Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

2D Incompressible Flow Solver

Fundamentals of CFD
Semester Project

Omar Suleman

May 15, 2014

Instructor: Prof. Dr. Andreas Haselbacher

Dep. of Mechanical and Process Engineering, ETH Zürich

Abstract

This report discusses the development of a CFD program for solving 2D incompressible flow, using the Stream-Vorticity formulations of the Navier Stokes equations. The code uses 2^{nd} order, centered spatial discretizations, and an explicit first order temporal discretization. The SOR and ADI method are both implemented as iterative solvers for the Poisson Equation, then compared in terms of program run time. An order verification study is performed on the spatial discretizations used. The program is tested by simulating a Shear Driven Cavity Flow problem, and the results are compared with a similar study done previously by Ghia Et. Al. [1]. Finally, a grid refinement study is performed to observe the effects of a changing grid size on the numerical solution.

Contents

Contents **iii**

1 Introduction **1**
- 1.1 Governing Equations . . . 1
- 1.2 Test Case . . . 1
  - 1.2.1 Boundary Conditions . . . 2
  - 1.2.2 Initial Conditions . . . 3

2 Methods **5**
- 2.1 Program Structure . . . 5
- 2.2 Numerical Discretization . . . 6
  - 2.2.1 Spacial Discretization . . . 6
  - 2.2.2 Temporal Discretization . . . 6
- 2.3 Implementation of Boundary Conditions . . . 7
- 2.4 Solution Algorithms . . . 8
  - 2.4.1 Successive Over-Relaxation Method . . . 8
  - 2.4.2 Alternating Direction Implicit Method . . . 8

3 Results **11**
- 3.1 Order Verification Study . . . 11
  - 3.1.1 Spacial Discretization . . . 11
  - 3.1.2 Temporal Discretization . . . 13
- 3.2 Optimizing Solution Algorithm . . . 13
  - 3.2.1 Selection of Residual Tolerance . . . 13
  - 3.2.2 Selection of Relaxation Factor for SOR Solver . . . 13
  - 3.2.3 Comparison of SOR and ADI Run Time . . . 15
- 3.3 Model Validation . . . 15
  - 3.3.1 Comparing Steady State Solution with Literature . . . 15
  - 3.3.2 Effect of Grid Refinement on Steady State Solution . . . 17

4 Conclusion **23**

Bibliography **25**

Chapter 1

Introduction

This report outlines the background physics, numerical methods and validation of a CFD program developed in C++ to solve the 2D incompressible Navier Stokes equation, using the Steam-Vorticity formulation.

1.1 Governing Equations

The Vorticity equation 1.2 is obtained by taking the curl of the incompressible Navier Stokes momentum equations. In doing so, the pressure term is eliminated from these equations. In 2D, the problem is greatly simplified to only two unknowns, because the vorticity only has one component. The continuity equation can be written in terms of the other unknown, in the Stream Function 1.1.

$$\frac{\delta^2\overline{\psi}}{\delta\overline{x}^2} + \frac{\delta^2\overline{\psi}}{\delta\overline{y}^2} = -\overline{\zeta} \tag{1.1}$$

$$\frac{\delta\overline{\zeta}}{\delta\overline{t}} + \overline{u}\frac{\delta\overline{\zeta}}{\delta\overline{x}} + \overline{v}\frac{\delta\overline{\zeta}}{\delta\overline{y}} = \frac{1}{Re}\Big(\frac{\delta^2\overline{\zeta}}{\delta\overline{x}^2} + \frac{\delta^2\overline{\zeta}}{\delta\overline{y}^2}\Big) \tag{1.2}$$

$$\overline{u} = \frac{\delta\overline{\psi}}{\delta\overline{y}} \tag{1.3}$$

$$\overline{v} = -\frac{\delta\overline{\psi}}{\delta\overline{x}} \tag{1.4}$$

1.2 Test Case

The problem being simulated to validate the model is shear driven cavity flow. This problem involves laminar flow of an incompressible, viscous fluid in a square cavity, where all the walls are stationary, except for the top wall moving at a constant velocity. The steady state data from this study will be compared to a previous study of this problem by Ghia Et. Al. [1].

1.2.1 Boundary Conditions

For all the walls, the velocity normal to the wall is zero because the mass flux at the wall must be zero, since no mass leaves or enters the control volume. Based on this condition and Equations 1.3 and 1.4, $\psi|_{wall}$ is constant in the direction parallel to the walls. Since all the walls are connected, and $\psi|_{wall}$ is constant along each wall, $\psi|_{wall}$ is the same constant for all walls. This constant was arbitrarily set to zero; it's value will not affect the vorticity or velocity field, since these quantities are functions of the derivative of ψ, and not ψ itself.

$$\psi\Big|_{wall} = c = 0 \tag{1.5}$$

The Poisson Equation 1.1 can be simplified knowing that ψ is constant when moving along the wall. This simplification is shown in Equations 1.6 and 1.7.

$$\overline{\zeta}(x,0) = \overline{\zeta}(x,1) = -\frac{\delta^2\overline{\psi}}{\delta\overline{y}^2} \tag{1.6}$$

$$\overline{\zeta}(0,y) = \overline{\zeta}(1,y) = -\frac{\delta^2\overline{\psi}}{\delta\overline{x}^2} \tag{1.7}$$

The velocity parallel to the wall is equal to the wall velocity, to satisfy the no slip boundary condition. This is expressed in Equations 1.8 to 1.9. This second boundary condition will be applied to Equations 1.6 and 1.7 when the equations are discretized, to create a boundary condition for ζ.

