DEVELOPMENT OF AN EDUCATIONAL CFD SOFTWARE FOR TWO DIMENSIONAL INCOMPRESSIBLE FLOWS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÜNEŞ NAKİBOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

JULY 2007

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. S. Kemal İder
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Cüneyt Sert
Supervisor

Examining Committee Members

Prof. Dr. Haluk Aksel (METU, ME) _____

Asst. Prof. Dr. Cüneyt Sert (METU, ME) _____

Assoc. Prof. Dr. Abdullah Ulaş (METU, ME) _____

Asst. Prof. Dr. Merve Erdal (METU, ME) _____

Dr. Mine Yumuşak (ROKETSAN Ind. Inc.) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Güneş NAKİBOĞLU

ABSTRACT

DEVELOPMENT OF AN EDUCATIONAL CFD SOFTWARE FOR TWO DIMENSIONAL INCOMPRESSIBLE FLOWS

Nakiboğlu, Güneş

M.S., Department of Mechanical Engineering

Supervisor: Asst. Prof. Dr. Cüneyt Sert

July 2007, 105 pages

The main purpose of this research is to develop a Computational Fluid Dynamics (CFD) software to be used as an educational tool in teaching introductory level fluid mechanics and CFD courses. The software developed for this purpose is called Virtual Flow Lab. It has a graphical user interface (GUI) that enables basic pre-processing, solver parameter and boundary condition setting and post-processing steps of a typical CFD simulation. The pressure-based solver is capable of solving incompressible, laminar, steady or time-dependent problems on two-dimensional Cartesian grids using the SIMPLE algorithm and its variants. Blocked-cell technique is implemented to extend the types of the problems that can be studied on a Cartesian grid. A parametric study is conducted using a number of benchmark problems in order to test the accuracy and efficiency of the solver and successful results are achieved.

Keywords: CFD, Software-Based Education, Finite Volume Method, Pressure Based Methods, SIMPLE Algorithm, Block-Cell Method.

ÖZ

İKİ BOYUTLU, SIKIŞTIRILAMAYAN AKIŞLAR IÇIN EĞİTİM AMAÇLI BİR HAD YAZILIMI GELİŞTİRİLMESİ

Nakiboğlu, Güneş

Yüksek Lisans, Makina Mühendisliği Bölümü

Tez Yöneticisi: Asst. Prof. Dr. Cüneyt Sert

Temmuz 2007, 105 sayfa

Bu çalışmanın temel amacı, başlangıç düzeyindeki teorik ve hesaplamalı akışkanlar dinamiği (HAD) eğitiminde kullanılabilecek bir HAD yazılımı geliştirmektir. Bu çalışma bünyesinde geliştirilen yazılıma "Virtual Flow Lab" adı verilmiştir. HAD yazılımlarının genelinde bulunan, ön işlem, çözüm ve sınır şartları parametrelerinin tanımlanması ve son işlem aşamalarının görsel olarak gerçekleştirilebileceği bir kullanıcı arayüzü yazılmıştır. Kullanılan basınç temelli çözücü sıkıştırılamayan, laminer, zamana bağlı ve zamana bağlı olmayan akışları iki boyutlu kartezyen çözüm ağlarında SIMPLE ve türevi methodlar kullanarak hesaplama yeteneğine sahiptir. Kartezyen çözüm ağı kullanılabilecek problem çeşitlerinin artırılabilmesi için geliştirilen yazılıma blok hücre tekniği eklenmiştir. Çözücünün doğruluğunu ve verimliliğini test etmek için farklı problemler kullanılarak parametrik bir çalışma gerçekleştirilmiştir ve başarılı sonuçlar alınmıştır.

Anahtar Kelimeler: HAD, Yazılım Tabanlı Eğitim, Sonlu Hacim Metodu, Basınç Temelli Metodlar, SIMPLE Algoritması, Blok-Hücre Metodu.

Dedicating to my family . . .

ACKNOWLEDGMENTS

I would like to express my profound gratitude to my advisor, Dr. Cüneyt Sert, for his advices, comments, and encouragement during my dissertation research. I am also grateful to him for his valuable efforts in every step of this thesis, especially when I was stuck. I give my warmest thanks to him, the best supervisor one can have.

I wish to express my sincere appreciation to Dr. Haluk Aksel, Dr. O. Cahit Eralp and Dr. İlker Tarı for their support and guidance to improve my skills in the field of fluid mechanics during my study at Middle East Technical University.

My heartfelt gratitude must be expressed to Dr. İ. Sinan Akmandor for his valuable advices and trust in me, which motivates me a lot.

I am grateful to each and every member of Propulsion System Design Department in ROKETSAN Missiles Industries Inc. for their encouragement and patience while I was struggling with research; especially my superiors Suzan Koç and Dr. Mine Yumuşak. I am thankful to my colleague Murat Akkuş for his devoted study to compensate my absence in the company.

I also appreciate the useful discussions we had with Emre Gürdamar, Bercan Siyahhan, Ender Özden and Mustafa Akdemir during the preparation of this thesis.

I owe a special thank to Onur Baş for his valuable recommendations and LaTeX assistance.

I also like to express my appreciation to Kayhan İnce and Varlık Kılıç for their continuous help during the code development process.

I want to express my feelings of love to my beloved Esra Özkan, for her patience and spiritual support which kept me going on.

I am always thankful to my elder brother Barış, for not only leading but also forcing me to the right way. You will always be my hero.

Finally, I would like to express my deepest appreciation to my family, what can I possibly say? Twenty five years of your unconditional, dedicated and affectionate support cannot adequately be acknowledged in a few lines. I just want to say that I love you both and I dedicate this dissertation to you.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

ROMAN SYMBOLS

u,v	Velocity components
p	Pressure
$\vec{V}$	Velocity vector
$\vec{n}$	Unit normal vector
$\vec{i}, \vec{j}$	Unit vectors
CV	Control volume
A	Area
S	Source term
b	Mass source term
F	Convective mass flux per unit area
D	Diffusion conductance
A	Area vector
Re	Reynolds number
Pe	Peclet number
Δt	Time step
a	General coefficient term

GREEK SYMBOLS

Φ	General scalar variable
ρ	Density
μ	Kinematic viscosity
Γ	Diffusion coefficient
α	Relaxation parameter
θ	Weighting parameter

SUBSCRIPTS

∞	Freestream values
u	u momentum equation term
v	v momentum equation term
i, I	x face values, x center values
j, J	y face values, y center values
P	Central node
n, N	North face, North cell
s, S	South face, South cell
e, E	East face, East cell
w, W	West face, West cell
nb	All neighbouring nodes
f	All neighbouring faces
knw	Known

SUPERSCRIPTS

$-$	Average
$*$	Guessed or value from the previous iteration
$'$	Difference between two successive iterations
$\circ$	Value of previous time step

CHAPTER 1

INTRODUCTION

1.1 Background

In the chronological sense, Computational Fluid Dynamics (CFD) is a subset of fluid mechanics that uses numerical methods to solve the partial differential equations of continuity, momentum and energy which can not be solved analytically except from some special cases. Depending on the definition used to classify a CFD study, it can be remarked that the first use of CFD was emerged as early as 1940s [1]. However, the first common knowledge uses of CFD appeared and gained prominence during the mid 1950s and early 1960s with the simultaneous development of digital computers.

During the last few decades, the exponential growth of computational sources in storage and execution speeds not only increased the accuracy of the numerical approximations but also enabled the solution of geometrically (e.g. fuselage of an aircraft, automobile body) and physically (e.g. combustion, multiphase) more complex flows. Today, CFD finds so extensive usage in basic and applied research that it can be viewed as a new "third dimension" [2] in fluid dynamics. Although the other two dimensions, experimental and theoretical methods, are still widely used, the trend is clearly toward extensive use of computational methods in design phase which is much cheaper and faster than the conventional methods.

1.2 Numerical Solution Techniques

Among the various numerical methods that have evolved over years, the most commonly used solution techniques are finite difference, finite element and finite volume methods. There are two main differences between these methods; the way which the unknown variables are approximated by means of simple algebraic functions and the discretisation process where the approximations are done to simplify the governing flow equations. The accuracy of the numerical solution depends on the quality of the discretisation, a relation analogous with the one between experimental data and the tools which are used.

