\frac{1}{2},j} = 0$$

$$u^n_{M+\frac{1}{2},j} = 0$$

$$v^n_{i,N+\frac{1}{2}} = 0$$

$$v^n_{i,\frac{1}{2}} = 0$$

$$v^n_{0,j+\frac{1}{2}} = -v^n_{1,j+\frac{1}{2}}$$

$$v^n_{M+1,j+\frac{1}{2}} = -v^n_{M,j+\frac{1}{2}}$$

These same exact concepts apply to the predictor method and the boundary conditions of the predictor method can be defined as:

$$u^*_{i+\frac{1}{2},N+1} = 2A - u^*_{i+\frac{1}{2},N}$$

$$u^*_{i+\frac{1}{2},0} = 2B - u^*_{i+\frac{1}{2},1}$$

$$u^*_{\frac{1}{2},j} = 0$$

$$u^*_{M+\frac{1}{2},j} = 0$$

$$v^*_{i,N+\frac{1}{2}} = 0$$

$$v^*_{i,\frac{1}{2}} = 0$$

$$v^*_{0,j+\frac{1}{2}} = -v^*_{1,j+\frac{1}{2}}$$

$$v^*_{M+1,j+\frac{1}{2}} = -v^*_{M,j+\frac{1}{2}}$$

The Lagrange multiple boundary layers significantly easier to calculate than the velocity ones. Using the Neumann criteria the boundary layers for φ can be defined with the following equations:

$$\varphi^{n+1}_{i,0} = \varphi^{n+1}_{i,1}$$

$$\varphi^{n+1}_{i,N+1} = \varphi^{n+1}_{i,N}$$

$$\varphi^{n+1}_{0,j} = \varphi^{n+1}_{1,j}$$

$$\varphi^{n+1}_{M+1,j} = \varphi^{n+1}_{M,j}$$

When looking at the boundary conditions for the vorticity because vorticity is exactly one entry smaller in both directions the boundary layer can be determined with in the loop of the interior. Also for the streamline functions all the boundaries are zero resulting in the following index equations:

$$\psi_{\frac{1}{2},j+\frac{1}{2}} = 0$$

$$\psi_{M+\frac{1}{2},j+\frac{1}{2}} = 0$$

$$\psi_{i+\frac{1}{2},\frac{1}{2}} = 0$$

$$\psi_{i+\frac{1}{2},N+\frac{1}{2}} = 0$$

Stability:

There are three different limits that constrain both the spatial and time steps associated with the FTCS method. There are both parabolic and hyperbolic constraints as well the Reynold Cell. To simplify the problem the following is defined for the spatial step sizes $dy = dx = h$. The parabolic time constraints are defined with the following equation:

$$\frac{\nu\Delta t}{h^2} \leq \frac{1}{4}$$

Solving for Δt produces the following equation for the parabolic time step:

$$\Delta t = \frac{h^2}{4\nu}$$

The hyperbolic time step is defined using the following equation:

$$\frac{(a+b)\Delta t}{h} \leq 1$$

Solving for Δt yields the following equation for the hyperbolic criteria:

$$\Delta t = \frac{h}{(a+b)}$$

Where the variables a and b are based off the maximum value of both the u and v velocity vectors. The variable a and b are defined with the following equations:

$$a = \max(u)$$

$$b = \max(v)$$

The Reynolds Cell is calculated using the following equation:

$$\frac{(a+b)h}{\upsilon} \leq 2$$

To solve for the stable time step in the script the smallest value between the parabolic and hyperbolic. The Reynolds Cell value is not used in determining the stable time step since it was found it can be diverged from and still remain stable. However, the Reynolds Cell value is checked and if it is violated an error message displayed to warn the user.

Convergence:

The convergence for the steady state velocity is set in the script to be 1e-5. For solving Lagrange multiplier, the Poisson's equation convergence is dynamic and changes as the overall convergence reduces. It is first defined at 1e-3 and as the total convergence reduces to 1e-3 the Poisson's equation is reduced to 1e-5. Lastly the Streamline function convergence criteria is set to 1e-5 and does not have a dynamic aspect like the Lagrange multiplier.

Accuracy:

The accuracy analysis is performed using the Richardson Extrapolation and Grid Convergence Index. The GCI analysis requires three different grids and to determine the accuracy of the solution. The accuracy requirement in the problem is set at .5% maximum.

The GCI method requires three values for the maximum velocity for both *u* and *v* velocities at steady state. The grids used in the in the analysis are 64,96, and 144 with a *r* value of 1.5. According to Dr. Herrmann the minimum *r* value has to be higher than 1.1. Using the three grids defined the *p* value is solved using the following equation:

$$p = \ln\left(\frac{f_3 - f_2}{f_2 - f_1}\right) / \ln(r)$$

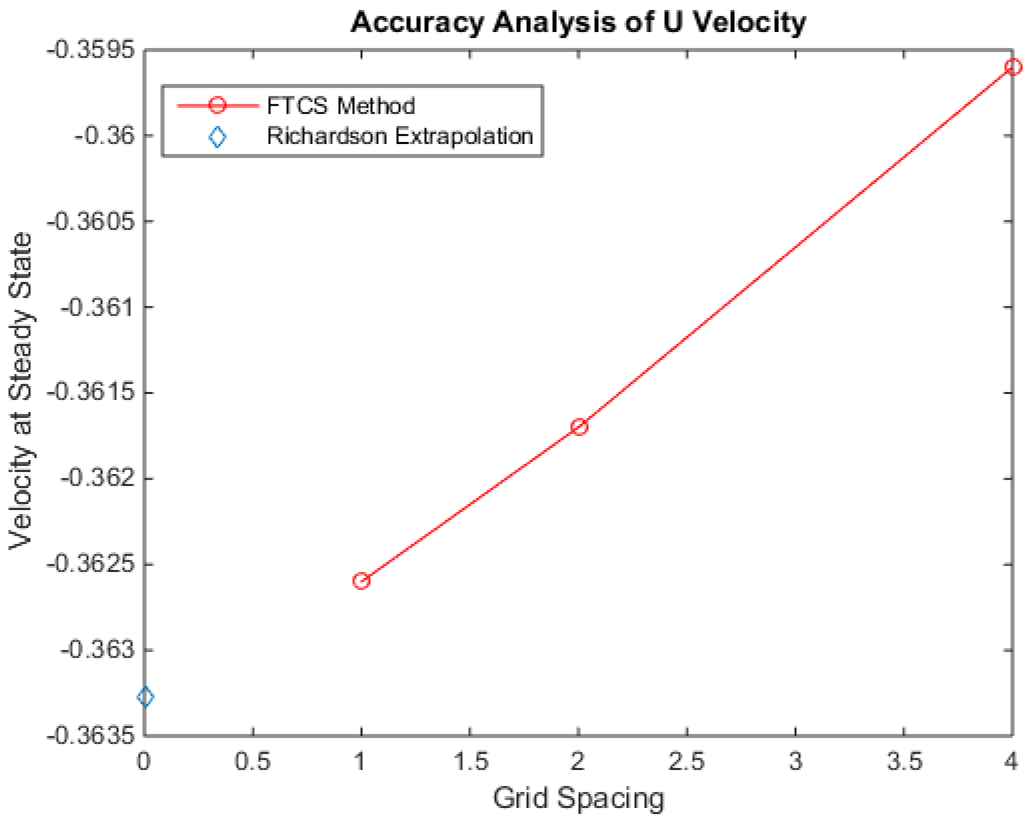
Next, the Richardson Extrapolation or the estimated true solution is found using the following equation:

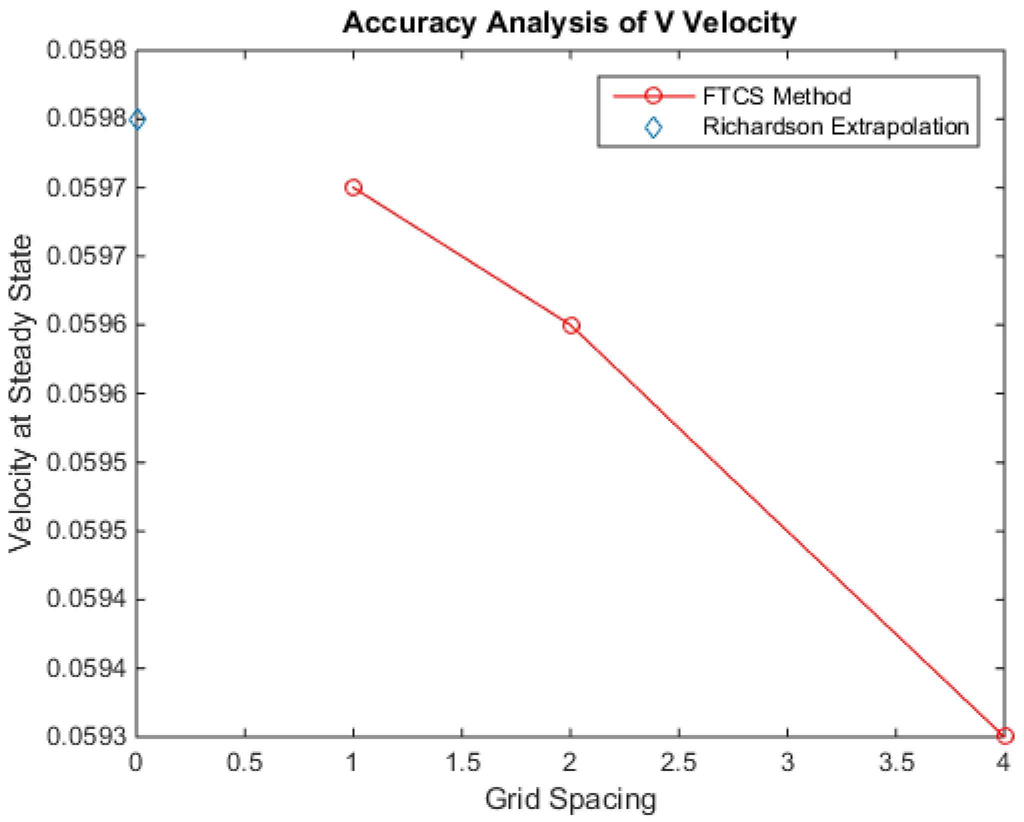
$$f_{h=0} \approx f_1 + \frac{f_1 - f_2}{r^p - 1}$$

The accuracy of the GCI is solved using the following equation:

$$GCI_{12} = 1.25\frac{\left|\frac{f_1 - f_2}{f_1}\right|}{r^p - 1}$$

Using the GCI method outlined above the accuracy solution plots can be generated. The following plots represent the accuracy analysis:


Accuracy Analysis of U Velocity
FTCS Method
Richardson Extrapolation
Velocity at Steady State
Grid Spacing
-0.3595
-0.36
-0.3605
-0.361
-0.3615
-0.362
-0.3625
-0.363
-0.3635
0
0.5
1
1.5
2
2.5
3
3.5
4


Accuracy Analysis of V Velocity
FTCS Method
Richardson Extrapolation
Velocity at Steady State
Grid Spacing
0.0598
0.0598
0.0597
0.0597
0.0596
0.0596
0.0595
0.0595
0.0594
0.0594
0.0593
0
0.5
1
1.5
2
2.5
3
3.5
4

From the GCI analysis it was determined that the error band for the *u* velocity was calculated to be **.233%** and the error band of the *v* velocity was calculated to be **.104%**. The following table represents the values and the accuracy for both the *u* and *v* velocities:

Mesh Size	Steady State Velocity @ x=.5 and y=.5
u-velocity 64	-.3596
u-velocity 96	-.3617
u-velocity 144	-.3626
v-velocity 64	.0593
v-velocity 96	.0596
v-velocity 144	.0597

Solutions:

The following graphs represent the *u* and *v* velocities plotted at the vertical and horizontal center lines:

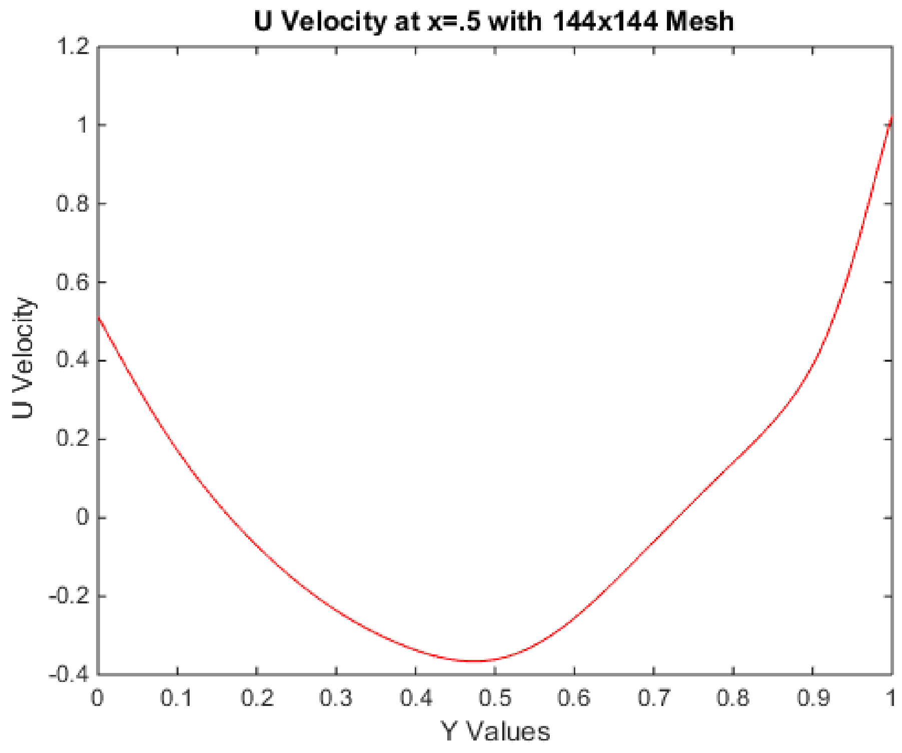


Figure4: Vertical Centerline U velocities

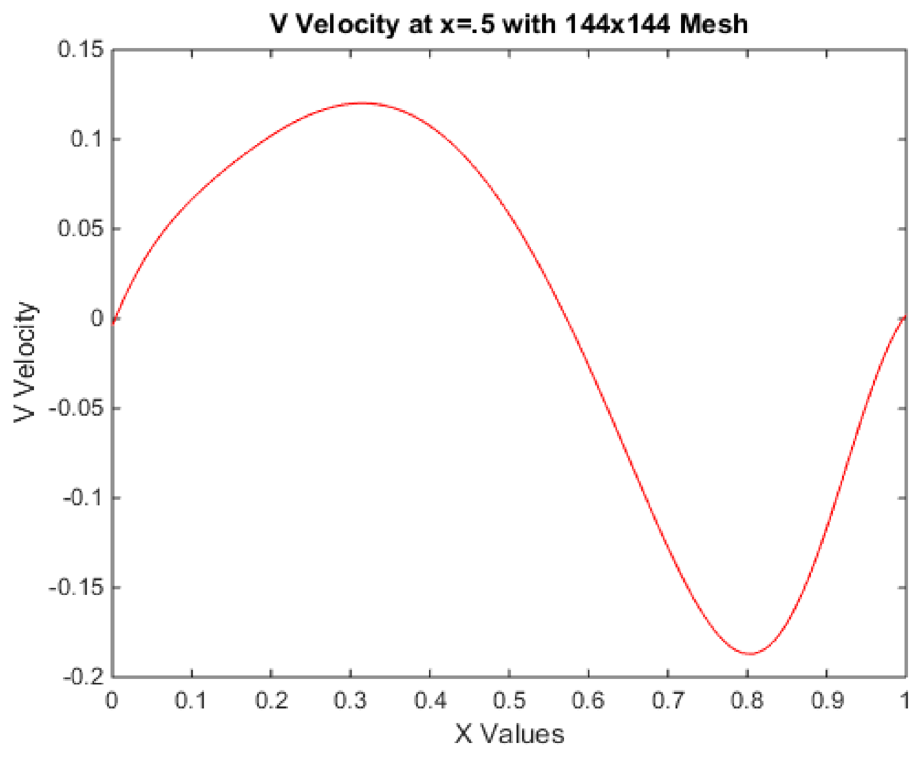


Figure 5: V Velocity at the horizontal centerline

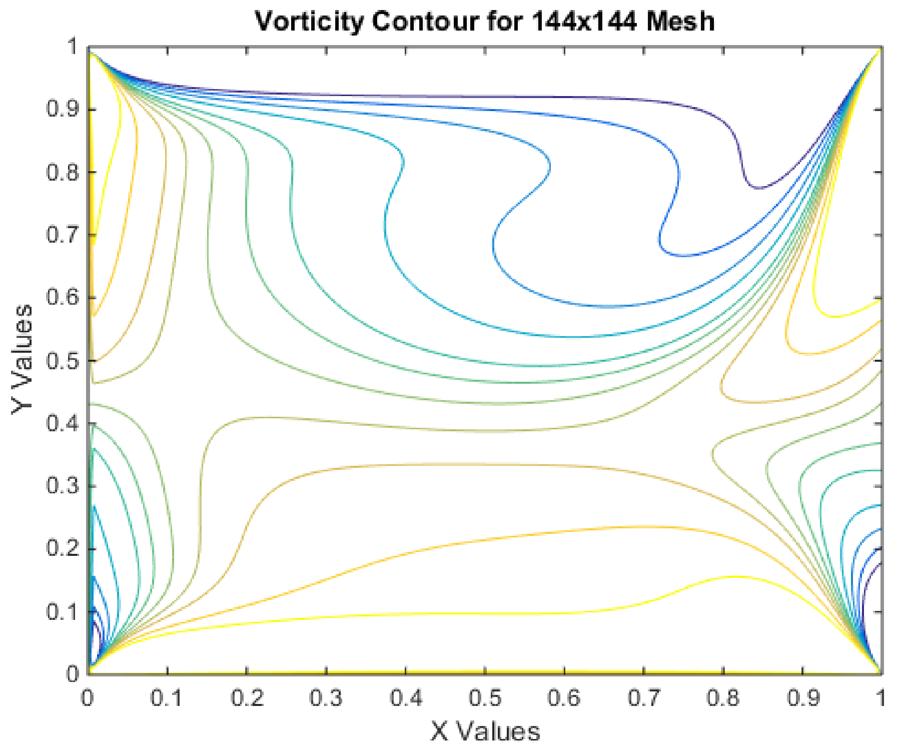


Figure 6: Vorticity plot

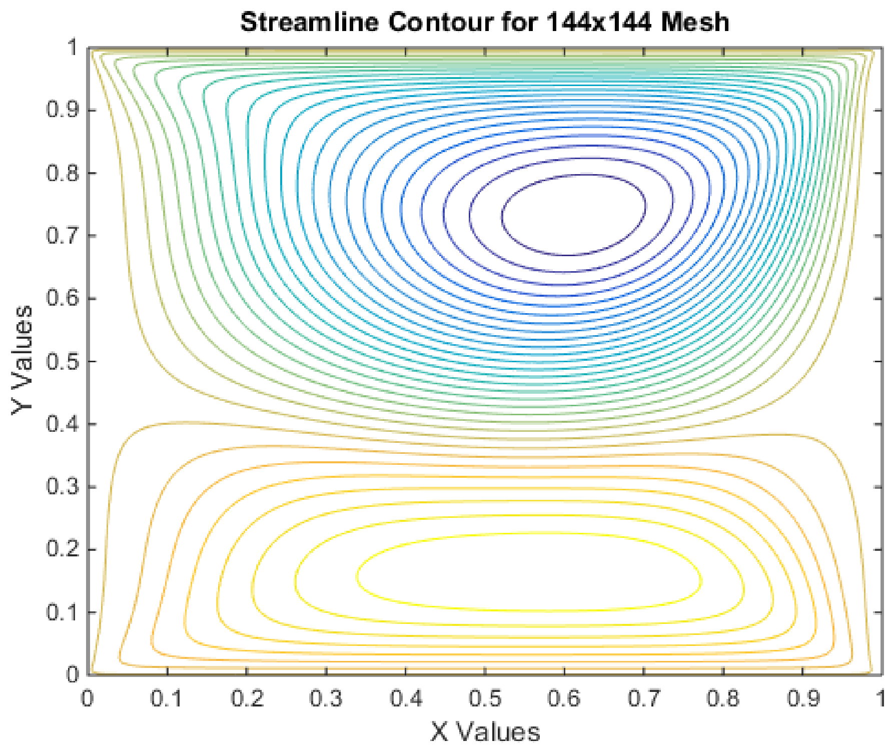


Figure 7: Streamline Contour

Extra Credit:

Problem 3:

Problem consist of solving the above problem but with a larger Reynolds number.

