INCOMPRESSIBLE NAVIER STOKES EQUATIONS

Computational Fluid Dynamics I: Driven Cavity Flow

Gandharv Kashinath

University of Michigan, Ann Arbor, MI

12/8/2008

In this project a Driven Cavity Problem was solved which was governed by the Incompressible Navier Stokes Equations. A finite-element based on the Projection Method algorithm was developed and tested with published data. The project also involved simulation of high Re flows which were also compared and evaluated.

Introduction

In this project an attempt was made to solve the two-dimensional driven cavity problem. The geometry of the problem is shown in Figure 1. The flow in this problem is described by the incompressible Navier-Stokes equations and in order to solve these equations an algorithm was developed and implemented using FORTRAN.

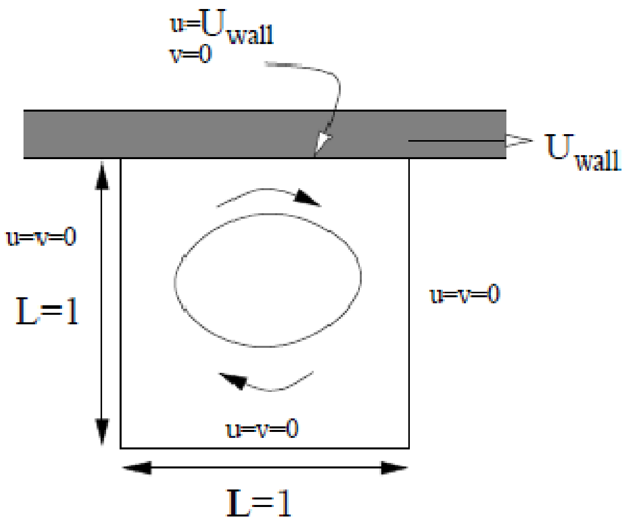


Figure 1: Domain for the driven cavity problem including the Boundary conditions.

In order to solve this problem a Primitive Variable Approach namely, the Pressure Correction method (Projection Method) was employed and the schemes developed during the project were chosen to be second-order accurate (higher order with the limiter) in space and first order accurate in time. The solutions thus obtained were validated using published articles and by simulations run in the commercial software, Fluent. Specific cases for Reynolds' number, Re= 1, 10, 100 were presented as prescribed in the project description and higher Reynolds' number solutions were also attempted and solved. Presented below is a brief description of the problem formulation and solution techniques employed during the course of the project, followed by a presentation of the results and a brief discussion in support of these solutions.

Problem Formulation

1.1 Conservation Equation

The driven cavity problem is governed by the Incompressible Navier Stokes (N-S) Equations. The 2D N-S equations can be written in conservative form, assuming ρ=1:

$$u_t+(F+p)_x+H_y^x=0$$

$$v_t+H_x^y+(G+p)_y=0$$

where,

$$F = u^2 - \nu u_x, \quad G = v^2 - \nu v_y, \quad H^x = uv - \nu u_y, \quad H^y = uv - \nu v_x$$

The fluxes F and G are stored at the cell centers, while H^x, H^y are stored at the grid nodes dues to a staggered storage setup as illustrated in the following section.

1.2 Staggered Storage

Storing each of the primitive variables u, v, p at the nodes of a regular grid ("collocated storage") makes straightforward discretization susceptible to spurious solutions. Hence in order to address this problem, a staggered storage was used to guarantee discrete well-posedness (Figure 2):

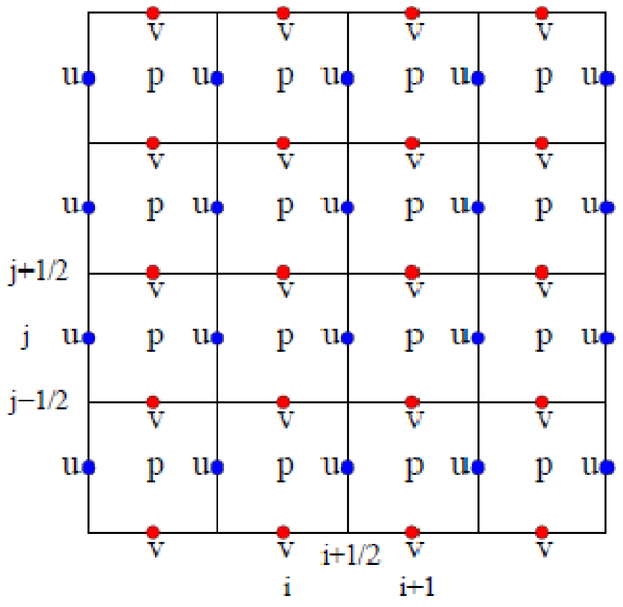


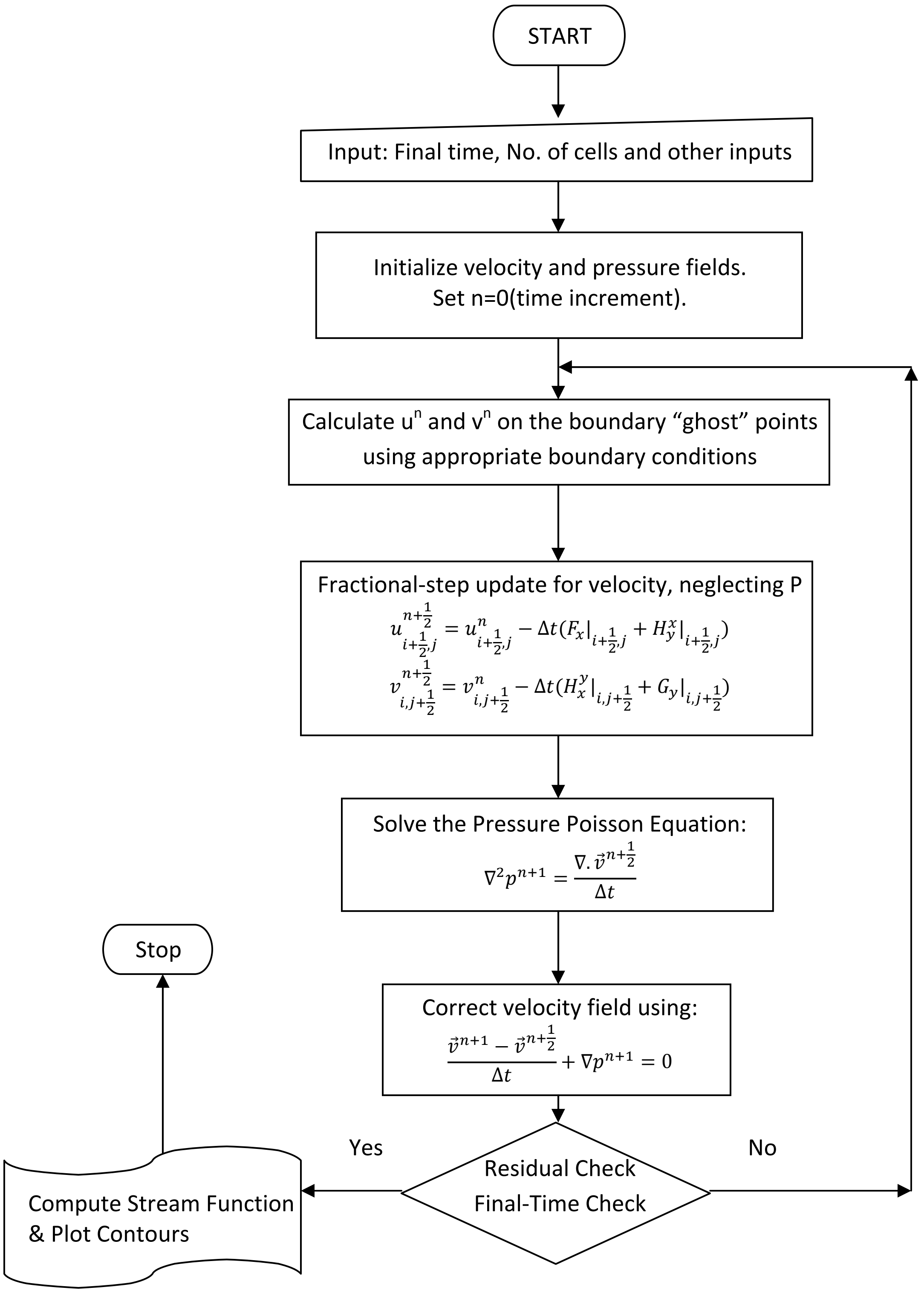
Figure 2: Staggered storage for primitive variables.

The cells were indexed with i,j, with half-integer indices for the grid lines, so that:

- The x-velocity $u_{i+1/2,j}$ was stored at the midpoint of every vertical edge.
- The y-velocity $v_{i,j+1/2}$ was stored at the midpoint of every horizontal edge.
- The pressure $p_{i,j}$ was stored in the middle of each cell.