Finite difference method (FDM) is a simple and well-established method that solves an equation by approximating continuous quantities as a set of quantities at discrete points. The approximation of the derivatives is the key point of the method. The finite difference is the discrete analog of the derivative which uses finite quantities instead of infinitesimal ones. The main drawback of the method is that it raises difficulties when dealing with complex engineering geometries. However, this method is proved to be succesful in many heat tranfer problems [3].

Finite element method (FEM) considers that the solution region comprises many small, interconnected sub-regions called finite elements. It is possible to systematically construct the approximation functions which is needed to reduce complex partial differentials to simple piecewise (linear or quadratic) functions that are valid in these sub-regions. As a result, a set of algebraic equations for the unknown coefficients of the approximating functions can be developed. Although the method is generally used in structural analysis, where it has been initially developed for [4], the method is later extended to the general field of continuum mechanics [5]. Although most of the commercial CFD packages use finite volume method, there exist a few (E.g. Ansys Flotran,CFdesing) that is based on FEM.

Finite volume method (FVM) is a further refined version of the finite difference method; indeed it is originally developed as a special finite difference method. FVM is one of the most widely

used [6] and versatile discreization techniques used in CFD. Most of the core commercial CFD packages use this technique (e.g. FLUENT, CFX, STAR-CD, FLOW3D). Similar to the FDM, values are calculated at discrete points on a meshed domain. One advantage of FVM over FDM is that it can be used on both structured and unstructured mesh whereas the FDM requires a structured mesh. The detail of the FVM which is used in the current study is explained in Chapter 2.

1.3 Incompressible Flow

Numerical solutions of the governing equations of incompressible flows are in great demand because of their fundamental nature and practical importance. Almost all of the natural flows like biological flows (e.g. blood circulation, respiratory flows), geographical flows (e.g. rivers, oceans) and atmospheric flows (e.g. winds, tornados); most of the engineering flows like low Mach number aerodynamics (e.g. ground and sea vehicles) and hydrodynamics (e.g. pumps, valves) can be classified as incompressible fluid flows. The main assumption of the incompressible flow is constant fluid density which is valid not only for liquids but also for gases which have low Mach numbers (e.g. $M < 0.3$). This assumption both helps and hinders numerical solutions of Navier-Stokes equations(conservation of mass and momentum). The constant density assumption simplifies the equations and decouples the energy equation from the momentum and continuity equations so that if the problem involves heat and work interactions then the energy equation can easily be solved seperately because of the decoupling. However the main difficulty in numerical solutions also arises from this assumption which is the lack of time evolution term for pressure in the continuity equation. As a result, pressure field is decoupled from the velocity field and the continuity equation becomes just a constraint. Numerous techniques have been developed to overcome this problem and to solve incompressible Navier-Stokes equation which is covered in the subsequent chapters.

1.4 Methods For Solving Incompressible Viscous Flows

The literature on numerical solutions of the incompressible Navier-Stokes equations is so overwhelming that writing a complete review of numerical methods that has been developed so far is an impossible task. The current attempt is nothing more than a broad classification of some of the earlier developments in the era of viscous incompressible flows, before discussing the scope of the current study.

Depending on the variables used in the system of equations, solution approaches for solving incompressible Navier-Stokes equations can be classified into two categories,

- Primitive Variable Approaches ,
- Non-primitive Variable Approaches .

The primitive variable approaches, in which the system of equations is written with pressure and velocity components as independent variables, for the solution of incompressible Navier-Stokes equations, can be further classified.

- Pressure Based ,
  - The Marker and Cell (MAC) method
  - Fractional Step Method
  - Semi-Implicit Method for Pressure Linked Equations (SIMPLE)
- Density Based.
  - Artificial Compressibility Method

The Marker and Cell (MAC) method originally developed by Harlow and Welch [7] was the first primitive variable approach in literature. The usual procedure in this method is to assume an initial pressure field as a mapping parameter, and then follow an iterative process until the continuity equation is satisfied. The major drawback of this method is having to solve a Poisson equation for pressure which is formed taking the divergence of the momentum equations. Even

though this method is widely accepted and successfully applied to many problems, the solution accuracy and performance is highly dependent on the performance of the pressure Poisson equation solver which is the bottleneck of the method. Implementation of the MAC method in complex 3D geometries is extremely cumbersome and not practical. Additional information and examples of the method can be found in literature, [8][9][10].

The next pressure iteration based primitive-variable approach which has been used extensively is the fractional-step method originally introduced by Chorin [11]. This method advances the solution in time using two steps; in the first step an intermediate velocity field is solved using the momentum equations in which the pressure gradient term which is computed from the previous time step can be used. In the second step, the pressure is computed which will map the intermediate velocity field into a divergence free field, thus the solution for the next time level is determined [12]. The second step requires the solution of a pressure Poison equation as it is in the MAC method and similarly the efficiency of the fractional step method depends highly on the Poisson solver.

Another pressure based method mentioned in this context is the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE). This method is developed and implemented by Patankar and Spalding [6][13]. The major virtue of this method is the simple way of estimating velocity correction. Although this feature simplifies the calculation considerably, it introduces empiricism into the method which is called as relaxation parameters. The method has gone through numerous modifications to improve the technique. SIMPLER (SIMPLE Revised) algorithm of Patankar [6] and SIMPLEC (SIMPLE-Consistent) algorithm of Van Doormal and Raithby [14] are among the well known modifications of SIMPLE.

The last method discussed in the class of primitive variable approaches is the artificial compressibility method proposed by Chorin [15] for steady problems and later extended to unsteady flows by Peyret [16]. Unlike other primitive variable methods a pseudo time derivative of pres-

sure is introduced into the continuity equation changing the mathematical character of the continuity equation from elliptic to hyperbolic. This change enables the system of equations to be solved with a variety of time marching schemes previously developed for compressible flow solvers. There exist numerous applications of artificial compressibility in the literature both for steady and unsteady flows [17][18].

Other than the primitive variable approaches, there exist several methods in which the non-primitive variables are used as independent variables to formulate incompressible Navier-Stokes equations. The primary variables, pressure and velocity components, are calculated later on as dependent variables. The main advantage of the non-primitive formulation is the complete decoupling between the velocity and pressure calculations. Vorticity-stream function [19][20] method is the most widely-used non-primitive variable approach. This method has a unique advantage of inherently satisfied continuity equation; however this solution technique is applicable to two dimensional computations only. This constraint comes from the stream function which is defined in two dimensions and does not exist in three dimensions. However different formulations have been developed to extend non-primitive variable approach to three dimensions, including vorticity - velocity approach [21][22], vector - potential approach [23], and vector stream function approach [24]. The major drawback of these methods is the implementation of boundary conditions which are quite challenging.

In this study, the SIMPLE algorithm and their derivatives are used. These algorithms are perhaps the most widely referred and reliable methods for the solution of incompressible flows, indeed most of the commercial CFD packages use these methods. Although the current study is limited for two dimensional, incompressible flows; the research project that involves this study has been planned to develop a solver which is capable of solving three dimensional, all-speed (Incompressible & Compressible) flows. Thus, during the process of selection for appropriate technique which will be employed in the solver, these facts are taken into account. In incompressible flows, the listed methods have similar accuracy and efficiency [25] but only the

SIMPLE algorithm and its derivatives can easily be extended to all speed flow regimes in three dimensional space without major changes in the methodology [26].

1.5 Iterative Solvers

Most of the governing equations of fluid mechanics and heat transfer are second-order partial differential equations (PDE) which turn into a system of algebraic equations after the discretization process. Depending on the nature of partial differential equations, the algebraic equation system can be linear or non-linear.

Linear algebraic systems can be solved using both direct and iterative methods, however for the non-linear case an iterative scheme has to be followed to solve the non-linearity, even if the system is solved by a direct method at each iteration step.