Convergence:

The convergence for all the methods remains the same above.

Stability:

The stability for all the methods remains the same above.

Accuracy:

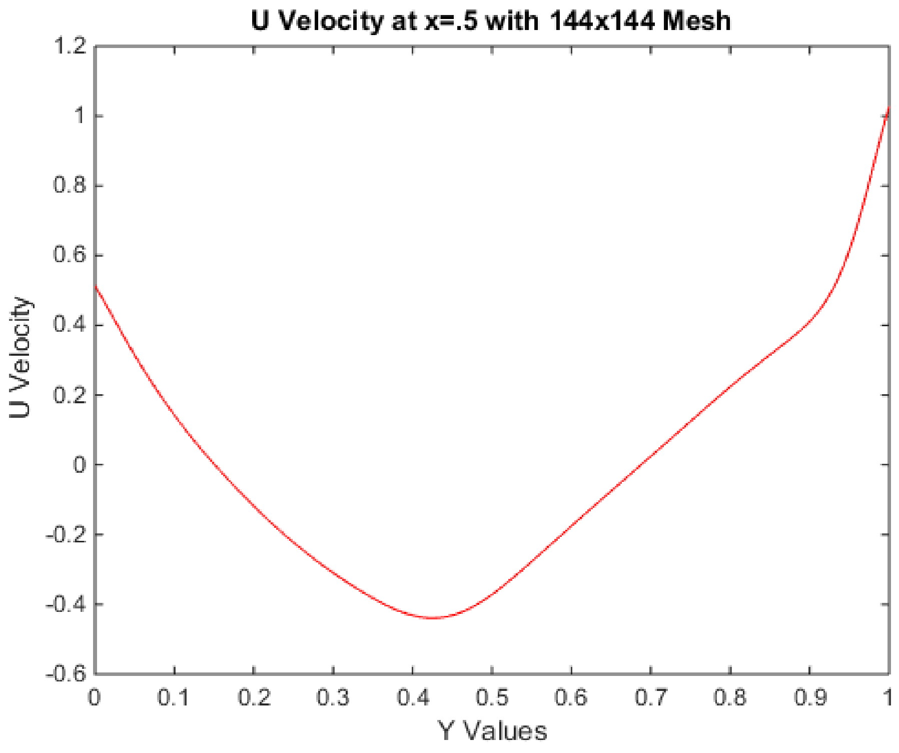
The following table represent the values of the steady state velocities for a *Re* of 400:

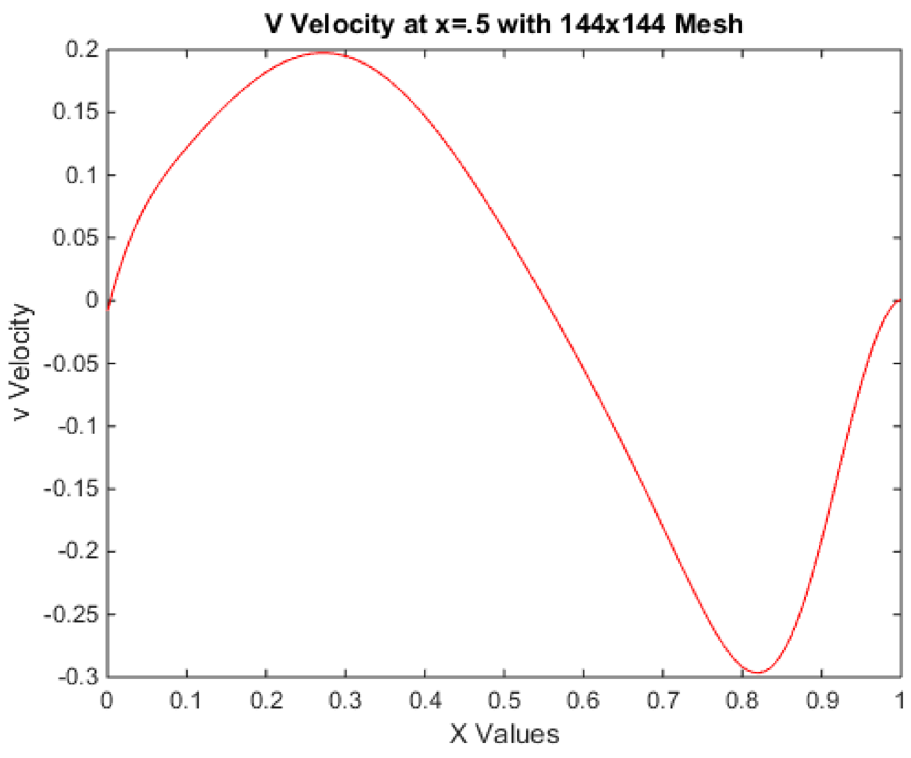
Mesh Size	Steady State Velocity @ x=.5 and y=.5
u-velocity 64	-.3705
u-velocity 96	-.3715
u-velocity 144	-.3718
v-velocity 64	.0566
v-velocity 96	.05675
v-velocity 144	.0568

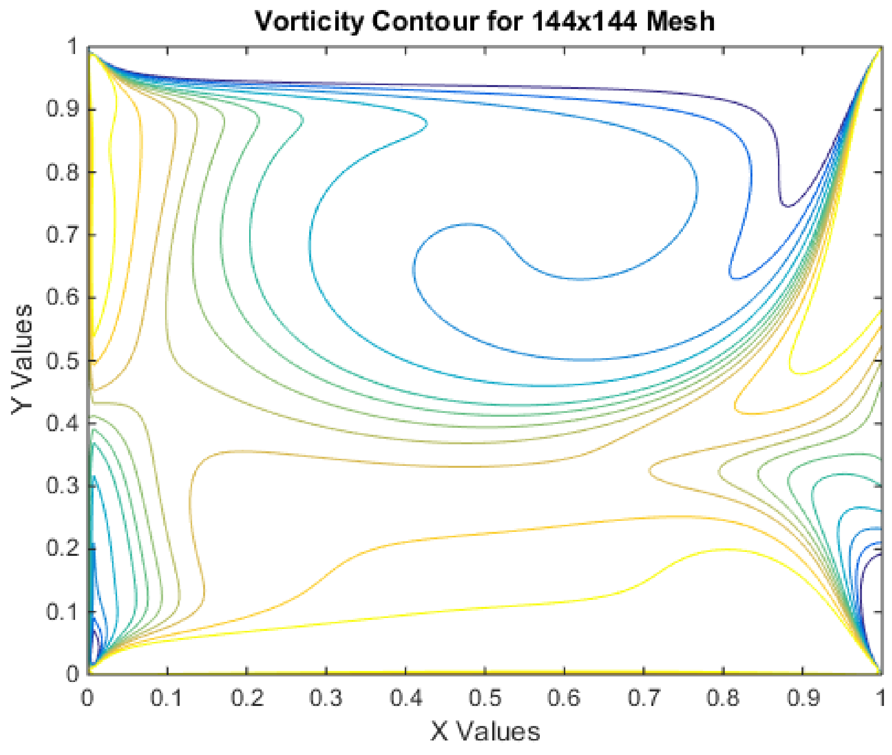
The following table represent the GCI analysis for the higher Reynolds number:

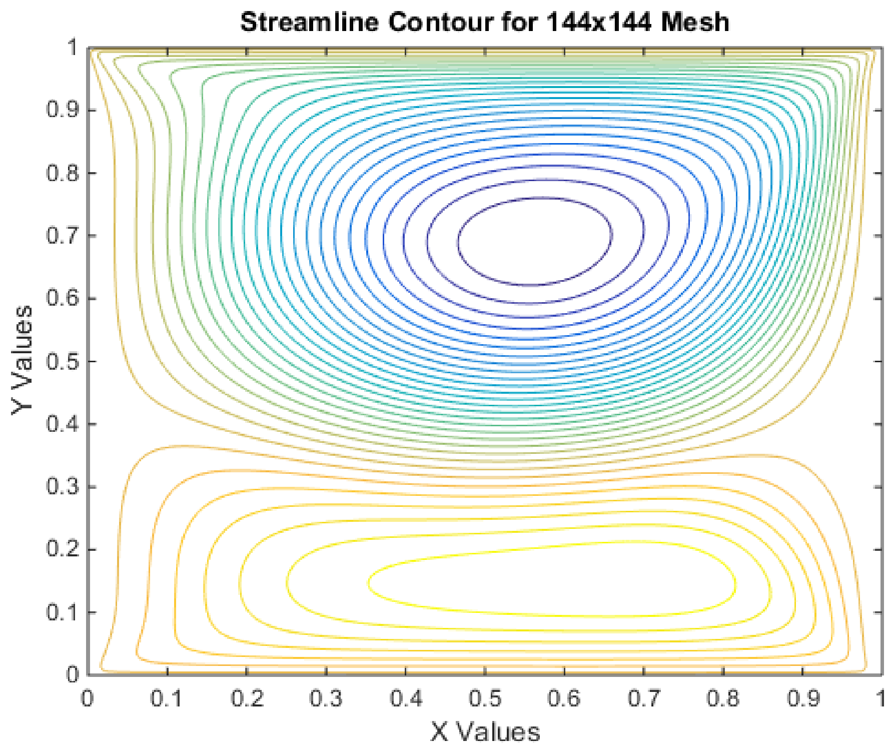
Velocity	GCI Accuracy 2-3	GCI Accuracy 1-2
U velocity	.14%	.04%
V velocity	.16%	.05%

Solution:




V Velocity at x=.5 with 144x144 Mesh
v Velocity
X Values
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Vorticity Contour for 144x144 Mesh
Y Values
X Values
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Extra Credit:

Problem 2:

The second extra credit problem was to solve the transport problem using the equation below:

$$\frac{\partial Y}{\partial t} + \nabla \cdot (\boldsymbol{v}Y) = \frac{Sc}{Re}\nabla^2 Y\,.$$

Taking a finite difference and solving for the Y^{n+1} yields the following equation:

$$Y_{i,j}^{n+1} = Y_{i,j}^{n} - dt * \frac{(Y_{i+1,j}^{n} * u_{i+1,j}^{n} - Y_{i-1,j}^{n} * u_{i-1,j}^{n})}{2 * dx} - dt * \frac{(Y_{i,j+1}^{n} * v_{i,j+1}^{n} - Y_{i,j-1}^{n} * v_{i,j-1}^{n})}{2 * dy} + \frac{Sc * dt}{Re} * \left(\frac{Y_{i+1,j}^{n} - 2Y_{i,j}^{n} + Y_{i-1,j}^{n}}{\Delta x^2} + \frac{Y_{i,j+1}^{n} - 2Y_{i,j}^{n} + Y_{i,j-1}^{n}}{\Delta y^2}\right)$$

Because the velocity used in the finite difference are not located at the cell face they will have to be averaged using the same method outlined above in Problem 1. However though because of the location of the velocity vectors different methods have to be used for the stencils next to the boundaries to ensure there are enough values to be averaged. On the left wall a forward difference for the u velocity term was used since and can be seen below:

$$Y_{i,j}^{n+1} = Y_{i,j}^{n} - dt * \frac{(Y_{i+1,j}^{n} * u_{i+1,j}^{n} - Y_{i,j}^{n} * u_{i,j}^{n})}{2 * dx} - dt * \frac{(Y_{i,j+1}^{n} * v_{i,j+1}^{n} - Y_{i,j-1}^{n} * v_{i,j-1}^{n})}{2 * dy} + \frac{Sc * dt}{Re} * \left(\frac{Y_{i+1,j}^{n} - 2Y_{i,j}^{n} + Y_{i-1,j}^{n}}{\Delta x^2} + \frac{Y_{i,j+1}^{n} - 2Y_{i,j}^{n} + Y_{i,j-1}^{n}}{\Delta y^2}\right)$$

Likewise on the top wall a backwards difference was used for the v velocity difference:

$$Y_{i,j}^{n+1} = Y_{i,j}^{n} - dt * \frac{(Y_{i+1,j}^{n} * u_{i+1,j}^{n} - Y_{i-1,j}^{n} * u_{i-1,j}^{n})}{2 * dx} - dt * \frac{(Y_{i,j}^{n} * v_{i,j}^{n} - Y_{i,j-1}^{n} * v_{i,j-1}^{n})}{2 * dy} + \frac{Sc * dt}{Re} * \left(\frac{Y_{i+1,j}^{n} - 2Y_{i,j}^{n} + Y_{i-1,j}^{n}}{\Delta x^2} + \frac{Y_{i,j+1}^{n} - 2Y_{i,j}^{n} + Y_{i,j-1}^{n}}{\Delta y^2}\right)$$

There are eight different location in the domain were this has to be applied and can be seen in the table below:

Location	U Method	V Method
Left Wall	Forward	Central
Right Wall	Backward	Central
Top Wall	Central	Backward
Top Wall Far Left	Forward	Backward
Top Wall Far Right	Backward	Backward
Bottom Wall Far Left	Forward	Forward
Bottom Wall Far Right	Backward	Forward

It is important to note that even though it the table says left wall….etc… the true location that the table refers is the cell center inside the domain but next to the features in the table.

Boundary Conditions:

Using the Neumann conditions the boundary conditions can be written as flows:

$$Y_{i,0}^{n} = Y_{i,1}^{n}$$

$$Y_{i,N+1}^{n} = Y_{i,N}^{n}$$

$$Y_{0,j}^{n} = Y_{1,j}^{n}$$

$$Y_{M+1,j}^{n} = Y_{M,j}^{n}$$

Stability:

The stability criteria used was the same as the one above it was broken into both the parabolic and hyperbolic terms.