1.3 Solution Algorithm

In order to implement the projection-based primitive-variable solution technique the following algorithm was constructed and implemented;


START
Input: Final time, No. of cells and other inputs
Initialize velocity and pressure fields.
Set n=0(time increment).
Calculate u^n and v^n on the boundary "ghost" points using appropriate boundary conditions
Fractional-step update for velocity, neglecting P
$u^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = u^n_{i+\frac{1}{2},j} - \Delta t(F_x|_{i+\frac{1}{2},j} + H^x_y|_{i+\frac{1}{2},j})$
$v^{n+\frac{1}{2}}_{i,j+\frac{1}{2}} = v^n_{i,j+\frac{1}{2}} - \Delta t(H^y_x|_{i,j+\frac{1}{2}} + G_y|_{i,j+\frac{1}{2}})$
Solve the Pressure Poisson Equation:
$\nabla^2 p^{n+1} = \frac{\nabla.\vec{v}^{n+\frac{1}{2}}}{\Delta t}$
Correct velocity field using:
$\frac{\vec{v}^{n+1} - \vec{v}^{n+\frac{1}{2}}}{\Delta t} + \nabla p^{n+1} = 0$
Residual Check
Final-Time Check
Yes
No
Compute Stream Function & Plot Contours
Stop

1.4 Fractional-Step Velocity Update

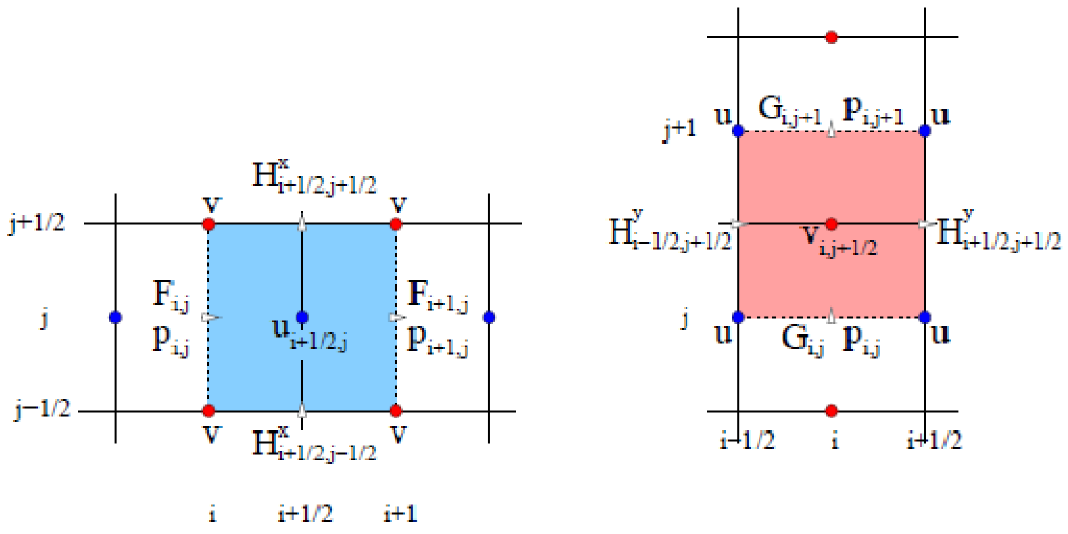


Figure 3: Flux, Velocity and Pressure locations.

In order to perform the fractional step update a forward Euler time discretization was considered and the pressure contribution was neglected. The finite-difference formula for this procedure was as follows;

$$u^{n+\frac{1}{2}}_{i+\frac{1}{2},j} = u^n_{i+\frac{1}{2},j} - \Delta t(F_x|_{i+\frac{1}{2},j} + H^x_y|_{i+\frac{1}{2},j})$$

$$v^{n+\frac{1}{2}}_{i,j+\frac{1}{2}} = v^n_{i,j+\frac{1}{2}} - \Delta t(H^y_x|_{i,j+\frac{1}{2}} + G_y|_{i,j+\frac{1}{2}})$$

In order to update the fractional velocity fields the Fluxes at corresponding locations were required and in order to compute these, a second-order accurate central difference scheme was employed as follows (h=Δx=Δy);

$$F_x|_{i+\frac{1}{2},j} = \frac{1}{h}(F_{i+1,j} - F_{i,j}) \stackrel{\text{def}}{=} \frac{1}{h}\delta_x|_{i+\frac{1}{2},j}F$$

$$H^x_y|_{i+\frac{1}{2},j} = \frac{1}{h}(H^x_{i+\frac{1}{2},j+\frac{1}{2}} - H^x_{i+\frac{1}{2},j-\frac{1}{2}}) \stackrel{\text{def}}{=} \frac{1}{h}\delta_y|_{i+\frac{1}{2},j}H^x$$

$$G_y|_{i,j+\frac{1}{2}} = \frac{1}{h}(G_{i,j+1} - G_{i,j}) \stackrel{\text{def}}{=} \frac{1}{h}\delta_y|_{i,j+\frac{1}{2}}G$$

$$H^y_x|_{i,j+\frac{1}{2}} = \frac{1}{h}(H^y_{i+\frac{1}{2},j+\frac{1}{2}} - H^y_{i-\frac{1}{2},j+\frac{1}{2}}) \stackrel{\text{def}}{=} \frac{1}{h}\delta_x|_{i,j+\frac{1}{2}}H^y$$

where δ_x and δ_y are the difference operators centered at the specific locations. The expressions for updating $v_{i,j+1/2}$ were similar.

1.5 Pressure Equation

The pressure correction step was used to make the velocity field at n+1 divergence free. The pressure equation at i, j was discretized using a central differencing scheme;

$$\delta_x^2|_{i,j}p^{n+1} + \delta_y^2|_{i,j}p^{n+1} = \frac{h}{\Delta t}\left(\delta_x|_{i,j}u^{n+\frac{1}{2}} + \delta_y|_{i,j}v^{n+\frac{1}{2}}\right)$$

where

$$\delta_x^2|_{i,j}p^{n+1} = p_{i-1,j}^{n+1} - 2p_{i,j}^{n+1} + p_{i+1,j}^{n+1}$$

$$\delta_y^2|_{i,j}p^{n+1} = p_{i,j-1}^{n+1} - 2p_{i,j}^{n+1} + p_{i,j+1}^{n+1}$$

The solution to the Pressure Poisson Equation was the most expensive part of the program and the solution algorithm for this equation is discussed in detail in the section 1.8.

1.6 Boundary Conditions

At a wall, the no slip condition required the tangential fluid velocity to match the wall velocity and the zero-flow through required the normal fluid velocity to be zero. Specifically, in the case of staggered storage this meant that $H^x = uv - \nu u_y = 0$ on vertical walls and $H^y = uv - \nu v_x = 0$ on horizontal boundaries. However, a pressure boundary condition was required which meant that the H^x and H^y needed to be evaluated at the horizontal and vertical boundaries respectively.

The method employed to evaluate these fluxes was called the ghost-point method, in which the fluid was assumed to extend through the wall, but in such a way that the required boundary conditions were satisfied.

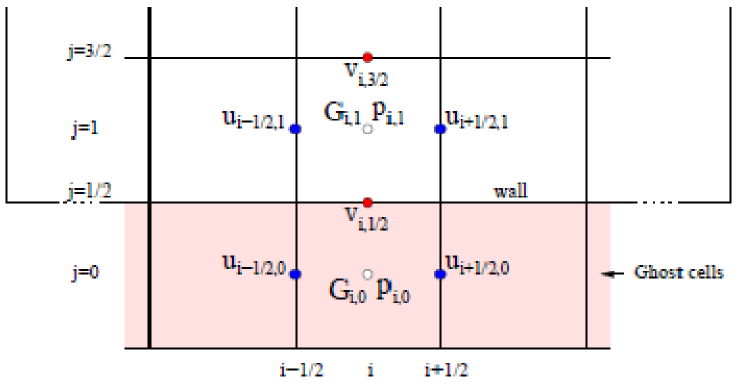


Figure 4: Ghost cell Layer for horizontal boundary used in Ghost-Point Method.

On the horizontal wall, at y=0, u_x=0, so by continuity v_y=-u_x=0. That meant that v was an even function of y and its derivative v_y=-u_x (and hence u) was an odd function of y.

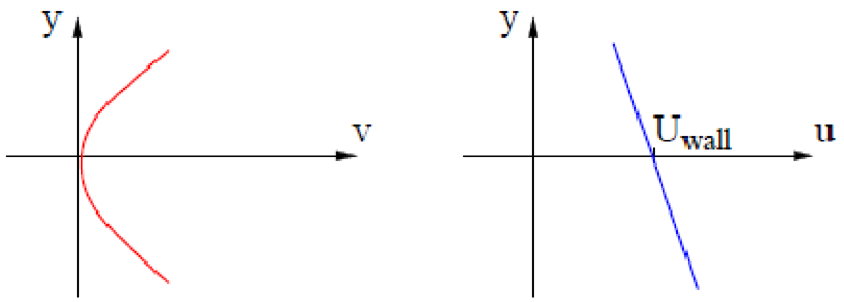


Figure 5: Velocity functions; odd and even functions.

Thus, near the wall $\boxed{v = \frac{1}{2}v_{yy}y^2 + O(y^3), u = U_{wall} + u_y y + O(y^2)}$ where v_{yy} and u_y were evaluated at y=0.