Direct methods are applied to find the exact solutions (within the accuracy allowed by the round-off errors) of linear algebraic systems using finite number of arithmetic operations. Iterative methods, on the other hand, are based on a succession of approximate solutions, and then provide an accuracy based on a user-specified tolerance level.

In practice, direct methods are generally not preferred since the discretization errors are usually much larger than the round-off errors (truncation errors). Besides, direct methods need enormous amount of arithmetic operations to produce a solution. These methods are only applicable to small systems without becoming clearly disadvantageous than iterative methods [27]. They also do not take the advantage of initial guess of the solution. Gauss elimination and LU decomposition are among the well known direct methods.

In fluid mechanic problems, mostly the fine meshes are used in order to resolve the physics of the flow so that direct methods are seldom applied. The idea behind the iterative methods is to perform a simple guess-and-correct procedure. The procedure starts with an initial guess

then new values are computed using this guessed fields; based on this new computed values, a newer set of values are sought, and the loop is repeated until a predefined convergence criterion is satisfied.A large number of iterative techniques are available in literature with different convergence speeds and complexity levels.

The simplest iterative method is the Jacobi method, which is a point iterative method where the initially guessed or previously computed values of the neighboring nodes are used to calculate the new value of the dependent variable. This method is rarely used in the solution of elliptic equations because, depending on the convergence definition, this method can be more expensive than a direct solver.

Another well known basic iterative method is the point Gauss-Seidel method, which is nothing but a simple extension of the Jacobi method. Different than the Jacobi method, the newly computed values of the dependent variables are immediately used for the calculation of neighboring nodes in the same iteration step. This simple modification speeds up the convergence rate by twice [28].

The line Gauss-Seidel iteration method (Line by line TDMA) is similar to the point Gauss-Seidel iteration method however in this case there exist three unknown node values at each equation which is formed and it results in a tri-diagonal coefficient matrix. For this often encountered system, a very efficient direct solver known as the Thomas algorithm or Tri-diagonal matrix algorithm (TDMA) is applied. This method converges faster than the point Gauss-Seidel iteration method by a factor of 3/2 [29] as expected. In the current study this iterative method is used and the details will be covered in the subsequent chapters.

Successive over-relaxation method is another technique that is developed to further increase the convergence rate. This method has two variants; point successive and line successive over-relaxation as it was in the Gauss-Seidel method. The idea behind this iterative method is

straightforward. If trend in the computed values is captured during the iterations, then this trend can be used to extrapolate the values for the next iteration which will accelerate the solution procedure. The key point of success in the over-relaxation methods lies in the determination of optimum relaxation parameter. However there exists no general guidelines for this purpose, therefore for most cases, numerical experimentation is performed.

There exist also many other efficient solver algorithm like strongly implicit procedure (SIP) and alternating direction implicit (ADI). SIP is specifically developed for algebraic equations derived from discretized partial differential equations, and it is not applicable to general system of equations. Details of the method can be found in Stone 1968 [30]. ADI is another common method for elliptic problems although it is difficult to be optimized for general problems. A discussion of this topic can be found in Hageman and Young (1981) [31]

The last iterative method that is mentioned in this context is the multi-grid method which is the most efficient and general iterative technique known today [27]. This method is initially developed for the solution of elliptic partial differential equations [32][33], for which the near-optimum convergence characteristics have been demonstrated. Later the method is extended to improve other PDE solvers. The multi-grid idea is based on two principles: error smoothing and coarse grid correction. Throughout the solution process, meshes of different intensities are used to optimize the error smoothing using the coarser meshes whenever possible to reduce the computational cost. Multi-grid methods have shown substantial improvement in the solution of incompressible Navier-Stokes equations using pressure based algorithms [34][35][36][37].

1.6 Educational Aspect

1.6.1 Motivation

Although CFD is used extensively both in basic and applied science, the use of CFD in engineering education is mostly limited to graduate level courses where the mathematical background necessary to write CFD programs is taught. Although recent undergraduate level fluid me-

chanics books involve CFD related chapters, references indicating the use of CFD as a teaching aid are limited. This study is about the development of necessary software to use CFD as i) an educational aid in undergraduate level fluid mechanics courses and ii) a practicing tool in introductory level CFD related courses.

In many engineering departments students take their first fluid mechanics course in their second or third year. Although for some of the departments a single semester course is enough, usually undergraduate level fluid mechanics is taught as a two semester course. Typical outline of a two semester fluid mechanics course is given below

1. Fluid statics
2. Integral analysis of fluid motion (conservation of mass, momentum and energy)
3. Bernoulli equation
4. Fluid kinematics
5. Differential analysis of fluid motion
6. Similitude and dimensional analysis
7. Viscous flows in pipes and channels
8. Flow over immersed bodies
9. Introduction to compressible flow
10. Introduction to turbomachines

Due to the inherent complexity of fluid motion, fluid flow problems require a different viewpoint compared to solid mechanics problems. Understanding the topics like continuum assumption and its validity, proper comprehension of the field concept such as the velocity or the pressure field, making the switch from the classical Lagrangian approach, which is taught in earlier statics and dynamics courses, to the Eulerian approach, establishing the link between these two different view points, mathematical and physical understanding of the convective derivatives are some of the challenges that the students face with when they begin studying fluid mechanics.

Other than the above mentioned mathematical modeling related difficulties, students also get confused due to the simple fact that it is hard to observe and comprehend the behavior of fluids in everyday life. For example students often do not question the way a beam deflects under the action of certain forces, but they can easily get confused in a simple pipe flow problem, where the existence of viscous forces cause a pressure drop but not a velocity drop. Or they comfortably take apart a complicated solid structure into its simple elements by drawing free-body-diagrams with proper reaction forces and moments, but it is not so easy for them to work with an imaginary control volume that is open to mass, momentum and energy transfer. Or for example it is difficult to mentally visualize the way the properties, such as viscosity, of gas changes while it is being heated. In addition to these, students get quite puzzled when they learn that almost all practically important fluid flow applications involve turbulence, which is considered to be one of today's most challenging physical phenomena.

Another major difficulty in learning fluid mechanics is the necessity of proper simplification of a given problem. Fluid flow problems of engineering importance can be so complex that one almost always needs to make a number of simplifying assumptions in order to be able to approach it by analytical means. For example the analytical solution of conservation of linear momentum, in other words the Navier-Stokes equations, is only possible for a few very simple problems. Neglecting the viscous affects, these equations reduce to Euler equations, which are still quite difficult to solve. Another simplification comes when we consider the Euler equations along a streamline, which leads to the Bernoulli equation. Other than these, one might need to consider if the compressibility of the fluid or the unsteadiness of the flow is of importance or not. It is not easy to get comfortable with the use of these different levels of simplifications.

Visualizing the fluids in action is a very informative tool that helps overcome the above mentioned difficulties to a certain degree. Carefully designed educational experimental studies are important in this respect. Many of the recently published fluid mechanics textbooks come with

discs that include movies of many interesting fluid flow phenomena [38]. It is also possible to find excellent series of educational fluid mechanics films, such as the ones prepared by NCFMF and IIHR [39], [40]. Today another alternative is the use of CFD simulations as an educational tool. CFD enables us to solve fluid flow problems numerically by computers. One simple advantage of this is its power in attracting the attention of today's computer oriented students [41]. But the actual benefit is that students feel comfortable when they see that the governing equations, which are known to be impossible to solve analytically in most cases, can actually be solved with an acceptable engineering accuracy. The importance of such a numerical study is quite different than the importance of performing experiments. Experimentation is very valuable in understanding the underlying physics of a certain problem. This is necessary to establish and validate a mathematical model. But then we naturally feel the need to solve that mathematical model. If we can not do that, our model is not very useful. If, for example, we can not solve the Navier-Stokes equations by any means, then the value of being able to derive these equations becomes questionable. Today CFD serves as an important tool to test the validity of our mathematical models for almost all types of flow problems.

1.6.2 Sample CFD Simulation Ideas to be Used in an Undergraduate Fluid Mechanics Course

The outline of a typical undergraduate level fluid mechanics course was mentioned in the previous part. In this part the level of support that CFD simulations might provide in understanding the topics of this outline will be discussed.