There are both parabolic and hyperbolic constraints as well the Reynold Cell. To simplify the problem the following is defined for the spatial step sizes $dy = dx = h$. The parabolic time constraints are defined with the following equation:

$$\frac{\nu \Delta t}{h^2} \leq \frac{1}{4}$$

Solving for Δt produces the following equation for the parabolic time step:

$$\Delta t = \frac{h^2}{4\nu}$$

The hyperbolic time step is defined using the following equation:

$$\frac{a\Delta t}{h} \leq 1$$

Solving for Δt yields the following equation for the hyperbolic criteria:

$$\Delta t = \frac{h}{(a)}$$

Where the variables a is based off the maximum value of both the *u* and *v* velocity vectors. The variable a is defined with the following equations:

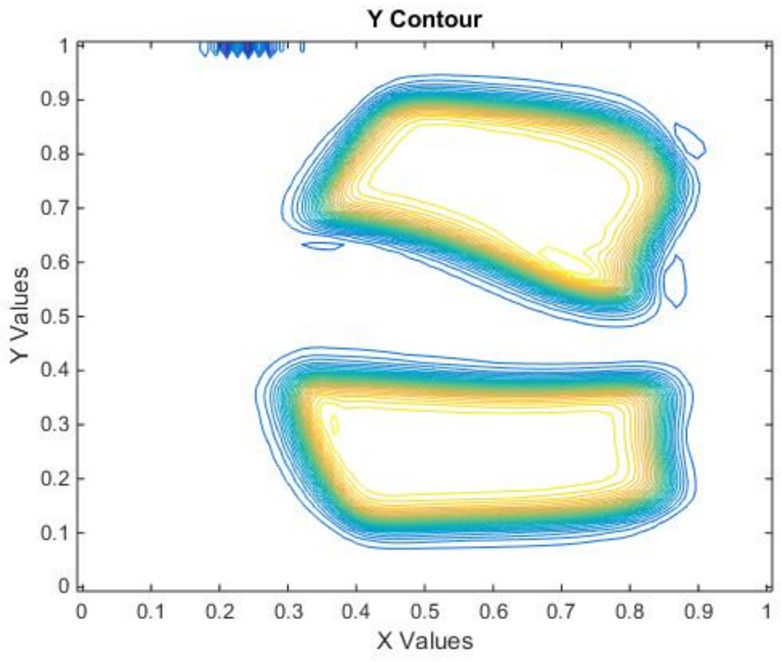
$$a = \max(Y)$$

The Reynolds Cell is calculated using the following equation:

$$\frac{(a)h}{\upsilon} \leq 2$$

<u>Solution:</u>

Unfortunately the solution was never obtained for the necessary intensity factors and was only able to be ran through a couple of iterations. The following plots were obtained after 100 iterations:



Appendix:

```
clear all
clc

tic

%%%%Givens
%Reynolds number
Re=200;
%viscosity
mu =1/Re;
%top wall velocity
A=1;
%bottom wall velocity
B=1/2;
%mesh size
M =196;
N =196;
%convergence for steady state
convtotal = 10^-5;

%%%Domain and the steps sizes precalculated
dx = 1/M;
dy = 1/N;
%used instead of diving
rec=1/dx;
rec2 = 1/dx^2;
recRe = 1/Re;
%%%%preallocating space and Initial Conditions
u = zeros(M+1,N+2);
v = zeros(M+2,N+1);
ustar = u;
vstar = v;
phi=zeros(M+2,N+2);
res2= phi;
rhsgs=phi;
res1=phi;
uplot = zeros(1,N+1);
vplot = zeros(1,N+1);

%initial time step
t = 0;
%gauss seidel convergance
n_gs_t = 0;
%arbiturary values for convergance
conv = 1;

%%%Boundary Conditions
%top wall
u(:,N+2) = 2*A-u(:,N+1);
%bottom wall
u(:,1) = 2*B-u(:,2);
%left wall
u(1,:) = 0;
%right wall
```

```
u(M+1,:) = 0;
%top wall
v(:,N+1) = 0;
%bottom wall
v(:,1) = 0;
%left wall
v(1,:) = -v(2,:);
%righ wall
v(M+2,:) = -v(M+1,:);
%counter
 n=0;
while conv>convtotal
    n=n+1;
    %finding the max ab and b values
    a = max(max(abs(u)));
    b = max(max(abs(v)));



    %first stability constraint, parabolic part
    dt1 = 0.25*dx^2/mu;

    %second stability constraint, hyperbolic part
    dt2 = dx/(a+b);

    %pick smallest dt
    if dt1 >= dt2
        dt = dt2;
    else
        dt = dt1;
    end



    %%%increase counters
    t = t+dt; %increase time by stable d


   %step 1 solving u and v star

    %%%%% store solution at n for convergence check
   uold = u;
    vold= v;

    %%%%% find solutions star using FTCS
    for j = 2:N+1
        for i = 2:M
            %averages because of staggered mesh
            %u(i+1,j)
            u1 = 0.5*(u(i+1,j)+u(i,j));
            %u(i-1,j)
            u2 = 0.5*(u(i,j)+u(i-1,j));
            %u(i,j+1/2)
            u3 = 0.5*(u(i,j+1)+u(i,j));
            %u(i,j-1/2)
            u4 = 0.5*(u(i,j-1)+u(i,j));
            %v(i+1/2,j)
            v1 = 0.5*(v(i+1,j)+v(i,j));
            %v(i+1/2,j-1/2)
            v2 = 0.5*(v(i+1,j-1)+v(i,j-1));

            %u-velocity star value
            ustar(i,j) = u(i,j)+dt*(-((u1^2-u2^2)*rec+(u3*v1-u4*v2)*rec)...
                +recRe*((u(i+1,j)-2*u(i,j)+u(i-1,j))*rec2+(u(i,j+1)-
2*u(i,j)...
                +u(i,j-1))*rec2));
        end
    end

    %%%%FTCS
    for j = 2:N
        for i = 2:M+1
            %averages because of staggered mesh
            %u(i,j+1/2)
            u3 = 0.5*(u(i,j+1)+u(i,j));
           %u(i-1,j+1/2)
            u5 = 0.5*(u(i-1,j)+u(i-1,j+1));
            %v(i+1/2,j+1/2)
            v1 = 0.5*(v(i+1,j)+v(i,j));
            %v(i,j+1)
            v3 = 0.5*(v(i,j+1)+v(i,j));
            %v*(i,j-1)
            v4 = 0.5*(v(i,j)+v(i,j-1));
            %v(i-1/2,j+1/2)
            v5 = 0.5*(v(i,j)+v(i-1,j));

            %%%solving vstar
            vstar(i,j) = v(i,j)+dt*(-((u3*v1-u5*v5)*rec+(v3^2-v4^2)*rec)...
                +recRe*((v(i+1,j)-2*v(i,j)+v(i-1,j))*rec2+(v(i,j+1)-
2*v(i,j)+...
                v(i,j-1))*rec2));
        end
    end

    %%%Boundary Conditions
    %ustar velocity
    %top wall
    ustar(:,N+2) = 2*A-ustar(:,N+1);
    %bottom wall
    ustar(:,1) = 2*B-ustar(:,2);
    %left wall
    ustar(1,:) = 0;
    %righ wall
    ustar(M+1,:) = 0;
    %vstar velocity
    %top wall
    vstar(:,N+1) = 0;
    %cottom wall
    vstar(:,1) = 0;
    %left wall
    vstar(1,:) = -vstar(2,:);
    %right wall
    vstar(M+2,:) = -vstar(M+1,:);

    %   Step II of Fractional Step Method

    %arbituary gauss convergance
    conv_gs = 1;
    %setting up the dynamic convergence
    if conv*10^-2 >= 10^-3
        conv_gs_limit = 10^-3;
    elseif conv*10^-2 <= 10^-5
        conv_gs_limit = 10^-5;
    end

 %solving the residual and right hand side foe computing speed
    for j = 2:N+1
        for i = 2:M+1
            %right hand side
            rhsgs(i,j) = 0.25*(dx/dt)*(ustar(i,j)-ustar(i-1,j)+vstar(i,j)-
vstar(i,j-1));
            %residual values
            res1(i,j) = (1/dt)*(((ustar(i,j)-ustar(i-1,j))*rec)+((vstar(i,j)-
vstar(i,j-1))*rec));
        end
    end
    nn=0;
    ngs=0;
    %using a awhile loop to solve for the Lagrange multiplier
    while conv_gs > conv_gs_limit &&ngs<1000
       ngs=ngs+1;
        nn=nn+1;

        %Gauss-Seidel
        for j = 2:N+1
            for i = 2:M+1
                phi(i,j) = 0.25*(phi(i-1,j)+phi(i+1,j)+phi(i,j-1)...
                    +phi(i,j+1))-rhsgs(i,j);
            end
        end

        %Boundary conditions for the lagrange multiplier
        phi(:,1) = phi(:,2);
        phi(:,N+2) = phi(:,N+1);
        phi(1,:) = phi(2,:);
        phi(M+2,:) = phi(M+1,:);


        %calcualting the residual
        if nn==100
        for j = 2:N+1
            for i = 2:M+1

                res2(i,j) = (phi(i+1,j)-2*phi(i,j)+phi(i-1,j))*rec2 ...
                +(phi(i,j+1)-2*phi(i,j)+phi(i,j-1))*rec2 ...
                -res1(i,j);
            end
        end
    %checking the residual
        conv_gs = max(max(abs(res2)));
        nn=0;
        end
    end

    %   Step 3 of Fractional Step Method

    %%%%% find u velocity solution at n+1
    for j = 2:N+1
        for i = 2:M
            u(i,j) = ustar(i,j)-(dt*rec)*(phi(i+1,j)-phi(i,j));
        end
    end

    %%%%% solving v n+1
    for j = 2:N
        for i = 2:M+1
            v(i,j) = vstar(i,j)-(dt*rec)*(phi(i,j+1)-phi(i,j));
        end
    end


    %u velocity
    %top wall
    u(:,N+2) = 2*A-u(:,N+1);
    %bottom wall
    u(:,1) = 2*B-u(:,2);
    %left wall
    u(1,:) = 0;
    %right walll
    u(M+1,:) = 0;
    %v velocity
    %top wall
    v(:,N+1) = 0;
    %bottom wall
    v(:,1) = 0;
    %left wall
    v(1,:) = -v(2,:);
    %right wall
    v(M+2,:) = -v(M+1,:);

    %%%%% convergence of final solution
    c1 = (u-uold)/dt;
    c2 = (v-vold)/dt;
    convu = max(max(abs(c1)));
    convv = max(max(abs(c2)));
    conv = max([convu convv]);

%outputting critcal information at certain points
if n==50
convdsi=conv
```

```
tdisp=t
n=0;
ngs
end


end

toc
%using the x values to find centerline values
x=linspace(0,1,M+2);
y=x;
%solving for the centerline velocities
for i=1:M+2
%vetical centerline
   uplot(i)=.5*(u(N/2,i)+u((N+2)/2,i));
   %horizontal line
   vplot(i)=.5*(v(i,N/2)+v(i,(N+2)/2));

end
%looping over the u and v to solve for voricity at the corners
for j = 1:N+1
    for i = 1:M+1
        omega(i,j) = rec*(v(i+1,j)-v(i,j)-u(i,j+1)+u(i,j));
    end
end
```

Plots:

```
%%%plotting the vorticity for 144x144 mesh
close all
load('Bonus_Problem_144.mat')
omega=Vortiscity(v,u,dx,M);
x=linspace(0,1,M+1);
y=x;
con = [0.0 -0.5 0.5 -1.0 1.0 -2.0 2.0 -3.0 3.0 -4.0 -5.0];
figure
contour(x,y,omega',con)
title('Vorticity Contour for 144x144 Mesh')
xlabel('X Values')
ylabel('Y Values')
%%
%%%plotting the streamline function for 144x144 mesh
close all
load('Bonus_Problem_144.mat')
psi=zeros(M+2,N+2);
x=linspace(0,1,M+2);
y=x;
psi=Streamline(psi,omega,M,dx);
figure
contour(x,y,psi',30)
title('Streamline Contour for 144x144 Mesh')
xlabel('X Values')
ylabel('Y Values')
```

```
%%
%%%%plotting the u vs vertical line
close all
load('Bonus_Problem_144.mat')
x=linspace(0,1,M+2);
y=x;
plot(x,uplot,'r')
title('U Velocity at x=.5 with 144x144 Mesh')
xlabel('Y Values')
ylabel('U Velocity')

%%
%%%%plotting the v vs vertical line
close all
load('Bonus_Problem_144.mat')
x=linspace(0,1,M+2);
y=x;
plot(x,vplot,'r')
title('V Velocity at x=.5 with 144x144 Mesh')
xlabel('X Values')
ylabel('v Velocity')
```

Streamline:

```
function [psi] =Streamline(psi,omega,M,dx)
%This function solve for the streamline function
%definign the domain
N=M;
%prealloactin space
res=zeros(M+2,M+2);
%Boundary conditions for psi
psi(1,:)=0;
psi(M+2,:)=0;
psi(:,1)=0;
psi(:,N+2)=0;
%arbituary convergence
conv1=1;
%convergence for the poisson equation
convgs=10^-5;
%while loop to solve for the streamline
while conv1>convgs

for j=2:M+1
    for i=2:M+1
        %gauss seidel to solve for stream line

        psi(i,j)=.25*(psi(i+1,j)+psi(i-1,j)+psi(i,j+1)+psi(i,j-
1)+dx^2*omega(i,j));

    end
end

%calculating the residual
for j=2:M+1
    for i=2:M+1
```

```
       res(i,j)=(psi(i+1,j)-2*psi(i,j)+psi(i-1,j)+psi(i,j+1)-
2*psi(i,j)+psi(i,j-1))/dx^2+omega(i,j);


    end
end


%checking the convergence
conv1=max(max(abs(res)));




end

end
```