The velocity $u_{i+1/2,0}$ was required for computing u_y in the viscous component of H^x at y=0. Since, from above, u was an odd function of y.

$$u_{i+\frac{1}{2},1} - U_{wall} = U_{wall} - u_{i+\frac{1}{2},0} + O(h^2)$$

$$u_{i+\frac{1}{2},0} = 2U_{wall} - u_{i+\frac{1}{2},1} + O(h^2)$$

From the pressure correction equation, the pressure boundary condition was of Neumann type. This was discretized using central difference:

$$\frac{p_{i,1}^{n+1} - p_{i,0}^{n+1}}{h} = \frac{v_{i,\frac{1}{2}}^{n+\frac{1}{2}} - v_{i,\frac{1}{2}}^{n+1}}{\Delta t}$$

For zero flow normal to the wall, this reduced to

$$p_{i,0}^{n+1} = p_{i,1}^{n+1}$$

This condition specified the pressure just outside the domain boundary and this was used in an iterative solution technique namely the Successive Over Relaxation (SOR) algorithm. The boundary condition for the other walls was derived similarly and is summarized below;

- At $y = 0, u_{i+\frac{1}{2},0} = -u_{i+\frac{1}{2},1}$ $(U_{wall} = 0)$
- At $x = 0, v_{0,j+\frac{1}{2}} = -v_{1,j+\frac{1}{2}}$ $(V_{wall} = 0)$
- At $y = 1, u_{i+\frac{1}{2},N+1} = 2U_{wall} - u_{i+\frac{1}{2},N}$ $(U_{wall} = 1)$
- At $x = 1, v_{N+1,j+\frac{1}{2}} = -v_{N,j+\frac{1}{2}}$ $(V_{wall} = 0)$

1.7 Flux Evaluation

The flux definitions were as follows;

$$F = u^2 - \nu u_x, \quad G = v^2 - \nu v_y, \quad H^x = uv - \nu u_y, \quad H^y = uv - \nu v_x$$

In order to compute the flux it was observed that a simple central differencing for all the terms yielded a method similar to the FTCS method. It was observed that this scheme resulted in unphysical oscillations for any reasonable Reynolds' number. Hence the Quadratic Upstream Interpolation for Convection Kinematics (QUICK) scheme was employed.

A closer look at the inviscid fluxes revealed;

- F : uu = transport of u in the x-direction
- G: vv = transport of v in the y-direction
- H^x: vu = transport of u in the y-direction
- H^y: uv = transport of v in the x-direction

All these were of the form: flux=qϕ, where q was the transport velocity (u when in the x-direction and v when in the y direction) and ϕ was the transported quantity. Thus with the QUICK flux approach:

- q was evaluated by central differencing
- ϕ was computed using Upwind scheme depending on the sign of q.

For example, the convective component u^2 for $F_{i,j}$,

- To evaluate $q_{i,j}$;

$$q_{i,j} = \left(\frac{u_{i-\frac{1}{2},j} + u_{i+\frac{1}{2},j}}{2}\right)$$

- If $q_{i,j} > 0$,

$$\phi_{i,j} = \frac{3u_{i+\frac{1}{2},j} + 6u_{i-\frac{1}{2},j} - u_{i-\frac{3}{2},j}}{8}$$

- If $q_{i,j} \leq 0$,

$$\phi_{i,j} = \frac{3u_{i-\frac{1}{2},j} + 6u_{i+\frac{1}{2},j} - u_{i+\frac{3}{2},j}}{8}$$

The above method was implemented similarly to the remaining fluxes and the flux equations including the viscous terms were summarized as follows;

$$F_{i,j} = q_{i,j}(u)\phi_{i,j}(u) - \frac{\nu}{h}\delta_x u, \quad G_{i,j} = q_{i,j}(v)\phi_{i,j}(v) - \frac{\nu}{h}\delta_y v$$
$$H^x_{i+\frac{1}{2},j+\frac{1}{2}} = q_{i,j}(v)\phi_{i,j}(u) - \frac{\nu}{h}\delta_y|_{i+\frac{1}{2},j+\frac{1}{2}}u, \quad H^y_{i+\frac{1}{2},j+\frac{1}{2}} = q_{i,j}(u)\phi_{i,j}(v) - \frac{\nu}{h}\delta_x|_{i+\frac{1}{2},j+\frac{1}{2}}v$$

1.8 Limiting for Higher Reynolds Number

The QUICK method was a great improvement over simple upwinding, but still suffered from oscillations at high Reynolds' numbers, especially for data that was nearly discontinuous. This problem was dealt with using a limiter. Suppose for three consecutive data values on a regular grid: ϕ_{j-1}, ϕ_j, ϕ_{j+1}, it was required to evaluate $\phi_{j+1/2}$ (i.e. upwinding with $q_{j+1/2} > 0$).

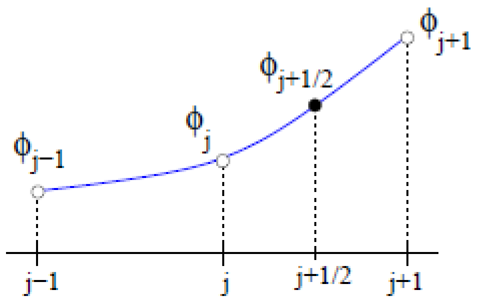


Figure 6: Sample Stencil for limiter application.

A smoothness monitor was defined as follows,

$$\hat{\phi}_j = \frac{\phi_j - \phi_{j-1}}{\phi_{j+1} - \phi_{j-1}}$$

$\hat{\phi}_j = 1/2$ denoted a smooth data, while $\hat{\phi}_j$ outside [0,1] denoted non-monotone data. A limiter was defined such that $\hat{\phi}_j$ was the input and;

$$\hat{\phi}_{j+\frac{1}{2}} = \frac{\phi_{j+\frac{1}{2}} - \phi_{j-1}}{\phi_{j+1} - \phi_{j-1}}$$

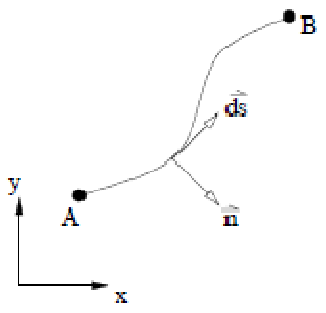
In order to compute $\hat{\phi}_{j+\frac{1}{2}}$ the SMART (Simple Monotone for Realistic Transport) limiter was used, which was defined as follows;

$$\hat{\phi}_{j+\frac{1}{2}} = \begin{cases} \hat{\phi}_j\,, & \hat{\phi}_j < 0 \; or \; \hat{\phi}_j > 0 \\ 3\hat{\phi}_j, & 0 < \hat{\phi}_j < \frac{1}{6} \\ \frac{3}{8}(2\hat{\phi}_j + 1), & \frac{1}{6} < \hat{\phi}_j < \frac{5}{6} \\ 1, & \frac{5}{6} < \hat{\phi}_j < 1 \end{cases}$$

1.9 Postprocessing

In order to visualize the streamlines on a regular staggered grid the stream function ψ was used which was defined by;

$$\frac{\partial \psi}{\partial y} = u, \qquad \frac{\partial \psi}{\partial x} = -v$$



Integrating the streamfunction on an arbitrary path between points A and B in (x,y);

$$\psi(B) - \psi(A) = \int_A^B \nabla\psi . \overrightarrow{ds}$$

$$= \int_A^B \frac{\partial \psi}{\partial x} ds_x + \frac{\partial \psi}{\partial y} ds_y$$

$$= \int_A^B \left(-\frac{\partial \psi}{\partial x} n_y + \frac{\partial \psi}{\partial y} n_x\right) ds$$

$$= \int_A^B (v n_y + u n_x)\, ds$$

$$= \int_A^B \vec{v}.\vec{n}\, ds$$

where $\vec{n} = [n_x, n_y]$ was the normal to the path, and $\overrightarrow{ds} = [ds_x ds_y] = [-n_y, n_x]ds$ was the tangent to the path. It was concluded from the above expression that the volume flow rate through any path joining points A and B would be the same, and equal to $\psi(B) - \psi(A)$.

This property of the stream function was utilized to evaluate it discretely. First a corner A (point (1,1) on the grid), was picked and set at ψ=0. Further, the ψ was calculated at any other point on the grid using a combination of vertical and horizontal paths. Care was taken to keep the signs of the velocity in check by determining the direction of the normal. Once the ψ was obtained at all points it was visualized using contours plotted using Tecplot.

Programming Issues & Solutions

2.0 Pressure Poisson Equation (PPE) Solver

In order to solve the pressure Poisson equation two algorithms were implemented;

- Gaussian Elimination
- Successive Over Relaxation (SOR)

Since the Boundary conditions for the PPE were all Neumann type, the pressure was only defined upto a constant. This generated a singular matrix during the implementation of the Gaussian Elimination type solution. To address the issue of singularity a value for pressure at one of the cells (say p (1, 1)=0) was set to zero. By doing so the matrix singularity was removed from the system of equations and thus the Gaussian Elimination was then conveniently applied to solve the system of linear equations.