Fluid statics is easy to learn since it involves no fluid motion. CFD simulations can still be used to explain the pressure distribution in a static fluid. Students can perform a number of numerical experiments with different shaped cups or tubes to see how the pressure increases linearly in a direction opposite to the gravitational acceleration independent from the shape of the container.

In integral analysis of fluid motion, students can check the mass conservation inside a box with multiple inlets and exits. They can be asked to use the numerical data of a CFD simulation of external flow over a body to calculate the drag and lift forces acting on the body. Comparing the forces calculated with a standard integral approach that uses the numerically obtained velocity profiles against the forces directly calculated by the differential approach of a CFD simulation could be quite instructive.

The Bernoulli equation is one of the most useful but also the most misused equations of fluid mechanics. It can be demonstrated by the simulation of fluid leaving a tank through a number of openings at different elevations. The concepts of static, dynamic and total pressure, which are difficult to grasp, can be explained by examining the result of a converging diverging duct simulation at different locations. CFD simulations could be excellent ways to discuss the use of Bernoulli equation in flowrate measuring devices, such as an orifice meter or a venturi meter.

While discussing fluid kinematics, movies of rotational and irrotational flow simulations can be used. Results of the simulation of a developing flow that enters a channel uniformly can be used to demonstrate the effect of viscosity and shear forces in the creation of rotational motion inside the boundary layers. Deformation of a straight line or square shaped fluid element can be visualized in different flow fields to understand the type of deformation that they go through.

About the differential analysis of fluid motion, inviscid and viscous simulations inside a number of different geometries can be considered. Analytical solutions of Couette, Poiseulle and Hagen-Poiseulle flows can be compared with the numerical ones. Possible sources of differences between numerical and analytical results can be discussed along with the limitations of numerical simulations.

CFD simulation of a carefully designed piping system with different sized pipes and a number of bends, expansions, contractions, etc. will be very valuable in understanding the major

and minor frictional losses and related pressure drops, as well as the application of extended Bernoulli equation for the calculation of these quantities.

Lectures about external flows over immersed bodies can be supported by a simple boundary layer growth demonstration over a flat plate and comparing the results with the analytical ones. It is also possible to visualize the solution of flow fields around streamlined and blunt bodies and discuss the affect of streamlining a body through the inspection of numerically calculated shear and pressure drag forces. A discussion about the D'alambert's paradox can be made comparing inviscid and viscous flow simulations.

CFD can also be used to enhance the understanding of compressible flow lectures. Simulations of a number of flows inside a converging-diverging de Laval nozzle can be used to see different flow regimes, formation of shocks and choking.

Of course the list of problems mentioned above is quite long and it is not possible to make use of all of them in a single or even in a two semester fluid mechanics course. Each problem should be considered one by one from an educational standpoint. Here the interest is not just being able to solve these problems numerically. They have to be designed in a way that they clarify the issues which are harder to understand through other means. Depending on the CFD software to be available, templates can be prepared for these problems and step by step instructions can be provided to the students to perform relatively easy simulations. For many of them, results of the simulations can directly be provided and the students can be asked only to perform the postprocessing step and then discuss the results.

Today it is possible to find a number of commercial CFD software in the market that can successfully perform all the simulations mentioned above. However as mentioned in the previous paragraph, when it comes to using CFD as an educational tool, a software tailored specifically for this purpose would be a better choice. One advantage of using such a software will be its

cost. Due to the relative simplicity of the problems that such a software is expected to be used for, its capabilities would be limited compared to a full featured CFD software, resulting in a more economical product. Also the price would be affected by the fact that the software would be used for educational purposes and it should be affordable by the students. Such educational software should also be easy to learn and use. It should support the use of templates, problems that are designed parametrically and created by the instructor to be used in a series of numerical experiments. Templates can be prepared in as much detail as the instructor wants. For example to study the drag characteristics of a certain object placed in a flow field, a template can be created based on the parameters such as the upstream fluid velocity, viscosity of the fluid and the dimensions of the object. Students can use such a template to easily perform a number of numerical tests to see the effect of different parameters. Mesh generation, solution and post-processing steps can all be predefined in the template, which means that the student can obtain the results by clicking a button after setting the problem parameters to the desired values. For an undergraduate level fluid mechanics course the details of mesh generation and the selection of solver parameters should be hidden from the students as much as possible.

When the literature about the use of CFD for educational purposes is investigated, it is easy to see that the software alternatives are very limited. Most of the studies use FlowLab [42], [43], which is a tailored version of the popular commercial software Fluent. Flowlab is designed to be used in teaching both fluid mechanics and CFD itself. CFD Studio is another example of CFD software written to be used as an educational tool [44]. In this study a new software called Virtual Flow Lab is developed.

1.7 Current Study

The main goal of this research is to develop a CFD software to be used as an educational tool in teaching introductory level fluid mechanics and CFD courses. The software developed for this purpose is called as Virtual Flow Lab (VFL). Being different than most of the other CFD related academic studies, which are composed of bare solvers only, Virtual Flow Lab is written

to be a complete CFD package. Simple but functional elements of a typical CFD software, pre-processor, solver and post-processor, are combined in a carefully designed graphical user interface.

VFL is distributed under the GPL license [45], which means that it can freely be downloaded from the internet. The source codes can also be downloaded and if necessary they can be modified by the users according to their specific needs. It is written in C++ language. User interfaces are designed with Qt [46], a C++ user interface library is freely available from Trolltech.

VFL is a platform independent software which can be compiled and run under different computer architectures without any or very little source code change. It is successfully tested under Linux and Windows operating systems. It is a multi-lingual software, which means that its graphical interface can be used in more than one language.

The heart of VFL, the flow solver, is a pressure-based solver which is capable of solving incompressible, laminar, steady or time-dependent problems on two-dimensional Cartesian grids using the SIMPLE algorithm of Patankar [6]. Two variants of SIMPLE, namely SIMPLER and SIMPLEC, are also available as alternative methods. Space discretization can be made using one of the central, upwind, power-law or hybrid schemes. It is possible to perform steady or unsteady simulations using iterative and time-marching algorithms. Block-off region technique of Patankar is implemented to extend the types of the problems that can be studied on Cartesian grids.

Next chapter of this thesis will provide detailed information on the numerical techniques used to develop the solver of Virtual Flow Lab. Chapter 3 will introduce the graphical user interface of the software. The performance of VFL in solving a number of classical bencmark problems can be found in Chapter 4. The final chapter will summarize the work and provide a route for future development of VFL.

CHAPTER 2

NUMERICAL METHOD

2.1 Governing Equations

In order to simulate fluid flow and heat transfer, it is necessary to describe the associated physics in mathematical terms. The extend of the mathematical model depends on the characteristics of the flow of interest. In this particular study two dimensional, incompressible, laminar flows of Newtonian fluids with constant viscosity are of interest. Governing equations of this type of flows can be derived from the following general transport equation of a scalar property ϕ

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{Transient} + \underbrace{\nabla\cdot(\rho\vec{V}\phi)}_{Convective} = \underbrace{\nabla\cdot(\Gamma\nabla\phi)}_{Diffusive} + \underbrace{S_\phi}_{Source} \tag{2.1}$$

where ρ, Γ and $\vec{V}$ are the density, diffusivity and the velocity field of the fluid in which ϕ is transported and t is the time.

Equation (2.1) highlights various modes of transport of the property ϕ. The first and the second terms on the left hand side are the transient and the convective terms, respectively. The first term on the right hand side is the diffusive term. Last term is the source term, which is the collection of the terms that can not be expressed in the form of the others.