$$\overline{v}(0,y,t) = \overline{v}(1,y,t) = \overline{u}(x,0,t) = 0 \tag{1.8}$$

$$\overline{u}(x,1,t) = \frac{u}{u_{ref}} = 1 \tag{1.9}$$

1.2.2 Initial Conditions

Since only the steady state data is of interest, the initial conditions can be arbitrary because the solutions will converge to the same steady state values, regardless of initial conditions. The initial conditions selected are given in Equation 1.10 and 1.11.

$$\overline{\zeta}(x,y,0) = 0 \tag{1.10}$$

$$\overline{\psi}(x,y,0) = 0 \tag{1.11}$$

Chapter 2

Methods

2.1 Program Structure

The program is implemented in C++, using object orientated programming. The goal was to make the structure modular so that in future, a variety of problems could be solved on many different style grids, in higher dimensions.

The spatial domain is discretized and stored as a grid object. The grid class holds a fluid element object for each discrete point in the spatial domain, as well as any parameters specific to that particular grid. The fluid element holds all variables of interest at that particular point in space, as well as its position in space. All the variables stored at the fluid element object level are non-dimensionalized by the reference values stored in the grid object. Since all the variables in the governing equations are stored in the fluid element, the equations being solved are non-dimensional.

The data is output using the Visualization Toolkit from Kitware. All the scalar variables, and the velocity fields are written to a VTK file at specified dump period. The data can be directly opened in ParaView and visualized directly. Additionally, a text file is written out at the end of each simulation and contains a text file with the u-velocity component along the vertical centerline, and the v-velocity component along the horizontal centerline. The program has no runtime arguments. All the parameters are specified in the Main.cpp file. For instruction on compiling and executing the program, refer to the readme.txt in the main project folder.

2.2 Numerical Discretization

The numerical schemes used to discretize the governing equations presented in Section 1.1 are provided below.

2.2.1 Spacial Discretization

The first and second spacial derivatives are discretized with second order accurate, centered schemes. The equations for the first order derivatives are given in Equation 2.1 and Equation 2.2. The equations for the second order derivatives are given in Equation 2.3 and Equation 2.4.

$$\left.\frac{\partial u}{\partial x}\right|_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \tag{2.1}$$

$$\left.\frac{\partial u}{\partial y}\right|_{i,j} = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \tag{2.2}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{2.3}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta x^2} \tag{2.4}$$

The derivative operator can be applied to any variable stored in the fluid element class. Note that this derivative approximation is only valid for constant Δx and Δy.

2.2.2 Temporal Discretization

The temporal derivative is discretized with the explicit, first order accurate Euler scheme, and used to solve for the next time step, as shown in Equation 2.5

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \left.\frac{\partial u}{\partial t}\right|_{i,j}^{n} \Delta t \tag{2.5}$$

The approximation in Equation 2.5 is only first order accurate, unlike the spacial approximation.

2.3 Implementation of Boundary Conditions

The boundary condition for ψ, given in Equation 1.5, can be directly implemented. The boundary condition for ζ must be extracted from the no slip boundary conditions, in Equations 1.8 and 1.9, and applied to the simplified Poisson Equations in 1.6 and 1.7. These equations are discretized with the schemes given in Section 2.2, and displayed in Equations 2.6 to 2.13. Also note that $\psi|_{wall}$ was set to zero in Equation 1.5, allowing further simplification of the discrete equations. Using these discrete equations, a boundary condition expression for ζ was obtained and implemented into the solver; these equations are displayed in 2.14 to 2.17.

$$\bar{\zeta}_{i,0} = -\frac{\psi_{i,1} - 2\psi_{i,0} + \psi_{i,-1}}{\Delta y^2} = \frac{\psi_{i,1} + \psi_{i,-1}}{\Delta y^2} \tag{2.6}$$

$$\bar{\zeta}_{i,ny-1} = -\frac{\psi_{i,ny} - 2\psi_{i,ny-1} + \psi_{i,ny-2}}{\Delta y^2} = -\frac{\psi_{i,ny} + \psi_{i,ny-2}}{\Delta y^2} \tag{2.7}$$

$$\bar{\zeta}_{0,j} = -\frac{\psi_{1,j} - 2\psi_{0,j} + \psi_{-1,j}}{\Delta x^2} = -\frac{\psi_{1,j} + \psi_{-1,j}}{\Delta x^2} \tag{2.8}$$

$$\bar{\zeta}_{nx-1,j} = -\frac{\psi_{nx,j} - 2\psi_{nx-1,j} + \psi_{nx-2,j}}{\Delta x^2} = -\frac{\psi_{nx,j} + \psi_{nx-2,j}}{\Delta x^2} \tag{2.9}$$

$$\frac{\psi_{i,1} - \psi_{i,-1}}{2\Delta y} = 0 \tag{2.10}$$

$$\frac{\psi_{i,ny} - \psi_{i,ny-2}}{2\Delta y} = 1 \tag{2.11}$$

$$\frac{\psi_{1,j} - \psi_{-1,j}}{2\Delta x} = 0 \tag{2.12}$$

$$\frac{\psi_{nx,j} - \psi_{nx-2,j}}{2\Delta x} = 0 \tag{2.13}$$

$$\zeta_{i,0} = 2\frac{\psi_{i,1}}{\Delta y} \tag{2.14}$$

$$\zeta_{i,ny-1} = 2\Big(\frac{\psi_{i,ny-2}}{\Delta y} - \Delta y\Big) \tag{2.15}$$

$$\zeta_{0,j} = 2\frac{\psi_{1,j}}{\Delta y} \tag{2.16}$$

$$\zeta_{nx-1,j} = 2\frac{\psi_{nx-2,j}}{\Delta x} \tag{2.17}$$

2.4 Solution Algorithms

To solve the Poisson equation, an iterative solver was required. Two different algorithms were used, and are detailed below.