Further, another convenient method called the Successive Over Relaxation (SOR) was employed to solve the PPE iteratively (1). This method is based on the Gauss-Seidel algorithm which is modified by the relaxation factor to obtain the SOR method. Applying the PPE to the Gauss-Seidel method for the Laplace operated, leads to the following iterative scheme (1);

$$\overline{P_{i,j}^{n+1}} = \frac{1}{4}\left(P_{i+1,j}^{n} + P_{i-1,j}^{n+1} + P_{i,j+1}^{n} + P_{i,j-1}^{n+1}\right) - \frac{h}{\Delta t}\left(\delta_x|_{i,j}u^{n+\frac{1}{2}} + \delta_y|_{i,j}v^{n+\frac{1}{2}}\right)$$

$$P_{i,j}^{n+1} = \omega\overline{P_{i,j}^{n+1}} + (1-\omega)P_{i,j}^{n}$$

where ω was the relaxation factor. The Neumann Boundary condition was accommodated in the above formulation by modifying the equation and presented below is a sample boundary equation for $\overline{P_{1,1}^{n+1}}$ and $\overline{P_{1,j}^{n+1}}$;

$$\overline{P_{1,1}^{n+1}} = \frac{1}{2}\left(P_{2,1}^{n} + P_{1,2}^{n}\right) - \frac{h}{\Delta t}\left(\delta_x|_{1,1}u^{n+\frac{1}{2}} + \delta_y|_{1,1}v^{n+\frac{1}{2}}\right)$$

$$\overline{P_{1,j}^{n+1}} = \frac{1}{3}\left(P_{1,j-1}^{n+1} + P_{2,j}^{n} + P_{1,j+1}^{n+1}\right) - \frac{h}{\Delta t}\left(\delta_x|_{1,j}u^{n+\frac{1}{2}} + \delta_y|_{1,j}v^{n+\frac{1}{2}}\right)$$

Similarly the $\overline{P_{i,j}^{n+1}}$, for other boundaries and corner points were derived and enforced during the SOR iteration loop. Further, during the implementation of the SOR iterative procedure it was critical to choose an optimum ω to solve the system and hence the following equation obtained from Tannehill (2), was used

$$\sigma = \frac{1}{1+\beta^2}\left(\cos\frac{\pi}{p} + \beta^2\cos\frac{\pi}{q}\right)$$

$$\omega_{opt} = \frac{2}{1+(1-\sigma^2)^{1/2}}$$

where, β is the grid aspect ratio, $\beta = \frac{\Delta x}{\Delta y}$ and p is the number of Δx increments, and q is the number of Δy increments along the sides of the rectangular region (2).

2.1 Convergence

In order to check for the convergence a residual was calculated using the governing equation and neglecting the time-derivative term;

$$u_t + (F+p)_x + H_y^x = 0$$

$$(F+p)_x + H_y^x \stackrel{\text{def}}{=} Residual \cong 0$$

The residual was calculated after the solution from the PPE was obtained. Also the change in velocity at each time-step was monitored during the iteration.

2.2 Time Step Calculation

In order to calculate a suitable time-step, Δt, the recommended equation as presented below was used (3);

$$\Delta t = min\left(\frac{h^2}{4\nu}, \frac{4\nu}{U_{wall}^2}\right)$$

This time step calculation was suitable in most cases but had to be modified at higher Reynolds' numbers.

Results and Discussion

In order to validate the code the results presented in Hirsch (1) were reproduced. The results of these runs were as follows;

3.0 Validation Case (Mesh 41x41, Re=100, ρ=1, μ=0.01)

a) Velocity Plots:

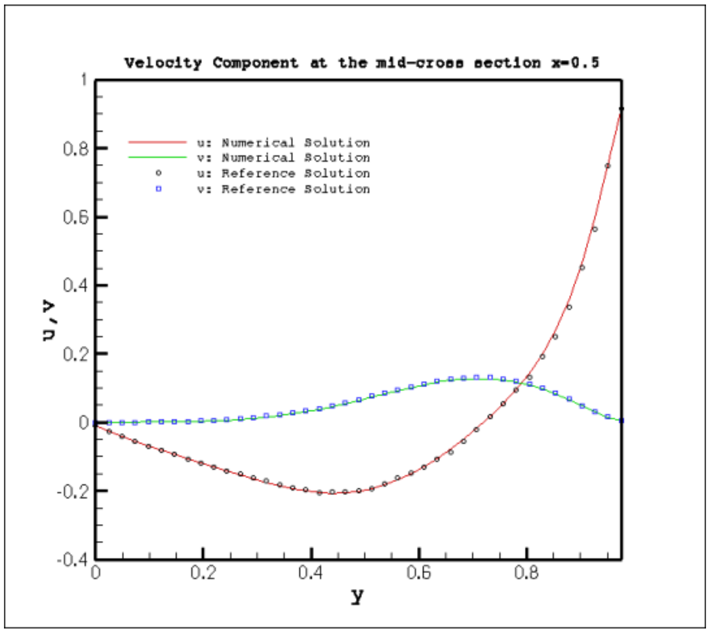


Figure 7: Comparison of the computed velocity distribution along the centerline x=0.5 (1).

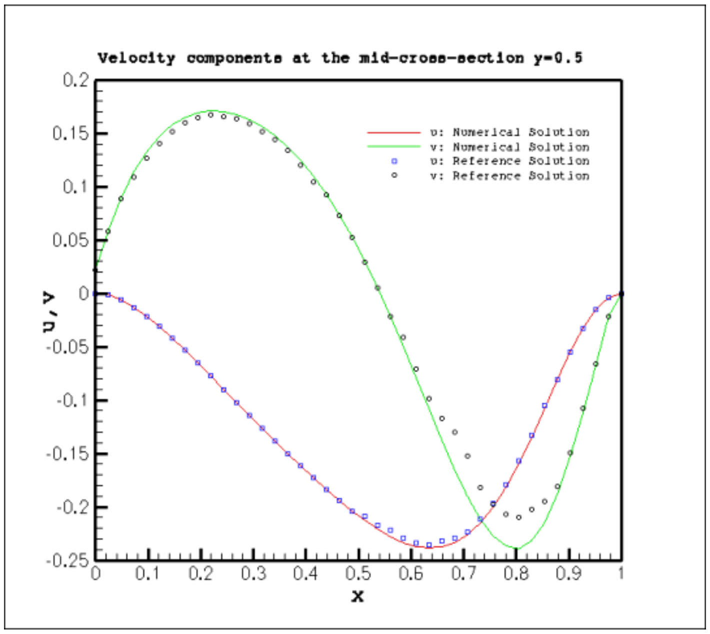
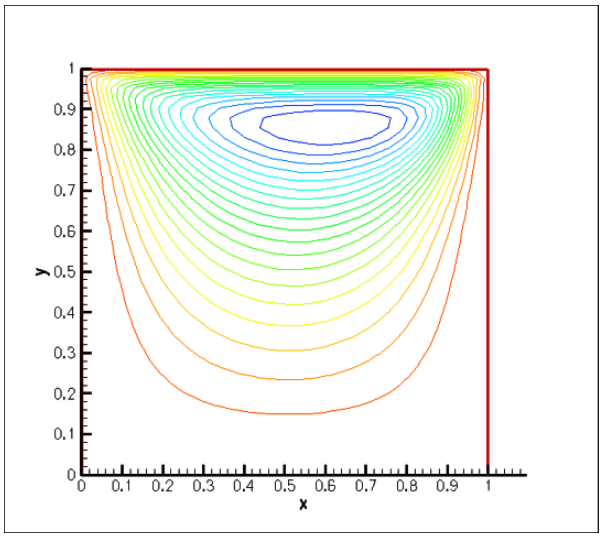
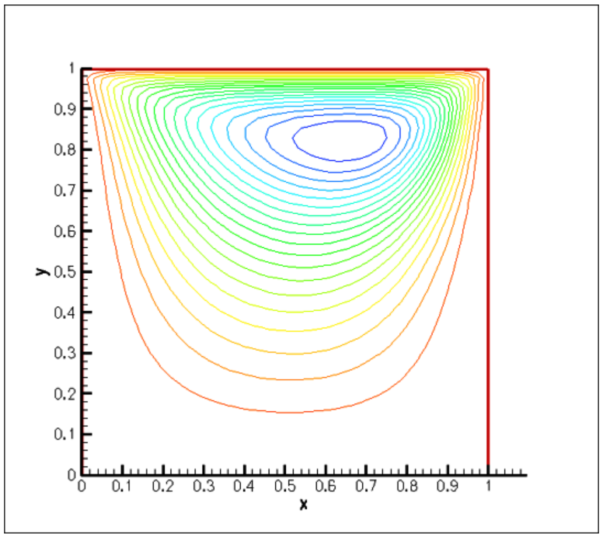


Figure 8: Comparison of the computed velocity distribution along the centerline y=0.5 (1).

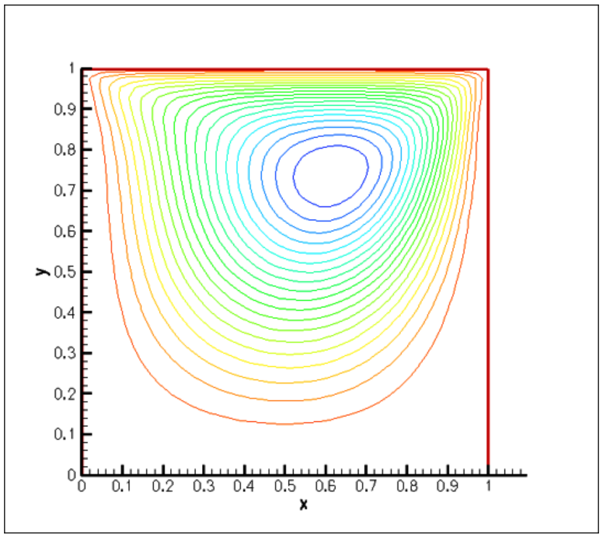
b) Streamfunction plots:



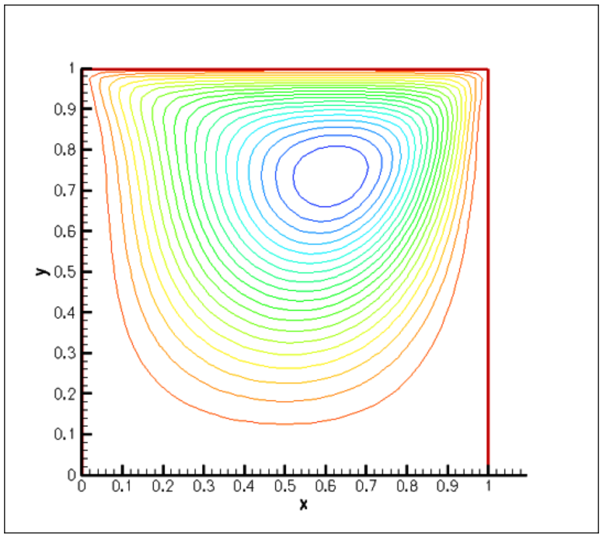
(a) Transient Solution at t=0.5



(b) Transient Solution at t=1



(c) Solution at t=10



(d) Solution at t=30

Figure 9: Streamlines of the flow field at different transient stages, for t=0.5, 1, 10, and 30 (Comparison Case (1)).