Governing equations of two-dimensional, incompressible flows, namely the continuity and two

components of the linear momentum equation can be derived from Equation (2.1) by setting ϕ equal to 1, u and v respectively

Continuity equation,

$$\frac{\partial(\rho)}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{2.2}$$

u-momentum equation,

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho \vec{V} u) = \nabla \cdot (\mu \nabla u) - \vec{i} \cdot \nabla p + S_u \tag{2.3}$$

v-momentum equation,

$$\frac{\partial(\rho v)}{\partial t} + \nabla \cdot (\rho \vec{V} v) = \nabla \cdot (\mu \nabla v) - \vec{j} \cdot \nabla p + S_v \tag{2.4}$$

where u and v are the components of the velocity vector in x and y directions of the Cartesian coordinate system, respectively. μ is the kinematic viscosity of the fluid and p is the pressure. Source terms of the momentum equations have the units of force per unit volume and they physically represent different forces acting on the fluid such as the gravitational force due to the weight of the fluid. Pressure gradient terms are actually part of the source terms, however they are usually considered seperately to emphasize their importance. Equations (2.2), (2.3) and (2.4) form a set of three, coupled, non-linear partial differential equations that need to be solved for three unknowns, u, v and p. Numerical solution of these equations require two different levels of discretization that will be described in the following sections.

2.2 Geometry Discretization

In a typical numerical solution, continuous velocity and pressure fields need to be determined at a finite number of discrete points in space (and in time for time dependent problems). The collection of these discrete points is called as mesh (grid), which is generated during the pre-processing step of the numerical solution. In this study two-dimensional problem geometries are discretized by introducing Cartesian grids as shown in Figure 2.1. As seen in the figure, two-dimensional Cartesian grids are made up of 4-noded square and/or rectangular regions called

as cells (elements). During the discretization of the governing equations the cell of interest is denoted by the letter P. Neighboring cells are called as W (West), E (East), S (South) and N (North). Lowercase letters w, e, s and n are used to denote the faces of the cell P. Face area vectors and normals are given as

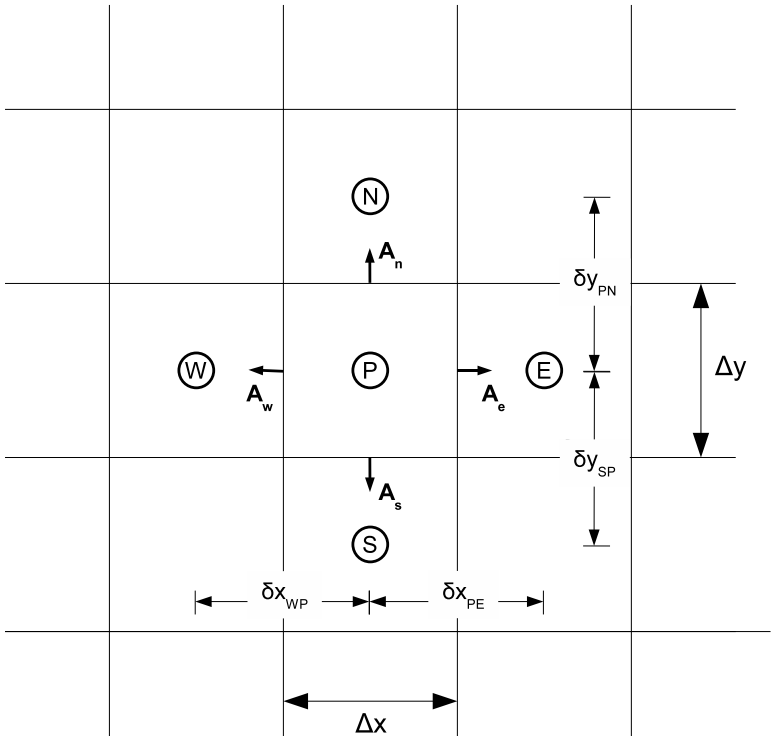

Figure 2.1: Two Dimensional Control Volume

$$\begin{aligned} \boldsymbol{A_e} &= \Delta y\ \boldsymbol{n_e} = \Delta y\ \vec{i} \\ \boldsymbol{A_w} &= \Delta y\ \boldsymbol{n_w} = -\Delta y\ \vec{i} \\ \boldsymbol{A_n} &= \Delta x\ \boldsymbol{n_n} = \Delta x\ \vec{j} \\ \boldsymbol{A_s} &= \Delta x\ \boldsymbol{n_s} = -\Delta x\ \vec{j} \end{aligned} \tag{2.5}$$

There are different approaches about the locations where the unknowns are stored. Colocated and staggered grid arrangements are the two common choices that will be explained in detail in the coming sections.

2.3 Equation Discretization

As mentioned earlier the three governing equations of interest are partial differential equations. In order to reduce them to a set of linear algebraic equations, the derivatives appearing in all three equations need to be discretized. This step makes use of approximate derivatives that involve the unknowns associated with the discrete points of the numerical grid. Here it is necessary to mention that a semi-discrete finite volume based discretization is used in this study. Semi-discrete means that the discretization of the derivatives with respect to space and time are treated seperately. The following sections provide details of the discretization of the time-independent forms of the governing equations. Time-dependent terms will be considered later.

2.3.1 Discretization of the Momentum Equation

The discretization of u- and v-momentum equations are very similar. The process is explained in detail for the u-momentum equation, which is given as follows for steady flows

$$\nabla \cdot (\rho \vec{V} u) = \nabla \cdot (\mu \nabla u) - \vec{i} \cdot \nabla p + S_u \tag{2.6}$$

Discretization begins by integrating Equation (2.6) over an arbitrary control volume CV (which is nothing but any cell of the numerical grid), the unique property of control volume method which distinguishes it from all other CFD techniques. The integrated form of the u-momentum equation is given as

$$\int_{CV} \nabla \cdot (\rho \vec{V} u) d\forall = \int_{CV} \nabla \cdot (\mu \nabla u) d\forall + \int_{CV} -\vec{i} \cdot \nabla p dV + \int_{CV} S_u d\forall \tag{2.7}$$

Using the Gauss' divergence theorem, the convective and diffusive terms can be expressed as integrals over the entire surface of the control volume

$$\int_{A} (\rho \vec{V} u) \cdot d\boldsymbol{A} = \int_{A} (\mu \nabla u) \cdot d\boldsymbol{A} + \int_{CV} -\vec{i} \cdot \nabla p d\forall + \int_{CV} S_u d\forall \tag{2.8}$$

Details of the discretization of the diffusive, convective and source terms are given in the following sections.

2.3.1.1 Discretization of the Diffusive Term

Consider the discretization of the diffusive (viscous) term for the control volume corresponding to cell P of Figure 2.1. The area integral of the diffusive term should be evaluated over the surface of cell P, which is composed of four faces denoted by the letters e, w, n and s. The basic assumption used is that the diffusive flux vectors may be represented by their values at the face centroids as follows

$$\int_A (\mu \nabla u) \cdot \boldsymbol{dA} = \sum_{f=e,w,n,s} (\mu \nabla u)_f \cdot \boldsymbol{A_f} \tag{2.9}$$

where the subscript f denotes the cell faces. Equation (2.9) can now be written in the following expanded form

$$\begin{aligned} &\int_A (\mu \nabla u) \cdot \boldsymbol{dA} = \\ &(\mu \frac{\partial u}{\partial x})_e \boldsymbol{n_e} \cdot \boldsymbol{A_e} + (\mu \frac{\partial u}{\partial x})_w \boldsymbol{n_w} \cdot \boldsymbol{A_w} + (\mu \frac{\partial u}{\partial y})_n \boldsymbol{n_n} \cdot \boldsymbol{A_n} + (\mu \frac{\partial u}{\partial y})_s \boldsymbol{n_s} \cdot \boldsymbol{A_s} \end{aligned} \tag{2.10}$$

To complete the discretization process of the diffusive term, the derivatives of Equation (2.10) are approximated using central differencing as follows,

$$\begin{aligned} \left(\frac{\partial u}{\partial x}\right)_e &= \left(\frac{u_E - u_P}{\delta x_{PE}}\right) \\ \left(\frac{\partial u}{\partial x}\right)_w &= \left(\frac{u_P - u_W}{\delta x_{WP}}\right) \\ \left(\frac{\partial u}{\partial y}\right)_n &= \left(\frac{u_N - u_P}{\delta y_{PN}}\right) \\ \left(\frac{\partial u}{\partial y}\right)_s &= \left(\frac{u_P - u_S}{\delta y_{SP}}\right) \end{aligned} \tag{2.11}$$