2.4.1 Successive Over-Relaxation Method

The Successive Over-Relaxation (SOR) Method iterates through the grid, and explicitly solves for the value of the variable being solved, so that it satisfies the governing equation. Next, the value of the variable is updated on the grid using a linear combination of the old and new value. This calculation is shown for one single point on the grid in Equation 2.18 and 2.19. This process is repeated until the the difference between $\psi_{i,j,new}$ and $\psi_{i,j}$ is less than a specified tolerance, everywhere in the grid.

$$\psi_{i,j,new} = \frac{(\psi_{i+1,j} + \psi_{i-1,j}) + (\psi_{i,j+1} + \psi_{i,j-1})\left(\frac{\Delta x}{\Delta y}\right)^2 + \zeta_{i,j}(\Delta x)^2}{2\left(1 + (\frac{\Delta x}{\Delta y})^2\right)} \tag{2.18}$$

$$\psi_{i,j} = \omega\psi_{i,j,new} + (1-\omega)\psi_{i,j,old} \tag{2.19}$$

2.4.2 Alternating Direction Implicit Method

The ADI Method is an implicit numerical scheme for temporal discretization of linear equations. Time steps are subdivided into smaller steps (2 steps for 2D ect.), and at each partial step, only terms in a single dimension are represented implicitly. After one partial time step, the terms previously represented implicitly, are represented explicitly, and the terms in the next dimension are represented implicitly. In doing so, a complicated block tri-diagonal linear system is avoided, and instead, the linear system being solve is tri-diagonal. This is cheaper computationally to solve than a tri-diagonal system, and this implicit scheme has favorable stability properties, allowing larger time steps.

A time derivative term was added to the Poisson Equation 1.1, and the Alternating Direction Implicit (ADI) Method was used to solve for the steady state solution of this equation. The modified Poisson Equation is given in 2.20. A variable time step is used for fast convergence, and is defined in equations 2.23 to 2.25. The resulting tri-diagonal system was solved with the Thomas algorthm for tri-diagonal matrices.

$$\frac{\delta\overline{\psi}}{\delta\overline{t}} = \frac{\delta^2\overline{\psi}}{\delta\overline{x}^2} + \frac{\delta^2\overline{\psi}}{\delta\overline{y}^2} + \overline{\zeta} \tag{2.20}$$

$$\overline{\psi}_{i,j}^{n+\frac{1}{2}} = \overline{\psi}_{i,j}^{n} + \frac{Re\Delta t}{\Delta x^2}(\overline{\psi}_{i+1,j}^{n+\frac{1}{2}} - 2\overline{\psi}_{i,j}^{n+\frac{1}{2}} + \overline{\psi}_{i-1,j}^{n+\frac{1}{2}}) + \frac{Re\Delta t}{\Delta y^2}(\overline{\psi}_{i,j+1}^{n} - 2\overline{\psi}_{i,j}^{n} + \overline{\psi}_{i,j-1}^{n}) + \zeta_{i,j}^{n}\Delta t \tag{2.21}$$

$$\overline{\psi}_{i,j}^{n+1} = \overline{\psi}_{i,j}^{n+\frac{1}{2}} + \frac{Re\Delta t}{\Delta x^2}(\overline{\psi}_{i+1,j}^{n+\frac{1}{2}} - 2\overline{\psi}_{i,j}^{n+\frac{1}{2}} + \overline{\psi}_{i-1,j}^{n+\frac{1}{2}}) + \frac{Re\Delta t}{\Delta y^2}(\overline{\psi}_{i,j+1}^{n+1} - 2\overline{\psi}_{i,j}^{n+1} + \overline{\psi}_{i,j-1}^{n+1}) + \zeta_{i,j}^{n+\frac{1}{2}}\Delta t \tag{2.22}$$

$$\rho = \frac{\Delta x^2}{\Delta t} \tag{2.23}$$

$$\rho = 4\cos^2\left(\frac{\pi dx}{2}\right)\tan^2\left(\frac{\pi dx}{2}\right)^{\frac{(2p-1)}{2p_{max}}} \tag{2.24}$$

$$p = 1, 2..p_{max}, 1, 2..p_{max} = \frac{\log 10(\tan^2(\frac{\pi dx}{2}))}{2\log 10(\sqrt{2}-1)}.. \tag{2.25}$$

Chapter 3

Results

3.1 Order Verification Study

3.1.1 Spacial Discretization

A grid refinement study was performed to verify the order of accuracy of the spatial discretizations discussed in Section 2.2. In order to determine the error of the approximations, The Method of Manufactured Solutions was implemented. A manufactured solution was selected, ψ_{manu}, and the analytic right hand side of the corresponding numerical scheme was computed. The analytic right hand side was used in the numerical scheme, to compute a numerical solution, ψ_{num}, that satisfied the numerical scheme to a specified tolerance. A tolerance of 10^{-12} was selected to ensure that the error owing from this tolerance would be negligible in comparison to the discretization error. The total error could then be computed from Equation 3.1.

$$\varepsilon = |(\psi_{manu} - \psi_{num})| \tag{3.1}$$

Using Equation 3.1, the error for each point in a specific size grid was computed to determine the maximum error, ε_{max}. The maximum error was compared for different grid sizes, and the results are plotted on a log scale in Figure 3.1. The dashed vertical lines indicate the minimum and maximum grid spacing that the numerical schemes will be implemented on.