3.1 Prescribed Cases

a) Streamfunction Plots:

- Reynolds' Number, Re=1.

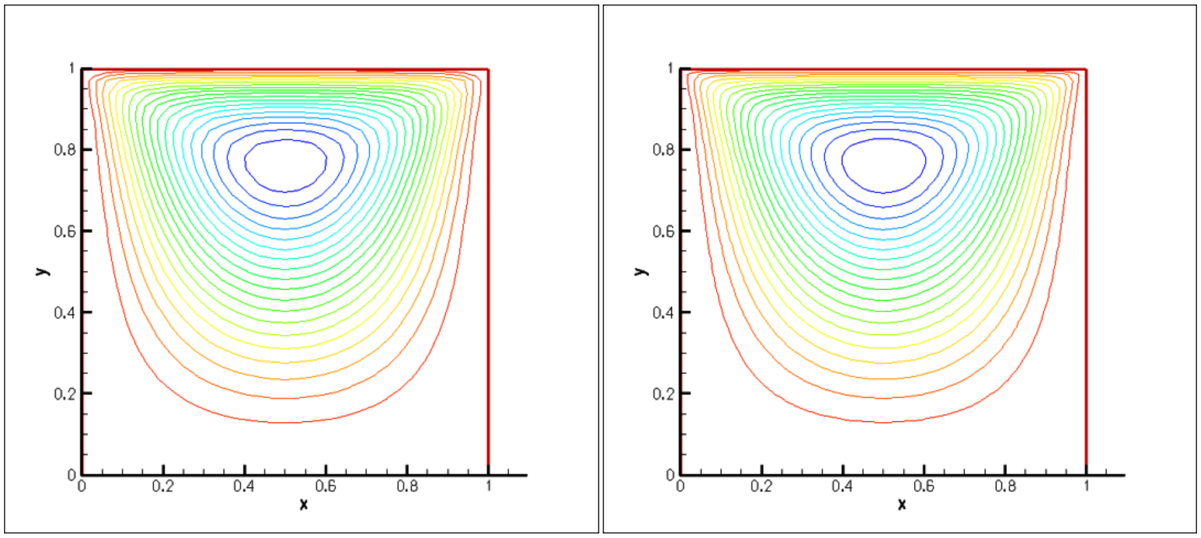


Figure 10: Comparison of a coarse (32x32) and fine (64x64) mesh solution for Re=1.

- Reynolds' Number, Re=10

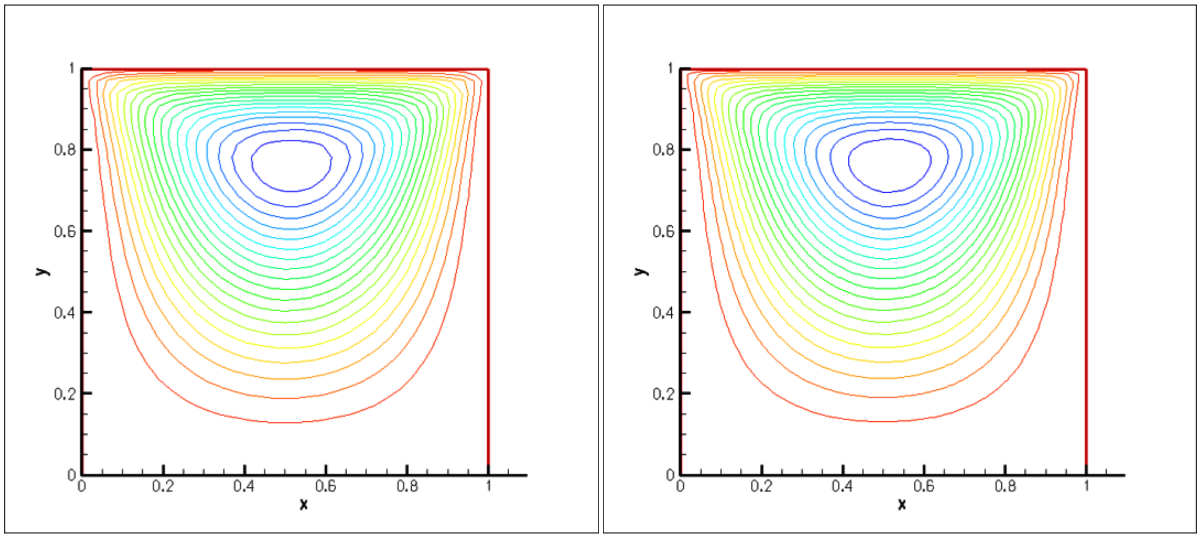


Figure 11: Comparison of a coarse (32x32) and fine (64x64) mesh solution for Re=10.

b) Velocity Plots:

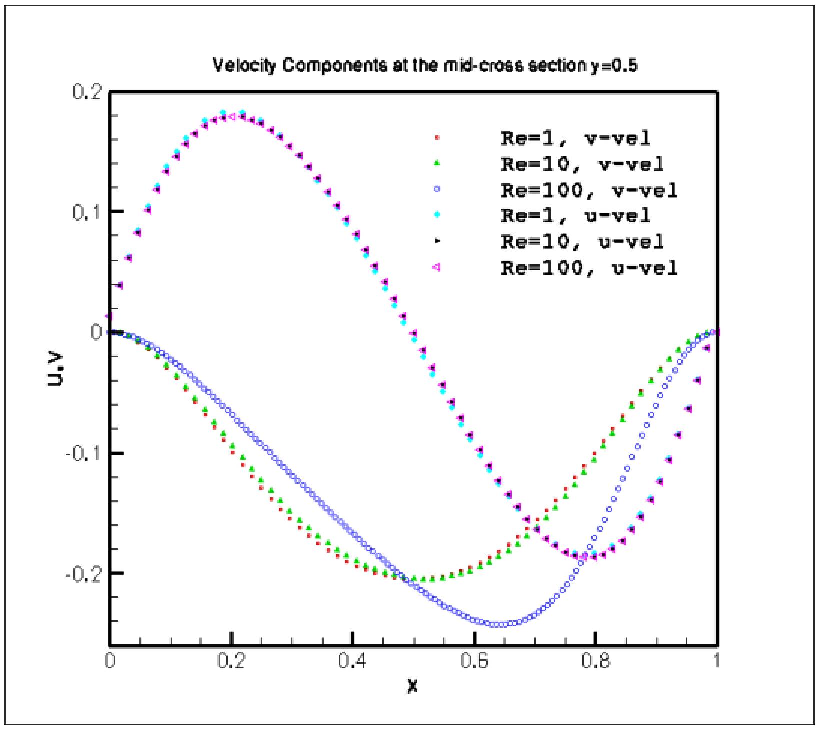


Figure 12: Velocity Distributions along the centerline y=0.5 for Re=1, 10, and 100 (Mesh 64x64).

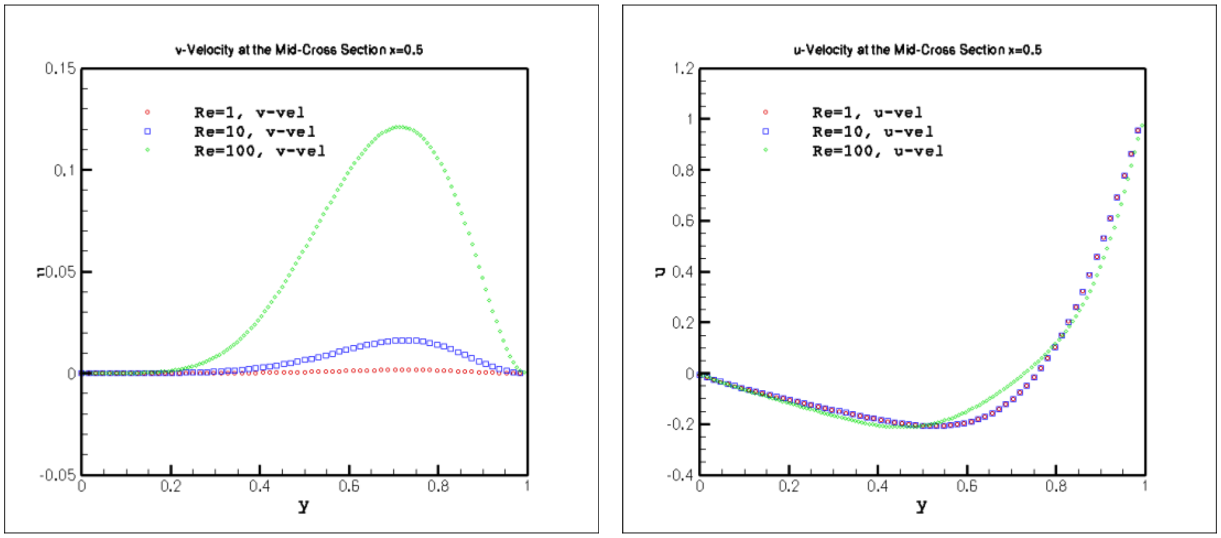
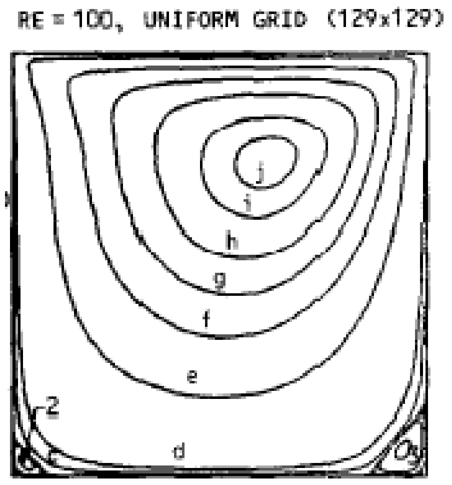


Figure13: Velocity Distributions along the centerline x=0.5 for Re=1, 10, 100 (Mesh 64x64).