Due to the direction-independency of the diffusion process central differencing is usually preferred. Defining a variable D to represent the diffusivity at cell faces,

$$\begin{aligned} D_e &= \frac{\mu}{\delta x_{PE}} \\ D_w &= \frac{\mu}{\delta x_{WP}} \\ D_n &= \frac{\mu}{\delta y_{PN}} \\ D_s &= \frac{\mu}{\delta y_{SP}} \end{aligned} \tag{2.12}$$

and substituting Equations (2.5), (2.11) and (2.12) into Equatin (2.10) the complete discrete form of the diffusive term is obtained as

$$\int_A (\mu \nabla u) \cdot \boldsymbol{dA} =$$
$$D_e A_e (u_E - u_P) - D_w A_w (u_P - u_W) + D_n A_n (u_N - u_P) - D_s A_s (u_P - u_S) \quad (2.13)$$

2.3.1.2 Discretization of the Convective Term

The discretization process of the convective term needs special attention due to two basic differences compared to the diffusion term. First of all convective term is nonlinear since the velocity vector $\vec{V}$ contains the unknown velocity component u as well. Other than this, unlike diffusion, convection process spreads the information of the flow field only in the flow direction. Many different discretizations can be derived depending on the ability to use the flow direction information.

Consider the convective term of Equation (2.8) integrated over the faces of the cell P of Figure 2.1. Similar to the diffusion term, it is assumed that convective flux is constant over each of the faces of cell P and that the face centroid value is representative of the face average. With this assumption the convective term can be written as

$$\int_A (\rho \vec{V} u) \cdot d\boldsymbol{A} = \sum_{f=e,w,n,s} (\rho \vec{V} u)_f \cdot \boldsymbol{A_f} \quad (2.14)$$

Defining a new variable F to represent the mass flux per unit area,

$$\begin{aligned} F_e &= (\rho u)_e \\ F_w &= (\rho u)_w \\ F_n &= (\rho v)_n \\ F_s &= (\rho v)_s \end{aligned} \quad (2.15)$$

the convective term can expressed as

$$\int_A (\rho \vec{V} u) \cdot d\boldsymbol{A} = F_e A_e u_e - F_w A_w u_w + F_n A_n u_n - F_s A_s u_s \quad (2.16)$$

Note that the nonlinearity of the momentum equation is due to the existence of the unknown velocities in the F terms. In this study Equation (2.16) is linearized using the Picard method,

which calculates the F terms using the available velocities from previous iterations.

Equation (2.16) requires the velocity components at the cell faces. This was not the case for the diffusive term which needed the derivatives of the velocity components at the cell faces. Four commonly used techniques to approximate cell face velocities are discussed next.

2.3.1.3 Approximation of the Face Velocities with Central Differencing

Central differencing is already used to represent the cell face velocity gradients of the diffusive term through Equations (2.11). It is possible to use it for approximating cell face velocities too. Assuming that u varies linearly between grid points, cell face velocities can be written as follows

$$\begin{aligned} u_e &= \frac{u_E + u_P}{2} \\ u_w &= \frac{u_W + u_P}{2} \\ u_n &= \frac{u_N + u_P}{2} \\ u_s &= \frac{u_S + u_P}{2} \end{aligned} \tag{2.17}$$

These can be inserted into Equation (2.16) to obtain the following discrete form of the convective term,

$$\int_A (\rho \vec{V} u) \cdot d\boldsymbol{A} = F_e A_e \frac{u_E + u_P}{2} - F_w A_w \frac{u_W + u_P}{2} + F_n A_n \frac{u_N + u_P}{2} - F_s A_s \frac{u_S + u_P}{2} \tag{2.18}$$

One advantage of central differencing is the second order space accuracy it provides, which means sharper drop of errors due to gird refinement compared to first order schemes. However, central differencing scheme has two major drawbacks. First one is its non-transportive nature which hinders the scheme to recognize the flow direction or the strength of convection relative to diffusion. This is especially important for convection dominated (high Reynolds number) flows. The second drawback is its chance to violate the Scarborough criteria [47]. The criteria can be stated as follows,

$$\begin{aligned} \sum |a_{nb}| &\leq a_P \qquad \text{at all nodes of the grid} \\ \sum |a_{nb}| &< a_P \qquad \text{at one node of the grid at least} \end{aligned} \tag{2.19}$$

Here the term a_P denotes the coefficient of the central node (P) and the summation in the numerator is taken over all the neighbouring nodes (nb). In the subsequent chapters, the coefficients a_P, a_{nb} are obtained when the discretization process for the momentum equations is fully completed. Scarborough has shown that coefficient matrices that are produced by differencing schemes which satisfy this criteria are always diagonally dominant which is an important property for the speed and convergence characteristics of iterative solution methods. In deed, this criteria is a sufficient condition for convergent iterative methods.

2.3.1.4 Approximation of the Face Velocities with Upwind Differencing

The main advantage of upwind differencing is related to the major drawback of central differencing, the transportiveness. As the name implies, upwind differencing takes the flow direction into account. Indeed the cell face value of u is determined by using only the upstream node value, as given below

$$\begin{aligned} u_e &= \left\{ \begin{array}{llll} u_P & if & F_e > 0 \\ u_E & if & F_e < 0 \end{array} \right\} \\ u_w &= \left\{ \begin{array}{llll} u_W & if & F_w > 0 \\ u_P & if & F_w < 0 \end{array} \right\} \\ u_n &= \left\{ \begin{array}{llll} u_P & if & F_n > 0 \\ u_N & if & F_n < 0 \end{array} \right\} \\ u_s &= \left\{ \begin{array}{llll} u_S & if & F_s > 0 \\ u_P & if & F_s < 0 \end{array} \right\} \end{aligned} \tag{2.20}$$

For example, when the flow is crossing cell P in the positive x direction such that $u_w > 0$ ($F_w > 0$) and $u_e > 0$ ($F_e > 0$) than the information is carried from left to the right. In such a case upwind differencing will predict face velocities of $u_e = u_P$ and $u_w = u_W$.

Substituting Equation (2.20) into Equation (2.16) gives the following discrete form of the con-

vective term

$$\int_A (\rho\vec{V}u)\cdot d\boldsymbol{A} = \quad A_e(max[F_e,0]\ u_P + max[-F_e,0]\ u_E)$$
$$-A_w(max[F_w,0]\ u_W + max[-F_w,0]\ u_P)$$
$$A_n(max[F_n,0]\ u_P + max[-F_n,0]\ u_N)$$
$$-A_s(max[F_s,0]\ u_S + max[-F_s,0]\ u_P) \tag{2.21}$$

Equations (2.20) inherently satisfies the Scarborough criteria, which indicates good convergence characteristics. However, the accuracy of the scheme is only first order. Another drawback of the upwind scheme is its diffusive nature [28]. This problem arises especially when the flow is not alined with the grid lines, and has various names in the literature such as artificial, numerical or false diffusion. Similar to the difficulty of central differencing, this difficulty can be resolved by grid refinement. However, this cure always brings the computational efficiency and resources into consideration.

In the literature there exist some other first-order schemes for the discretization of convection and diffusion terms. In this study other than central and upwind differencing schemes, which treat the convection and diffusion terms separately in the discretization process, two other approximations are used, namely hybrid and power-law schemes.

2.3.1.5 Hybrid and Power Law Schemes for the Discretization of Diffusive and Convective Terms

To understand the hybrid scheme, it is necessary to introduce an important flow variable called the Peclet number (Pe). It measures the relative importance or dominance of convection and diffusion in the transport of a scalar variable.

$$Pe = \frac{F}{D} \tag{2.22}$$

If it is based on cell length scale, δx (or δy) it is called as cell Peclet number. It should be noted that in the context of momentum equation as it is the case here, the cell Peclet number

corresponds to cell Reynolds number.

$$Pe = \frac{F}{D} = \frac{\rho u}{\Gamma/\delta x} = \frac{\rho u \delta x}{\mu} = Re \tag{2.23}$$

Note that for two-dimensional problems it is possible to define another Pe which is based on δy, or even a third one which is based on a linear length scale that uses a combination of both δx and δy. Convection dominated problems are associated with high Peclet numbers which are usually considered to be more challenging to solve numerically compared to diffusion dominated problems.