Using the plots in Figure 3.1, a linear curve was fit to the data, and the result are summarized in Table 3.1. The slope of all the curves is approximately 2, therefore the maximum error in the grid is proportional to the square of the grid spacing, as shown in Proposition 3.1, and is consistent with the numerical schemes selected. It can also be stated that for the selected grid

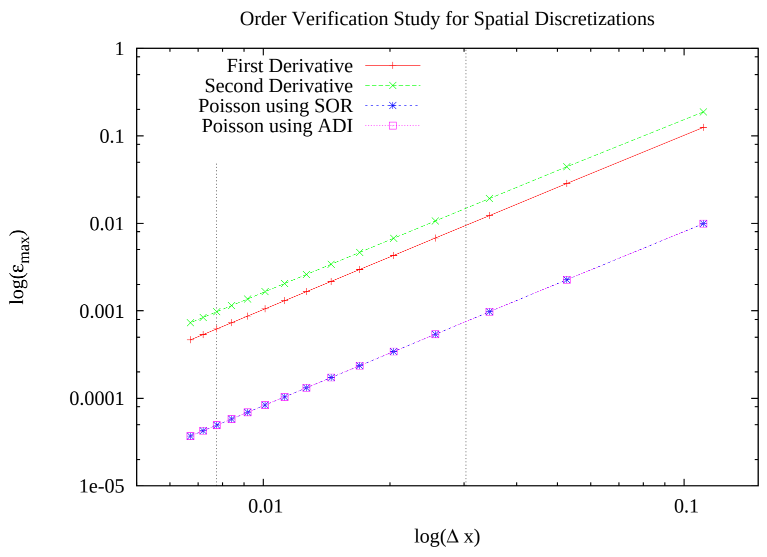

Figure 3.1: Maximum numerical error as a function of grid spacing for spatial discretizations, plotted on a log-log scale.

Table 3.1: Summary of Slope Data for Figure 3.1

Numerical Disc.	Slope	Error
1st Derivative	1.99461	$\pm$ 0.001142
1st Derivative	1.98334	$\pm$ 0.003548
Poisson w/ SOR	1.99383	$\pm$ 0.001311
Poisson w/ ADI	1.99384	$\pm$ 0.001312

sizes of interest for this study, the numerical error is mainly dominated by discretization error.

Proposition 3.1 *The slope of the log plot gives the power of the relationship between the two plotted variables.*

Proof

$$\log(y) = a\log(x) + b$$

$$e^{\log(y)} = e^{a\log(x)+b}$$

$$y = cx^a$$

$$y \propto x^a$$

$\square$

3.1.2 Temporal Discretization

A first order accurate temporal discretization was selected for these simulations because only steady state data was required, and the temporal discretization should have no effect on these results. An order verification study wasn't performed on the temporal discretization, but instead two more simulations were run using two different time steps, to ensure that the steady state results were indifferent to changes in time step. Unfortunately, some of the numerical precision was lost when the data was written out to file with only 7 decimal places of precision. The only conclusion that can be drawn is that the results were indifferent to 7 decimal places of precision.

Additionally, instabilities were observed for a CFL number of 0.5, and the solution failed to converge. A stability analysis was not performed for the forward time, central space (FTCS) scheme implemented, so no comments can be made on this result. The largest CFL number selected, that leads to stable solutions is 0.2. There is very little resolution on where the stability limit is; the stability limit is somewhere between $0.2 < CFL < 0.5$.

3.2 Optimizing Solution Algorithm

3.2.1 Selection of Residual Tolerance

The residual tolerance was selected to be as large as possible, without compromising the accuracy of the numerical schemes selected. This was done by varying the tolerance, performing an order verification study, as described in Section 3.1, and observing the effect of tolerance on the solutions order of accuracy. This process is shown, for both the SOR and ADI solvers, in Figures 3.2 and 3.3. The vertical lines correspond to the fine, medium and coarse grid respectively.

From Figures 3.2 and 3.3, the deviation from the linear slope of 2 indicates that the error owing from the residuals is non negligible. The tolerances were selected such that they were as large as possible, while still maintaining a linear slope, as they crossed the corresponding grid's vertical line, from right to left. The selected tolerance for each grid is the same for both the SOR and ADI Solvers, and are summarized in Table 3.2.