Extra Credit:

```
clc
clear
close all
%inital conditions as well as x and y
m=64;
dx=1/m;
re=200;
b=.5;
a=1;
x=0-.5*dx:dx:1+.5*dx;
vis=1/re;
%defining max dt for parabolic
dt=.5*(.25*dx^2)/(1/re);
%Preallocationg space
phi=zeros(m+2,m+2);
u=zeros(m+1,m+2);
v=zeros(m+2,m+1);
ustar=zeros(m+1,m+2);
vstar=zeros(m+2,m+1);
%setting constans for ftcs
z=dt/dx;
y=dt/(re*dx^2);

%boundary conditions
%bottom
u(:,1)=2*b-u(:,2);
v(:,1)=0;
%top
u(:,m+2)=2*a-u(:,m+1);
v(:,m+1)=0;
%left
u(1,:)=0;
v(1,:)=2*0-v(2,:);
```

```
%right
u(m+1,:)=0;
v(m+2,:)=2*0-v(m+1,:);
%setting inital time
t=0;
convt=1;
%using a whole loop to run until steady state is reached.
while convt>10e-4
    %solving for ustar using the ftcs method
    for j=2:m+1
        for i=2:m
            u1=.5*(u(i+1,j)+u(i,j));
            u2=.5*(u(i,j)+u(i-1,j));
            u3=.5*(u(i,j+1)+u(i,j));
            u4=.5*(u(i,j)+u(i,j-1));
            v1=.5*(v(i+1,j)+v(i,j));
            v2=.5*(v(i,j-1)+v(i+1,j-1));
            ustar(i,j)=u(i,j)-z*(u1^2-u2^2+(u3*v1)-(u4*v2))+y*(u(i+1,j)-
2*u(i,j)+u(i-1,j)+u(i,j+1)-2*u(i,j)+u(i,j-1));
        end
    end
    %solving for vstar using the ftcs method
    for j=2:m
        for i=2:m+1
            u1=.5*(u(i,j+1)+u(i,j));
            u2=.5*(u(i-1,j)+u(i-1,j+1));
            v1=.5*(v(i,j)+v(i+1,j));
            v2=.5*(v(i,j)+v(i-1,j));
            v3=.5*(v(i,j+1)+v(i,j));
            v4=.5*(v(i,j-1)+v(i,j));
            vstar(i,j)=v(i,j)-(z*(u1*v1-v2*u2+v3^2-v4^2))+(y*(v(i+1,j)-
2*v(i,j)+v(i-1,j)+v(i,j+1)-2*v(i,j)+v(i,j-1)));
        end
    end
    %redefing boundary conditions
    %bottom
    ustar(:,1)=2*b-ustar(:,2);
    vstar(:,1)=0;
    %top
    ustar(:,m+2)=2*a-ustar(:,m+1);
    vstar(:,m+1)=0;
    %left
    ustar(1,:)=0;
    vstar(1,:)=2*0-vstar(2,:);
    %right
    ustar(m+1,:)=0;
    vstar(m+2,:)=2*0-vstar(m+1,:);
    conv=1;
    convstore=[];
    while conv>10e-4
        for j=2:m+1
            for i=2:m+1
                %solving for phi using gauss
                phi(i,j)=.25*(phi(i+1,j)+phi(i-1,j)+phi(i,j+1)+phi(i,j-1)+...
                  (dx/dt*(ustar(i,j)-ustar(i-1,j)+vstar(i,j)-vstar(i,j-1))));
            end
        end
```

```
                phi(1,:)=phi(2,:);
                phi(m+2,:)=phi(m+1,:);
                phi(:,1)=phi(:,2);
                phi(:,m+2)=phi(:,m+1);

        for j=2:m+1
            for i=2:m+1
                %calculating the res
                res(i,j)=phi(i+1,j)-2*phi(i,j)+phi(i-1,j)+...
                         phi(i,j+1)-2*phi(i,j)+phi(i,j-1)-...
                         ((dx/dt)*((ustar(i,j)-ustar(i-1,j))+(vstar(i,j)-
vstar(i,j-1))));
            end
        end
                %finding the max residual value
                conv=max(max(abs(res)))
                convstore=[convstore conv];

    end

    %recalculating u and v
    for j=2:m+1
        for i=2:m
            u(i,j)=ustar(i,j)-(dt*((phi(i+1,j)-phi(i,j))/dx));
        end
    end

    for j=2:m
        for i=2:m+1
            v(i,j)=vstar(i,j)-(dt*((phi(i,j+1)-phi(i,j))/dx));
        end
    end
    %storing u and v values
    uold=u;
    vold=v;
    convu=(u-uold)/dt;
    convv=(v-vold)/dt;
    convt=max(max(abs([convu,convv])));
    %reapplying the u and v boundary conditions
    %bottom
    u(:,1)=2*b-u(:,2);
    v(:,1)=0;
    %top
    u(:,m+2)=2*a-u(:,m+1);
    v(:,m+1)=0;
    %left
    u(1,:)=0;
    v(1,:)=2*0-v(2,:);
    %right
    u(m+1,:)=0;
    v(m+2,:)=2*0-v(m+1,:);
    t=t+dt;
    %finding the max u and v
    A=max(max(abs(u)));
    B=max(max(abs(v)));
    %finding dt for the parabolic and hyperbolic parts
    dt1=dt;
    dt2=dx/(A+B);
    %using a for loop to determine the next dt
if dt1>dt2
    dt=dt2;
elseif dt1<dt2
    dt=dt1;
end
end
umid(i,j)=.5*(u(m/2+1,(m+2)/2)+u(m/2+1,((m+2)/2)+1));
vmid(i,j)=.5*(v((m+2)/2,m/2+1)+v(((m+2)/2)+1,m/2));
```

GCI:

```
 %courses grid max
f3=.0566;
%second finest grid
f2=.05675;
%finest grid
f1=.0568;
%ratio number
r=1.5;
%solving p from the power point
p = log((f3-f2)/(f2-f1))/log(r);
%fing f(h=0)
f0 = f1+((f1-f2)/(r^p-1));
 %solving the GCI
GCI12 = 1.25*(abs((f1-f2)/f1)/(r^p-1));
GCI23 = 1.25*(abs((f2-f3)/f2)/(r^p-1));
 %storing the solutions to plot
fplot= [f1 f2 f3]; %FTCS method solutions

figure

plot([1 2 4],fplot,'-or',0,f0,'d');

title('Accuracy Analysis of U Velocity');
ylabel('Velocity at Steady State');
xlabel('Grid Spacing')
legend('FTCS Method','Richardson Extrapolation')
```