3.2 Detailed Study of Higher Reynolds' Numbers:

Case 1: Streamfunction Plots: Re=100, Mesh 129X129



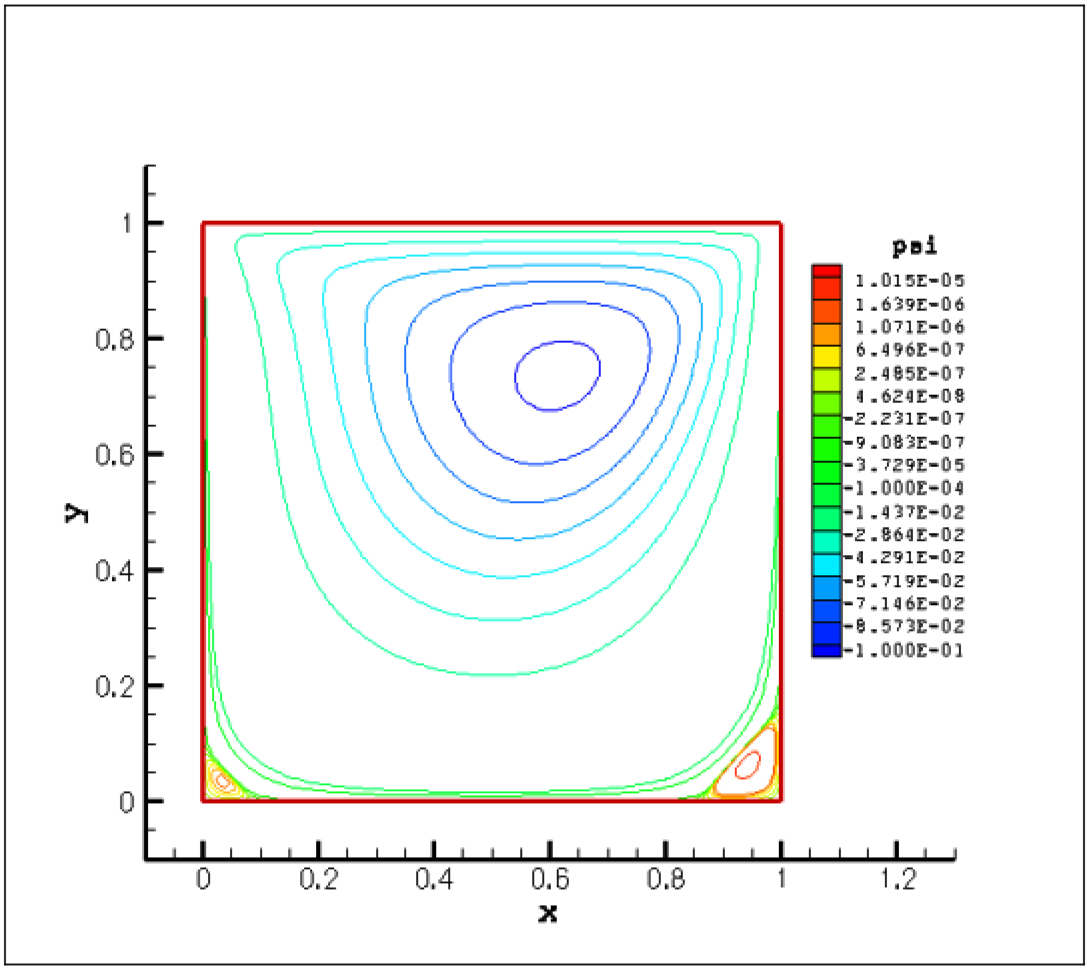
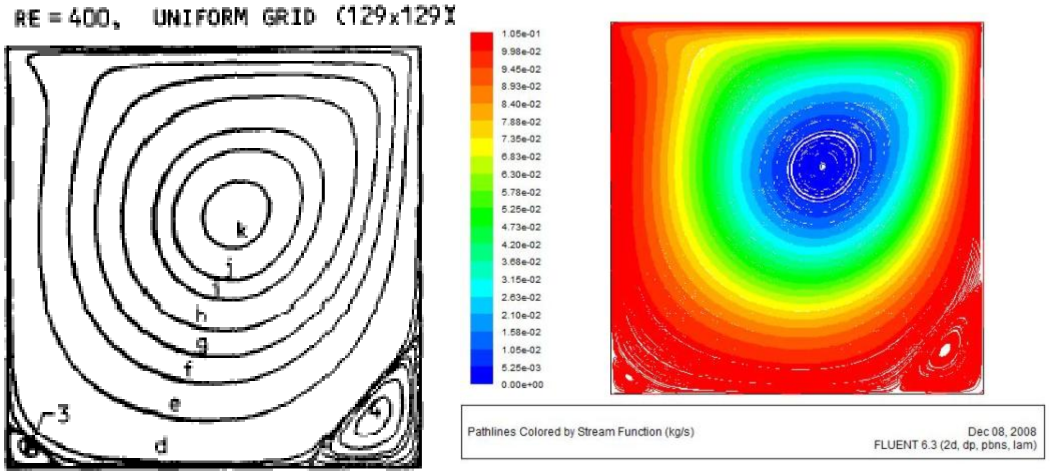


Figure14: Comparison of high Reynolds' number, Re=100 (Mesh 129x129), with Ghia et. al. (Streamfunction Plot) (4).

Case 2: Streamfunction Plots: Re=400, Mesh 129X129



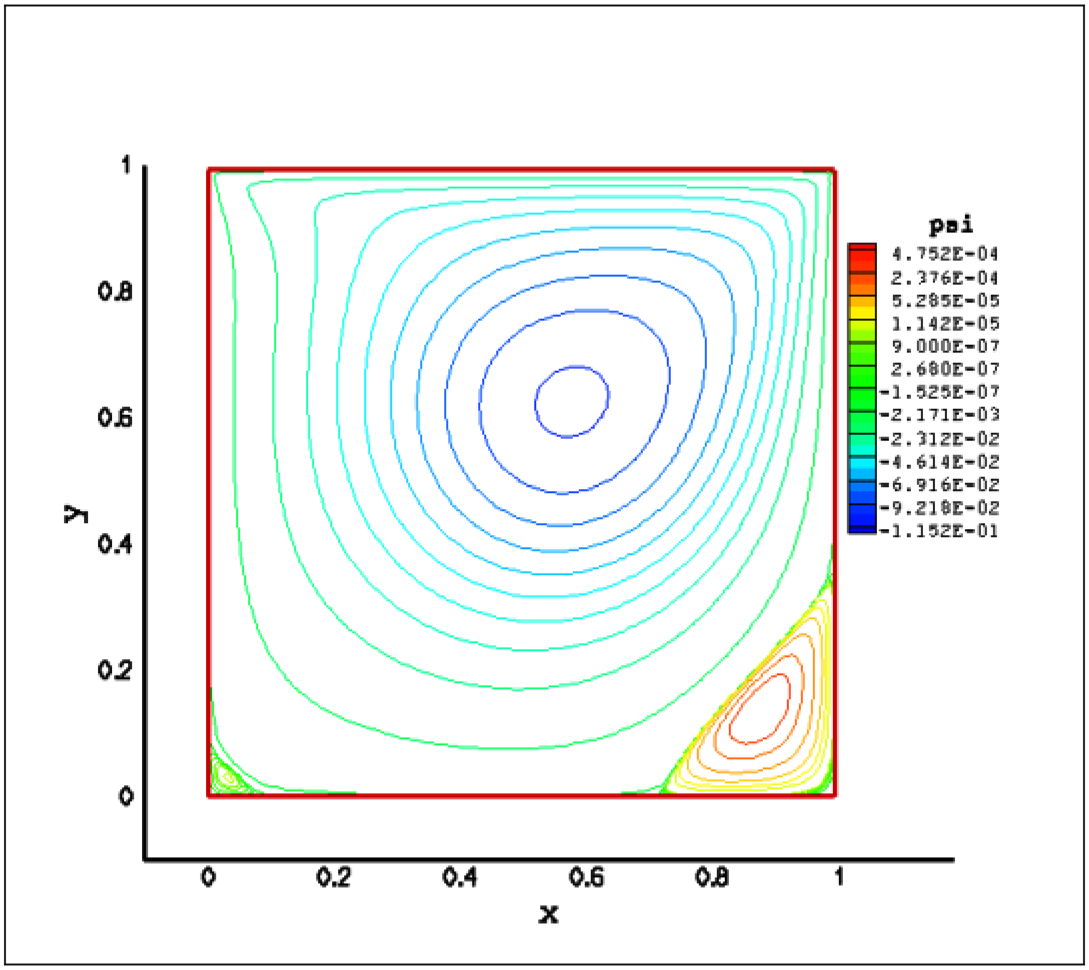
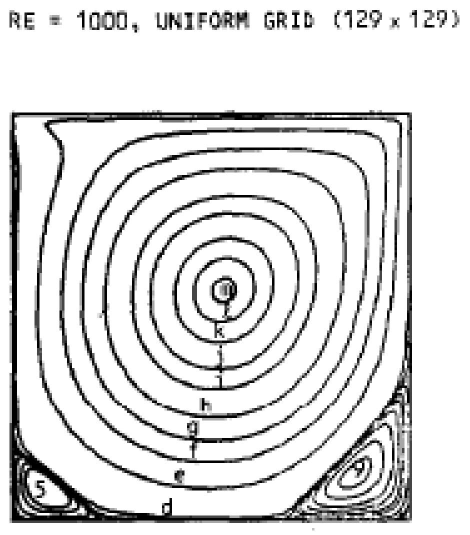