3.2.2 Selection of Relaxation Factor for SOR Solver

To pick an optimal relaxation factor ω for the SOR solver, the parameter was varied from 0.5 to 1.99, in increments of 0.01, and the number of iterations required to solve the Poisson equation was recorded and plotted in

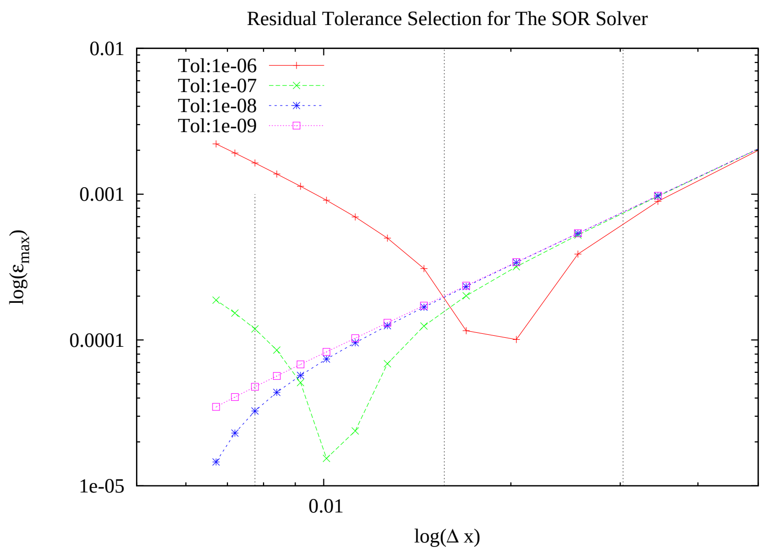

Figure 3.2: Maximum numerical error as a function of grid spacing, for different residual tolerances, plotted on a log-log scale, for the SOR Solver.

Table 3.2: Optimal Residual Tolerance

Grid Size	Optimal Tolerance
33x33	10^{-7}
65x65	10^{-8}
129x129	10^{-9}

Table 3.3: Optimal Relaxation Factor ω

Grid Size	Tolerance	Min. Iterations	ω
33x33	10^{-7}	129	1.82
65x65	10^{-8}	258	1.91
129x129	10^{-9}	513	1.95

Figure 3.4. The Relaxation Factor yielding the minimum iterations was selected as optimal. For each new value of ω, the same initial conditions for ψ were re-initialized, and the right hand side was kept constant. The optimal relaxation factors are summarized in Table 3.3.

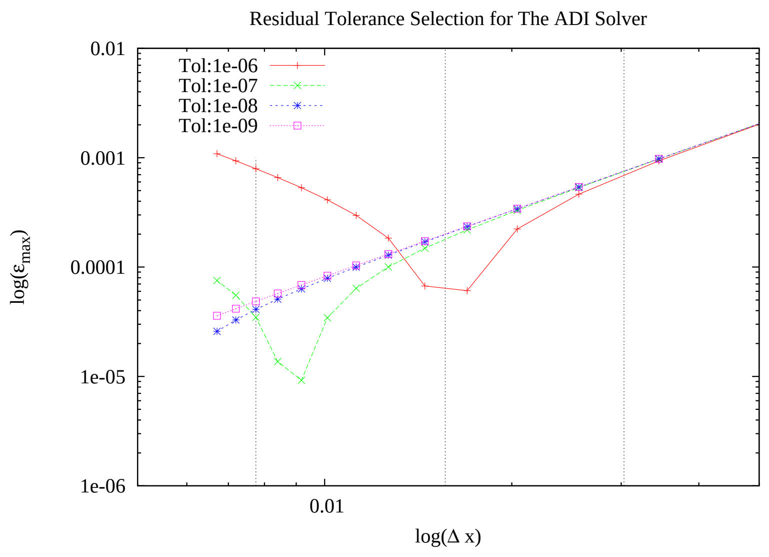

Figure 3.3: Maximum numerical error as a function of grid spacing, for different residual tolerances, plotted on a log-log scale, for the ADI Solver.

Table 3.4: Comparison of SOR and ADI Run Times

Grid	Reynolds	time steps	ADI Runtime	SOR Runtime
33x33	100	161	0m1.400	0m3.298
65x65	100	161	22.173s	0m3.298
129x129	100	65	2m58.250s	9.863s

3.2.3 Comparison of SOR and ADI Run Time

The SOR method was many times faster than the ADI method. Something was not right with the ADI method, but unfortunately the bug was not discovered in time. The results from the run time test are displayed in Table 3.4, but because of the falsely slow results of the ADI method, nothing can be concluded from this data.

3.3 Model Validation

3.3.1 Comparing Steady State Solution with Literature

The results from the simulation were compared with similar computational work done by Ghia Et. Al. Both studies solve the stream vorticity equa-

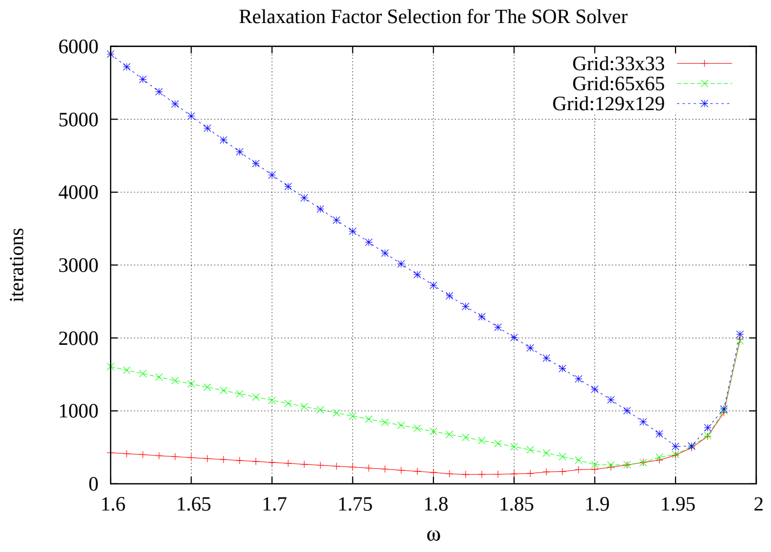

Figure 3.4: Number of iterations as a function of The Relaxation Factor for The SOR Solver

tion on a uniform 129*x*129 grid with second order accurate, finite difference schemes. Both studies also use second order accurate, centered schemes For the second order derivatives. For the convective terms in the vorticity equation, Ghia Et. Al. uses a first order, upwind scheme, and a corrective term at the end to provide second order accuracy. Both studies employ the same physical boundary conditions, but the schemes used to employ them differ slightly, though both are second order accurate [1].