As described by Patankar and Spalding [13], hybrid differencing is based on a piecewise formula that depends on the face Peclet number and provides a combination of central and upwind differencing. For low Peclet numbers ($Pe < 2$) evaluated at a cell face hybrid differencing uses the second order accurate central differencing for both convection and diffusion through that face. For relatively high Peclet numbers, ($Pe \geq 2$), upwind differencing is used for convective terms and diffusion is neglected. Hybrid differencing has the favorable properties of upwind and central differencing schemes. It inherently satisfies the Scarborough criteria as upwind differencing. It also takes the flow direction into account and provides first order accuracy.

Power-law differencing scheme is similar to hybrid differencing in the sense that it contains a switch based on the face Peclet numbers [6]. In this scheme diffusion through a face is set to zero when the face Peclet number exceeds 10. For lower values of Peclet numbers diffusive term is calculated using a fifth order polynomial that is again derived using the exact solution of the one-dimensional convection-diffusion problem. Convective flux calculations use upwind differencing.

Upto here the diffusive and convective terms of the u-momentum equation have been discretized. Remaining source terms of the momentum equation will be covered in the next section.

2.3.1.6 Discretization of the Pressure Gradient and the Source Term

Since the pressure gradient term is the main source term in most flows of engineering importance it is excluded from the rest of the source terms. Consider the discretization of the pressure term of Equation (2.8) for the control volume corresponding to cell P of Figure 2.1. Applying the gradient theorem the volume integral including the pressure gradient can be converted to the following surface integral

$$\int_{CV} -\vec{i} \cdot \nabla p d\forall = \int_{A} -\vec{i} \cdot p d\boldsymbol{A} \tag{2.24}$$

This integral should be evaluated over the entire closed surface of cell P. Assuming that the face centroid value of pressure represents the mean value of the face, the surface integral can be written as

$$\int_{A} -\vec{i} \cdot p d\boldsymbol{A} = \sum_{f=e,w,n,s} -\vec{i} \cdot p_f \boldsymbol{A_f} \tag{2.25}$$

or using the definition of face area vectors from Equation (2.5)

$$\int_{A} -\vec{i} \cdot p d\boldsymbol{A} = (p_w - p_e)\Delta y \tag{2.26}$$

Finally the source term of the u-momentum equation can be discretized by assuming that an average value of the source, $\overline{S}$, prevails inside the control volume

$$\int_{CV} S_u d\forall = \overline{S}\Delta\forall \tag{2.27}$$

where $\Delta\forall = \Delta x \Delta y$ is the area of the cell for which the source is being evaluated. Often the source term is a function of the dependent variable u itself, and it is desirable to consider this dependence in constructing the discrete equation. Constant and u-dependent parts of the source term can be seperated as

$$\overline{S}\Delta\forall = (S_c + S_P u_P)\Delta x \Delta y \tag{2.28}$$

At this point all the terms of Equation (2.8) are discretized. Now it is possible to combine these individual terms to write the discrete (algebraic) form of the u-momentum equation. Note that in the previous sections four different schemes were described for the discretization of diffusive

and convective terms. Therefore four different discrete momentum equations will be mentioned next.

If central differencing is used for both the diffusive and convective fluxes then the discrete u-momentum equation can be written by substituting Equations (2.13), (2.18), (2.26) and (2.28) into Equation (2.8) to get

$$F_e A_e \frac{u_E + u_P}{2} - F_w A_w \frac{u_W + u_P}{2} + F_n A_n \frac{u_N + u_P}{2} - F_s A_s \frac{u_S + u_P}{2} =$$
$$D_e A_e (u_E - u_P) - D_w A_w (u_P - u_W) + D_n A_n (u_N - u_P) - D_s A_s (u_P - u_S)$$
$$+ (p_w - p_e)\Delta y + (S_c + S_P u_P)\Delta x \Delta y \qquad (2.29)$$

which can be put into the following compact form

$$\begin{aligned} a_P u_P &= \sum_{nb} a_{nb} u_{nb} \\ &= a_E u_E + a_W u_W + a_N u_N + a_S u_S + (p_w - p_e)\Delta y + S_c \Delta x \Delta y \end{aligned} \qquad (2.30)$$

where the subscript nb stands for the four neigbouring nodes of the node P. The coefficients of Equation (2.30) are given as

$$\begin{aligned} a_E &= \Big(D_e - \frac{F_e}{2}\Big) A_e \\ a_W &= \Big(D_w + \frac{F_w}{2}\Big) A_w \\ a_N &= \Big(D_n - \frac{F_n}{2}\Big) A_n \\ a_S &= \Big(D_s + \frac{F_s}{2}\Big) A_s \\ a_P &= a_E + a_W + a_N + a_S + (F_e A_e - F_w A_w + F_n A_n - F_s A_s) - S_P \Delta x \Delta y \end{aligned} \qquad (2.31)$$

If central and upwind differencing are used for the discretization of diffusive and convective terms, respectively, Equations (2.13), (2.21), (2.26) and (2.28) can be combined into Equation

(2.8) to get a discrete u-momentum equation of the form (2.30). New coefficients will be

$$\begin{aligned}
a_E &= \Big(D_e + max\left[-F_e, 0\right]\Big)A_e \\
a_W &= \Big(D_w + max\left[F_w, 0\right]\Big)A_w \\
a_N &= \Big(D_n + max\left[-F_n, 0\right]\Big)A_n \\
a_S &= \Big(D_s + max\left[F_s, 0\right]\Big)A_s \\
a_P &= a_E + a_W + a_N + a_S + (F_eA_e - F_wA_w + F_nA_n - F_sA_s) - S_P\Delta x\Delta y \qquad (2.32)
\end{aligned}$$

Hybrid differencing will again result in a similar discrete u-momentum equation with the following coefficients

$$\begin{aligned}
a_E &= max\left[-F_e, \Big(D_e - \frac{F_e}{2}\Big), 0\right] \\
a_W &= max\left[F_w, \Big(D_w + \frac{F_w}{2}\Big), 0\right] \\
a_N &= max\left[-F_n, \Big(D_n - \frac{F_n}{2}\Big), 0\right] \\
a_S &= max\left[F_s, \Big(D_s + \frac{F_s}{2}\Big), 0\right] \\
a_P &= a_E + a_W + a_N + a_S + (F_eA_e - F_wA_w + F_nA_n - F_sA_s) - S_P\Delta x\Delta y \qquad (2.33)
\end{aligned}$$

Finally the coefficients of Equation (2.30) for power-law differencing scheme are given as

$$\begin{aligned}
a_E &= D_e max\left[0, \Big(1 - \frac{0.1\left|F_e\right|}{D_e}\Big)^5\right] + max\left[0, -F_e\right] \\
a_W &= D_w max\left[0, \Big(1 - \frac{0.1\left|F_w\right|}{D_w}\Big)^5\right] + max\left[0, F_w\right] \\
a_N &= D_n max\left[0, \Big(1 - \frac{0.1\left|F_n\right|}{D_n}\Big)^5\right] + max\left[0, -F_n\right] \\
a_S &= D_s max\left[0, \Big(1 - \frac{0.1\left|F_s\right|}{D_s}\Big)^5\right] + max\left[0, -F_s\right] \\
a_P &= a_E + a_W + a_N + a_S + (F_eA_e - F_wA_w + F_nA_n - F_sA_s) - S_P\Delta x\Delta y \qquad (2.34)
\end{aligned}$$

This concludes the discretization of the u-momentum equation. As stated before discretization of v-momentum equation follows similar steps.