Figure 15: Comparison of high Reynolds' number, Re=400 (Mesh 129x129), with Ghia et. al. and FLUENT® Solution (Streamfunction Plot) (4).

Case 3: Streamfunction Plots: Re=1000, Mesh 129X129



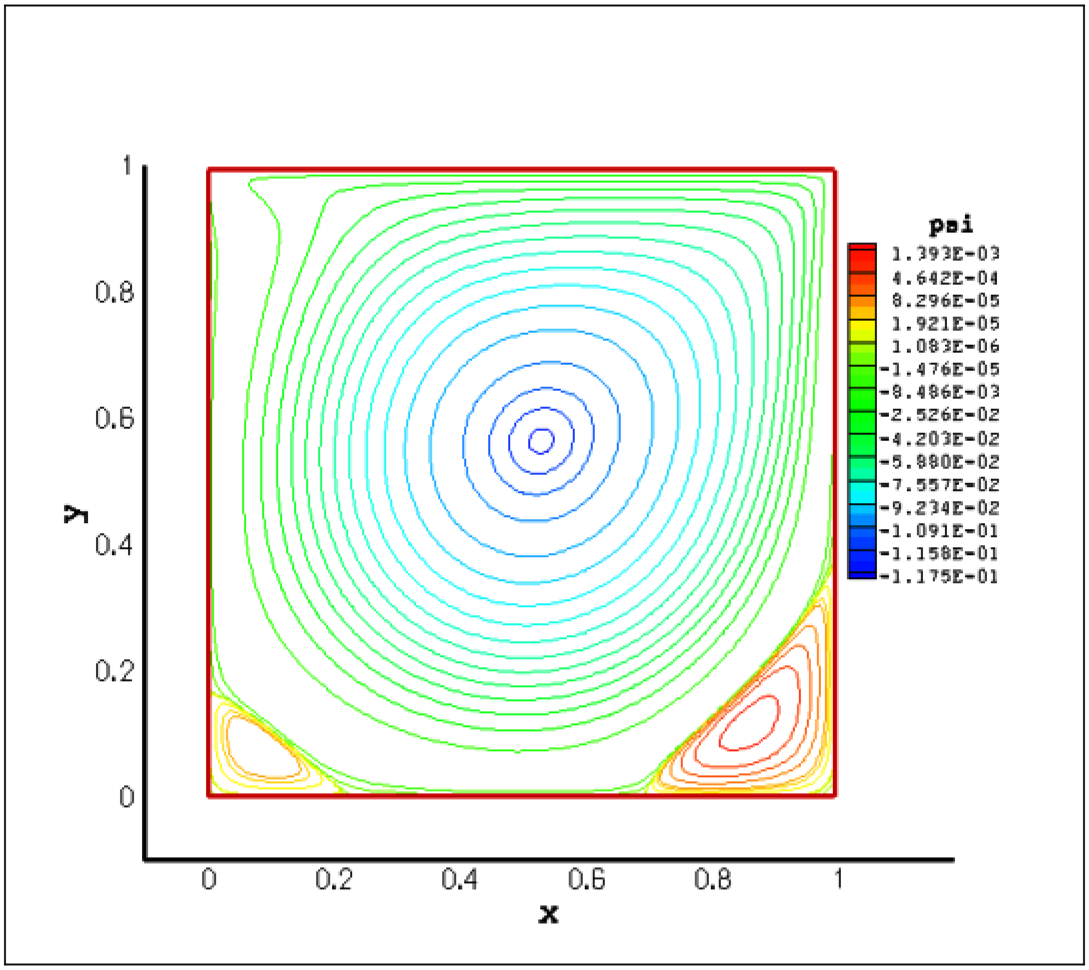


Figure16: Comparison of high Reynolds' number, Re=1000 (Mesh 129x129), with Ghia et. al. (Streamfunction Plot) (4).

b) Velocity Plots

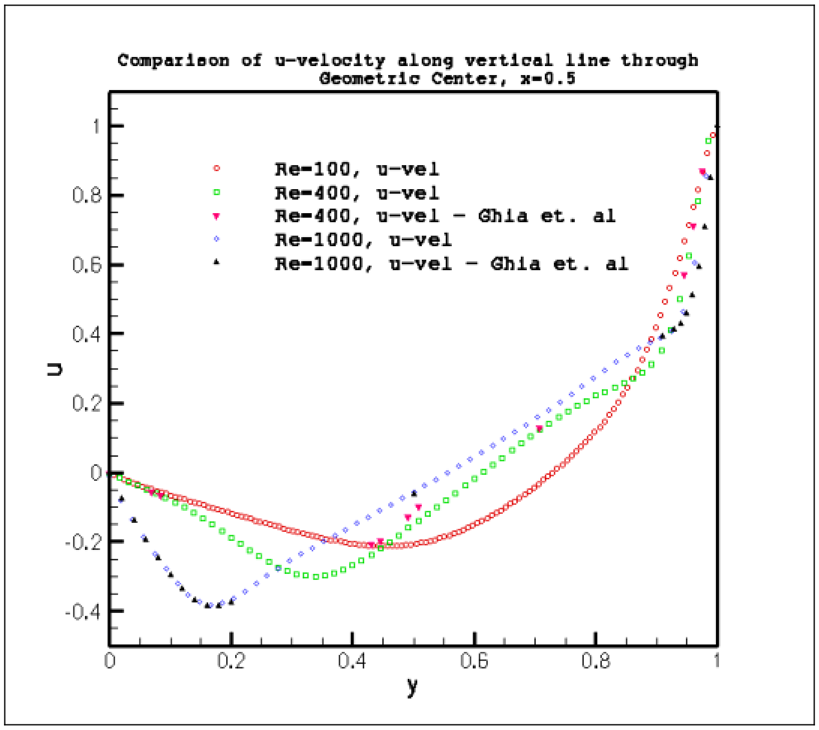


Figure17: Comparison of u velocity along a vertical line through Geometric Center, x=0.5 with Ghia et. al. (4).

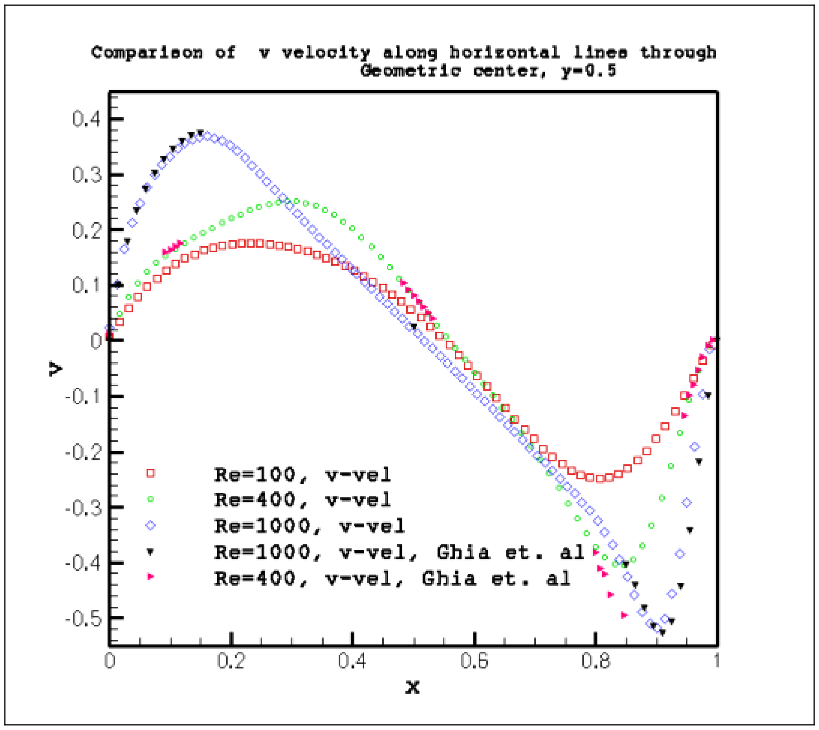


Figure18: Comparison of u velocity along a vertical line through Geometric Center, x=0.5 with Ghia et. al. (4)