The x component velocity, along the domains vertical center line, and the y component velocity, along the domains horizontal center line are compared with with Ghia Et. Al., for Reynolds Number of 100 and 1000, in Figures 3.5 and 3.12.

For the 100 Reynolds Number case, the velocity profiles from the two studies match up quite closely. In Figure 3.5, the u-velocity along the vertical centerline shows a slight discrepancy near the top surface though. At this point, very high velocity gradients are also observed. The differences in discretization schemes implemented at the boundaries may be more pronounced because of these high gradients, and could be causing these differences. Furthermore, the difference is more noticeable for the higher Reynolds Number case in Figure 3.7, where the gradients are also noticeably higher. Another observation from Figures 3.5 to 3.8 is that there are noticeable differences between the two curves in places with high curvature. These differences may be owing to truncation errors, where higher order derivatives become more significant.

3.3.2 Effect of Grid Refinement on Steady State Solution

Having validated the fine grid solution with literature, grid spacing and its affects on the steady state solution was examined. Simulations were run on $33x33$ and $65x65$ grids, and were compared to simulations on the fine grid. The u-velocity along the vertical centerline, and v-velocity along the horizontal centerline is plotted for the 3 grids in Figures 3.9 to 3.12.

At the lower Reynolds Number, Both the medium and the coarse grid are very close to the fine grid solution. In areas where the velocity has high curvature, the coarse grid deviates slightly off from the other two grid's velocity profile.

For the high Reynolds Number, the medium grid deviates slightly off the fine grid solution. The coarse grid has a significantly different profile in areas with steep gradients, near the boundaries. The error induced by steep gradients at the boundaries propagates inwards, and as a results, the coarse grid is never really accurate. Close to the center of the domain, where the profiles are flat, the coarse grid has a similar shape to the other grids, but error from the boundaries causes the solution to be drastically different. Since the truncation error is proportional to Δx^2, for the numerical schemes implemented, this is not a surprising result. The leading truncation error term is 16 times larger than that of the fine grid. As a result, truncation errors, which become significant in areas with changes in curvature, are 16 times higher in the coarse grid, then the fine grid.

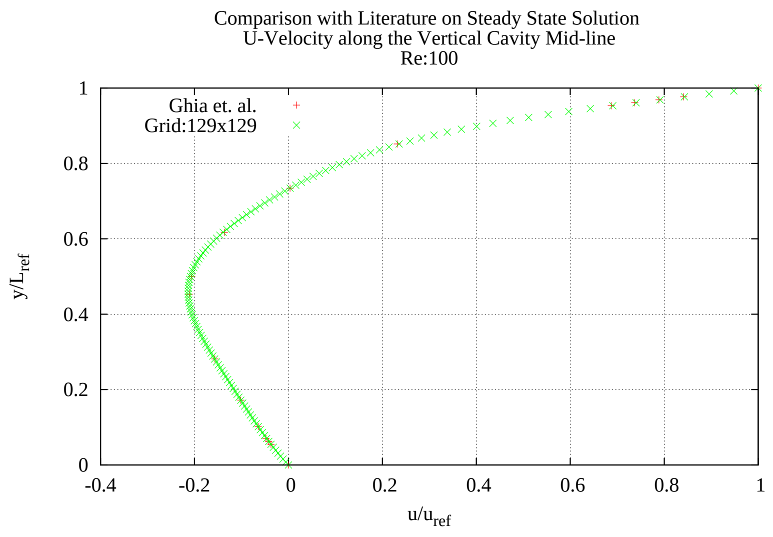

Figure 3.5: Steady state u-velocity as a function of y, along the vertical cavity mid-line, at Re:100, for comparison with literature.

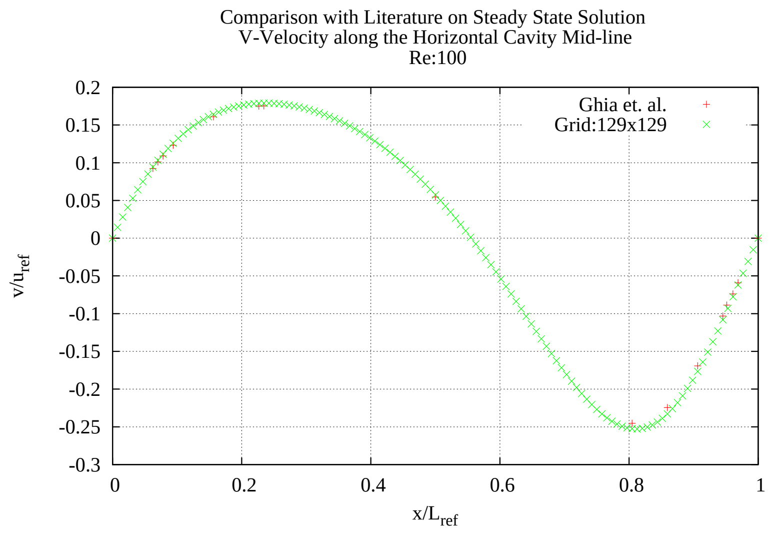

Figure 3.6: Steady state v-velocity as a function of x, along the horizontal cavity mid-line, at Re:100, for comparison with literature.

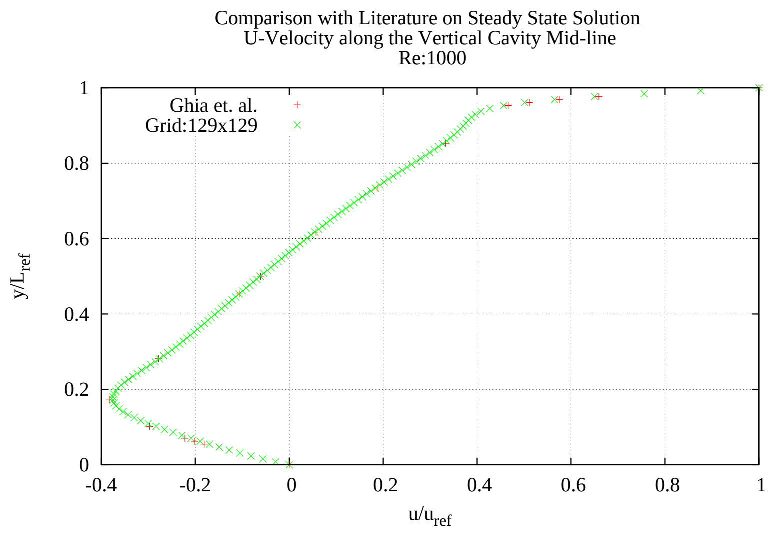

Figure 3.7: Steady state u-velocity as a function of y, along the vertical cavity mid-line, at Re:1000 for comparison with literature

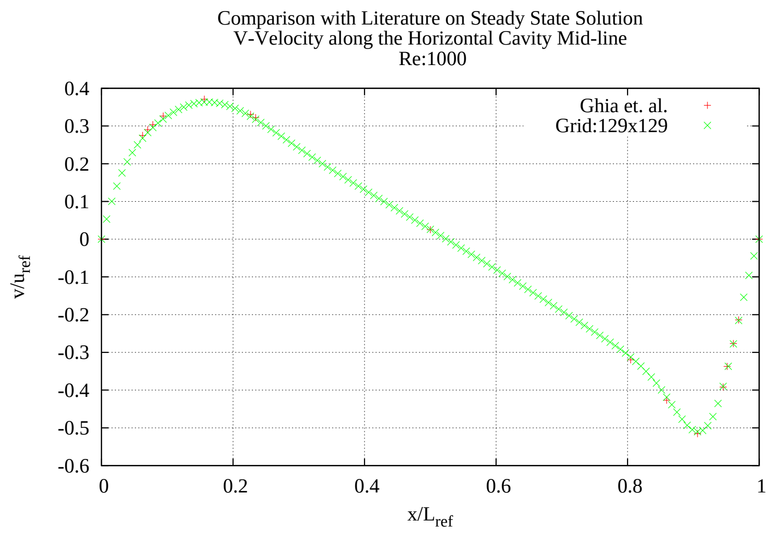

Figure 3.8: Steady state v-velocity as a function of x, along the horizontal cavity mid-line, at Re:1000 for comparison with literature.

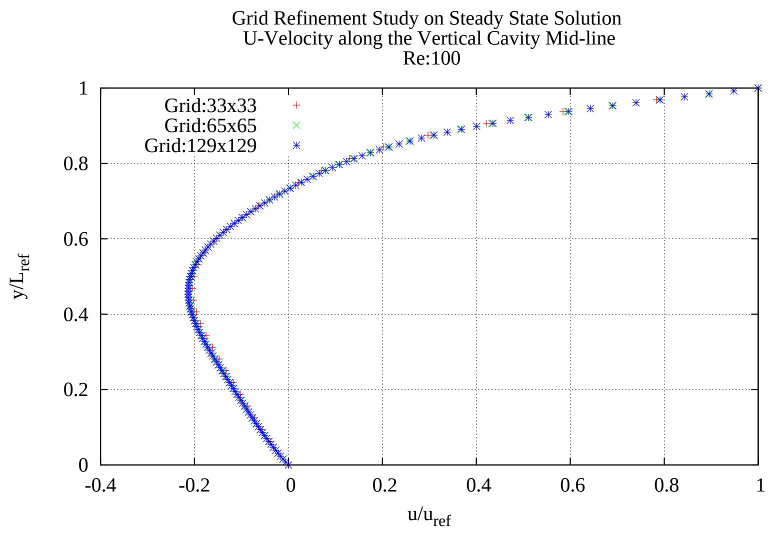

Figure 3.9: Steady state u-velocity as a function of y, along the vertical cavity mid-line, for 3 different grid sizes, at Re:100.