3.3 Grid Refinement Study

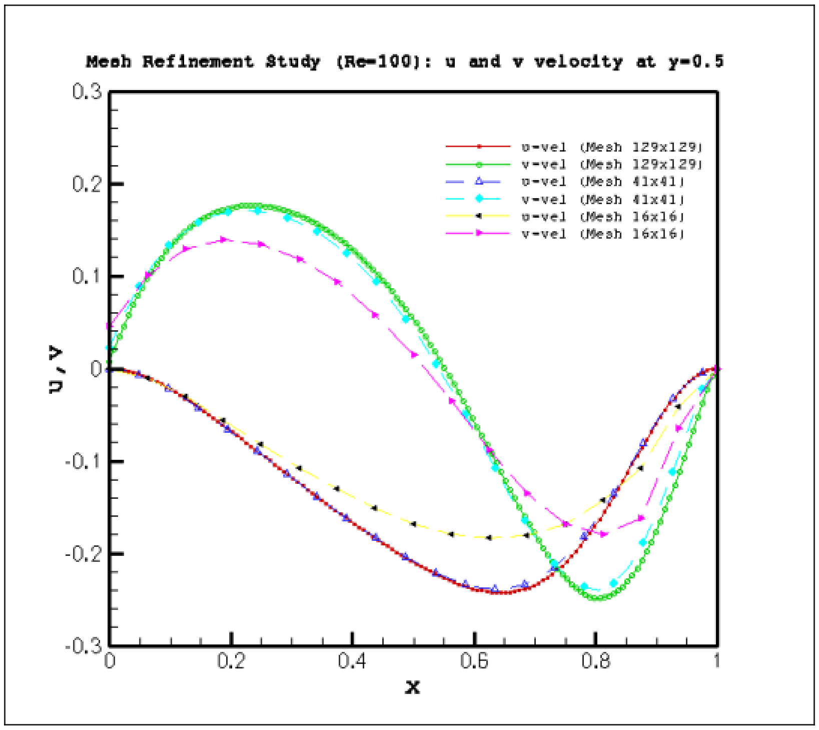
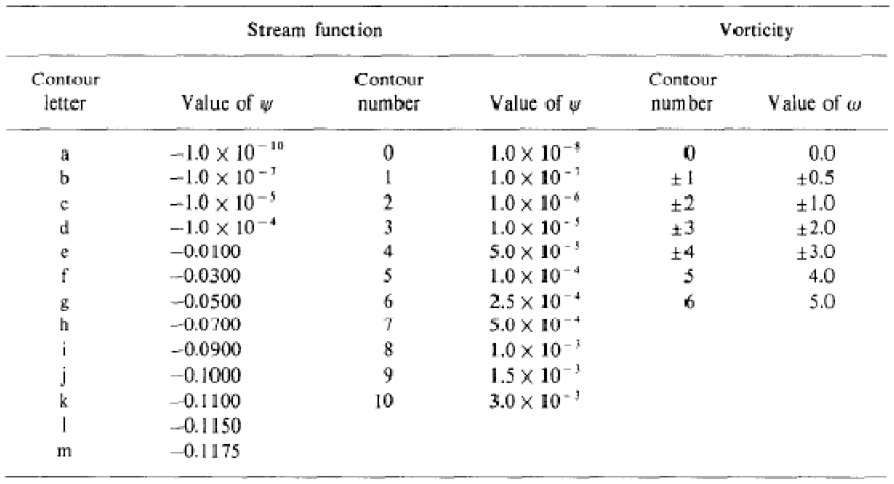


Figure 17: Mesh Refinement Study (Re=100)

3.3 Comparison and Discussion of Solutions

From the above results it can be concluded that the solution developed to solve the Incompressible Navier-Stokes equation was effective and was in close comparison with several published data. The validation plots obtained in section 3.0 was exactly comparable to the results obtained in the text by Hirsch (1). The velocity plots were in close correlation with the reference values and at the same time were exactly same as the numerical values presented in Hirsch. This was a foolproof comparison since the same solution technique was applied to obtain the numerical results presented. Hence the code developed was verified for low Reynolds' numbers from the validation case.

Further, for the high Re limit solutions published in the Journal of Computational Physics (4) were also in close comparison with the results obtained from this numerical analysis. Presented below (Figure 17) are the published results of the streamfunction values in Ghia et. al. (4) which when compared to the values obtained in section 3.2 was in close correlation.



Stream function				Vorticity	
Contour letter	Value of ψ	Contour number	Value of ψ	Contour number	Value of ω
a	-1.0×10^{-10}	0	1.0×10^{-8}	0	0.0
b	-1.0×10^{-7}	1	1.0×10^{-7}	± 1	± 0.5
c	-1.0×10^{-5}	2	1.0×10^{-6}	± 2	± 1.0
d	-1.0×10^{-4}	3	1.0×10^{-5}	± 3	± 2.0
e	-0.0100	4	5.0×10^{-5}	± 4	± 3.0
f	-0.0300	5	1.0×10^{-4}	5	4.0
g	-0.0500	6	2.5×10^{-4}	6	5.0
h	-0.0700	7	5.0×10^{-4}		
i	-0.0900	8	1.0×10^{-3}		
j	-0.1000	9	1.5×10^{-3}		
k	-0.1100	10	3.0×10^{-3}		
l	-0.1150				
m	-0.1175				

Figure 17: Streamfunction value for comparison on the Ghia et. al. Streamfunction plots (4).

It must be noted that discrepancies (negligible) in comparison with the Ghia et. al. (4)solution were attributed to the fact that a different formulation namely, the streamfunction-vorticity formulation, was used compared to the Projection method employed during the course of this project.

The Grid Refinement Study conducted in section 3.3 showed that the results converged to the reference solution as the mesh was made finer. This showed that the code was converging to the analytical solution with an increase in mesh size.

Comparing the performance of the two Pressure solvers namely; the Successive Over Relaxation Algorithm and the Gaussian Eliminations algorithm it can be concluded that both these algorithms are equally competent in solving the Pressure equation. The SOR has the advantage of being an iterative solver hence finer meshes can be solved without worrying about the memory of the system. But for the Gaussian Elimination algorithm finer meshes were harder to solve and was much more expensive in terms of memory requirements since it involved the storage of an N^2xN^2 matrix. It was also observed that due to the presence of the relaxation factor in the SOR algorithm the code accelerated with time. Presented below is a Residual comparison of the two schemes;

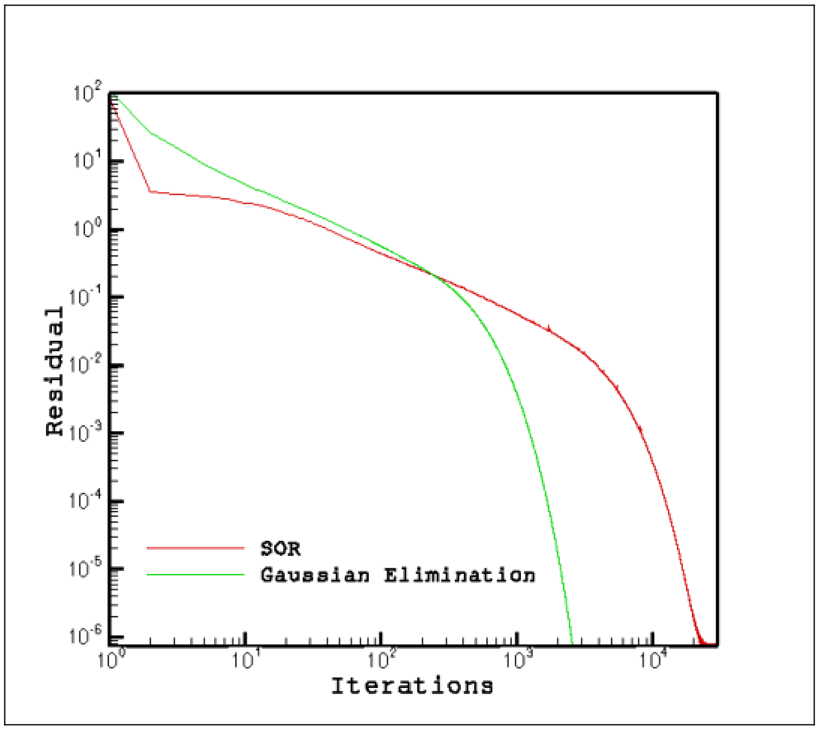


Figure 18: Residual comparison for SOR vs. Gauss Elimination

Overall the project was pleasantly successful but higher Reynolds' number cases were not computed due to time-constraints but will be tested in the future.

Works Cited

1. **Hirsch, Charles.** Numerical Computation of Internal & External Flows. *The Fundamentals of Computational Fluid Dynamics.* London : British Library, pp. 507-508.

2. **John C. Tannehill, Dale A. Anderson, Richard H. Pletcher.** *Computational Fluid Mechanics and Heat Transfer.* U.S.A : Taylor & Francis, 1997.

3. **Fidkowski, Krystof.** *Incompressible Navier Stokes Equations.* Ann Arbor : University of Michigan, 2008.

4. *High Re Solutions for Incompressible Flow Using Navier Stokes Equations and a Multigrid Method.* **U. Ghia, K.N. Ghia, and C.T. Shin.** Ohio : Jornal of Computational Physics , 1982, Vol. 48.