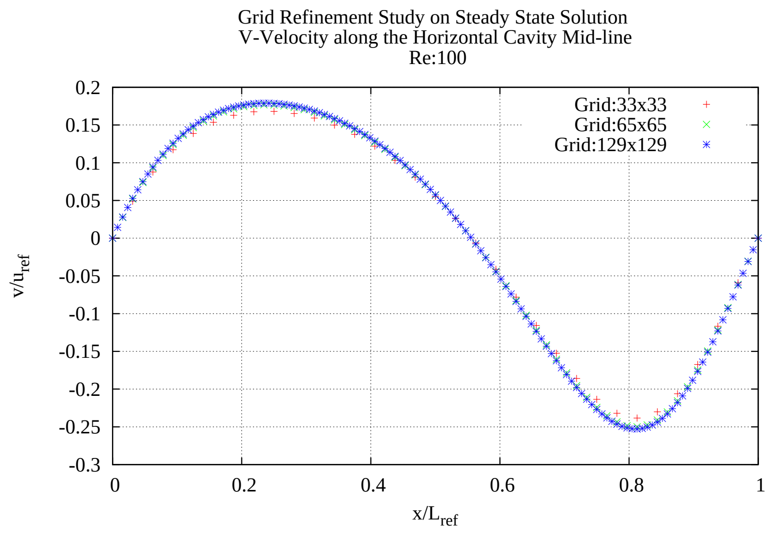

Figure 3.10: Steady state v-velocity as a function of x, along the horizontal cavity mid-line, for 3 different grid sizes, at Re:100.

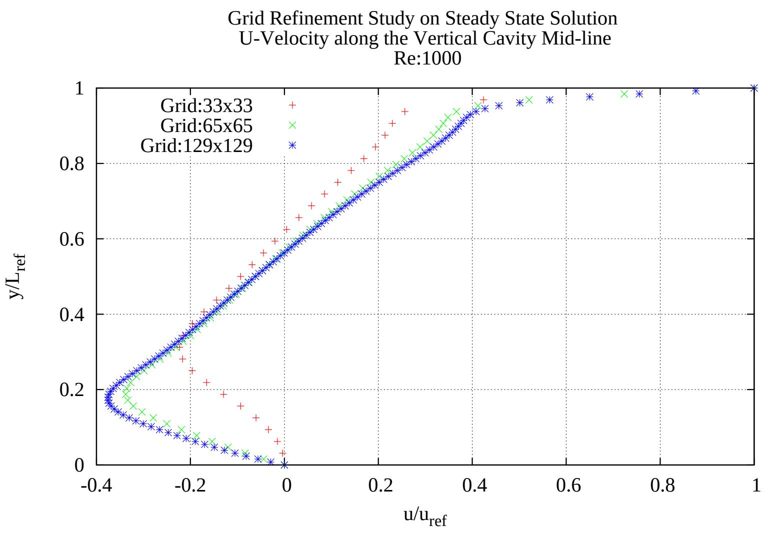

Figure 3.11: Steady state u-velocity as a function of y, along the vertical cavity mid-line, for 3 different grid sizes, at Re:1000.

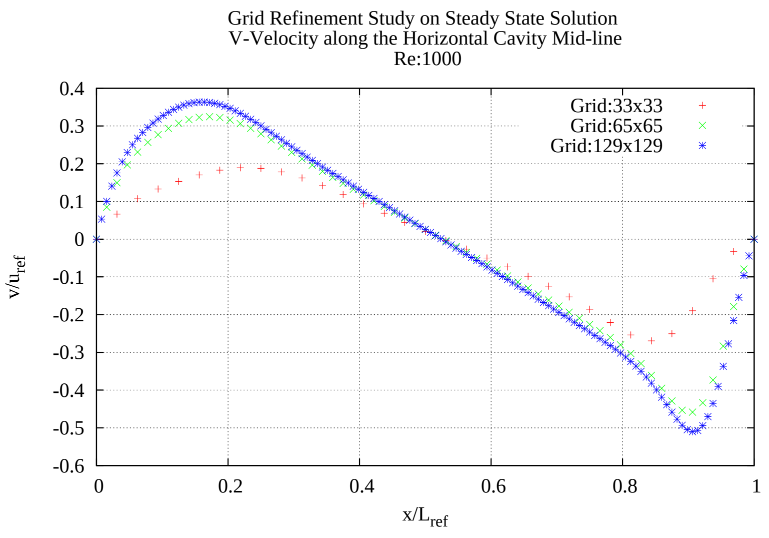

Figure 3.12: Steady state v-velocity as a function of x, along the vertical cavity mid-line, for 3 different grid sizes, at Re:1000.

Chapter 4

Conclusion

The discretization schemes used in this study were shown to be second order accurate. They yielded almost identical results to a similar study by Ghia Et. Al., but slight differences were evident near the walls. It's believed that differences in boundary condition implementation was a major contributing factor to these discrepancies.

The ADI method was used to speed up the computation. Unfortunately, there was a bug in the method, and in the end, the ADI method was far slower than SOR. This bug did not affect the ADI methods final solution, but it drastically increased the time required to converge to a solution. It is suspected that the bug lies somewhere in the optimal time step selection, for fast convergence, or in the method for solving tri-diagonal systems. Further de-bugging is required before any concrete conclusions can be drawn on why the ADI method is so slow to converge.

Some instabilities were observed with a CFL number greater than 0.5. A stability analysis should be performed for the FTCS scheme implemented to determine if this observation is expected. After completing a stability analysis, some criteria should be provided for CFL Number selection. Currently, the largest CFL number yielding a stable solutions is 0.2, and one should be cautious when running the program with larger CFL Numbers.

The program is structured such that it can be adapted to solve problems on a variety of grids. The class constructors, numerical schemes, and VTK data writer must be adapted accordingly. Future work includes solving these equations on a variety of domains involving more complicated geometry. Additionally, the 3D Navier Stokes equations could be implemented. Additionally, the VTK package should be included as a runtime argument. Currently, the program is duplicated to run without VTK, but that version will not stay current with any future modifications.

Bibliography

[1] C. T. Shin U. Ghia, K. N. Ghia. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.