2.3.2 Discretization of the Continuity Equation

For time-independent flows the continuity equation takes the following form

$$\nabla \cdot (\rho \vec{V}) = 0 \tag{2.35}$$

Integrating over the control volume and applying the Gauss' divergence theorem gives

$$\int_{CV} \nabla \cdot (\rho \vec{V}) d\forall = \int_{A} \rho \vec{V} \cdot d\boldsymbol{A} \tag{2.36}$$

As it is assumed for the discretization of the momentum equation, $\vec{V}$ on the face can be represented by its value at the face centroid to get

$$\int_{A} \rho \vec{V} \cdot d\boldsymbol{A} = \sum_{f=e,w,n,s} (\rho \vec{V})_f \cdot \boldsymbol{A_f} \tag{2.37}$$

which, for a Cartesian mesh, can be simplified to

$$(\rho u A)_e - (\rho u A)_w + (\rho v A)_n - (\rho v A)_s = 0 \tag{2.38}$$

This very straightforward discretization of the continuity equation unfortunately hides a major difficulty faced within the numerical solution of incompressible flows, i.e. not having the pressure as an unknown in it. The SIMPLE algorithm that will be discussed in the coming sections provides a way to convert the continuity equation into a form that can be used to solve for pressure. But before giving the details of it, it is necessary to understand the two major grid arrangement possibilities; colocated and staggered grids.

2.4 Grid Arrangement

Assume for the moment that the velocity vectors ($\vec{V}$) and the pressures (p) are stored at the cell centroids as shown in Figure 2.2, such a grid arrangement is called as collocated. Then an interpolation have to be made to find the pressure values at the cell faces for discrete momentum

equation, Equation (2.30).

$$\begin{aligned} p_e &= \frac{p_E + p_P}{2} \\ p_w &= \frac{p_W + p_P}{2} \\ p_n &= \frac{p_N + p_P}{2} \\ p_s &= \frac{p_S + p_P}{2} \end{aligned} \tag{2.39}$$

Therefore the pressure gradient term in the u-momentum equation becomes,

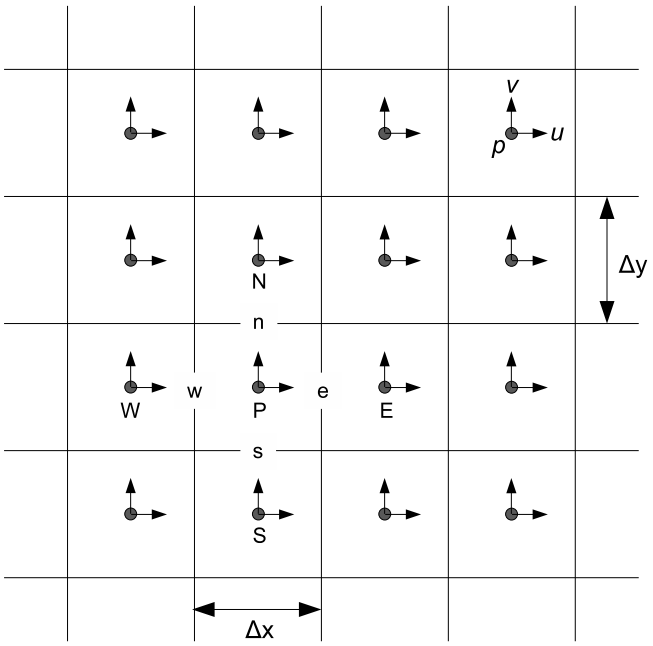

Figure 2.2: Two Dimensional Collocated Grid Arrangement

$$(p_w - p_e)\Delta y \quad \Rightarrow \quad (p_W - p_E)\frac{\Delta y}{2} \tag{2.40}$$

Similarly for v-momentum equation

$$(p_s - p_n)\Delta x \quad \Rightarrow \quad (p_S - p_N)\frac{\Delta x}{2} \tag{2.41}$$

It can easily be noticed that the pressure gradient term in u- and v-momentum equations are $2\Delta x$ apart on the mesh, as shown in Figure 2.2, and do not involve the pressure at the point P.

Similarly, for continuity equation (2.38) the velocity values at the faces are not available directly

and they must be interpolated from the cell centroid values.

$$\begin{aligned} u_e &= \frac{u_E + u_P}{2} \\ u_w &= \frac{u_W + u_P}{2} \\ v_n &= \frac{v_N + v_P}{2} \\ v_s &= \frac{v_S + v_P}{2} \end{aligned} \tag{2.42}$$

Gathering terms together and rewriting the continuity equation (2.38),

$$(\rho u)_E \Delta y - (\rho u)_W \Delta y + (\rho v)_N \Delta x - (\rho v)_S \Delta x = 0 \tag{2.43}$$

Now, assume a chekerboarded velocity pattern, as shown in Figure 2.3, is somehow obtained during the calculations. Although this velocity field is completely unphysical, the continuity equation will surprisingly sustain it, because the continuity equation (2.43) for cell P does not contain the velocity of this cell, which hinders the continuity equation to realize this high gradient in the velocity. During the solution a pressure field whose gradients exactly compensate the checkerboarding of momentum transport implied by the checkerboarded velocity field will also be developed. Consequently, the converged solution would exhibit checkerboarding. It can be thought other way around, this time assume that a checkerboarded pressure field exists. Then momentum equations will not be able to distinguish it from a completely uniform pressure field, since the pressure terms in the momentum equations are $2\Delta x$ or $2\Delta y$ apart depending on the equation.

In practical applications perfect checkerboarding is rarely encountered because of mainly irregular mesh, boundary conditions and physical properties. However, unphysical wiggles in the velocity and pressure fields which can be seen as a tendency toward checkerboarding,can often be encountered.

2.4.1 Staggered Grid

A widely accepted remedy for checkerboarding is to use staggered mesh [7]. A typical staggered mesh arrangement is shown in Figure 2.4. Pressure and all the scalar other variables (e.g.

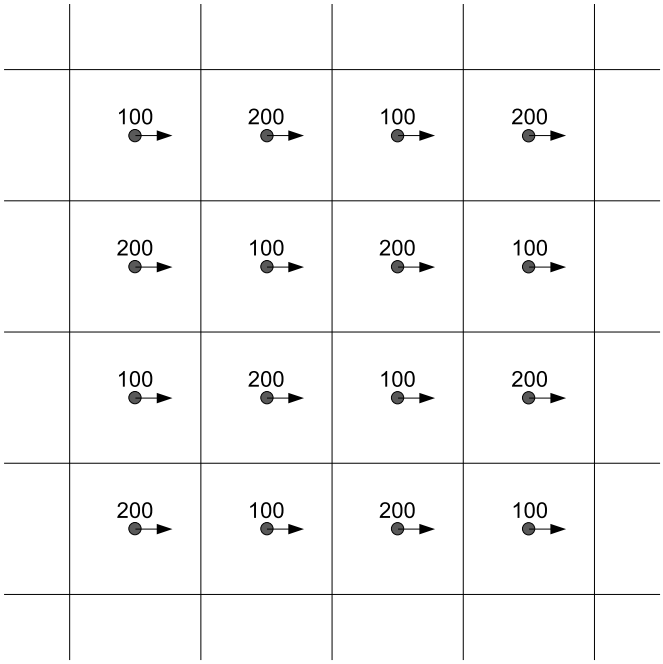

Figure 2.3: Checkerboard Velocity Field

pressure, density, enthalpy) are stored at the centroids of the main cells (•). The velocity components are stored on the faces of main cells, which are also the centroids of the corresponding staggered cells. Horizontal arrows (→) indicate the location where the u-velocities are stored and vertical arrows (↑) for v-velocities.

The staggered grid has many advantages over collocated grid. First, the discrete continuity equation (2.38) can be directly used without interpolation since velocities are available at the points where they are required. However, in collocated arrangement, further interpolation has to be made, equation (2.42), to carry the velocities to the cell faces which is the soul reason of velocity checkerboarding. Consequently, staggered grid arrangement not only decreases the computational load but also eliminates the possibility of velocity checkerboarding. Similarly for momentum equation, when staggered grid arrangement is used, the pressure gradient terms can be written directly using the pressures on the faces of momentum control volumes. As a result, no further interpolation is required as in collocated grid, equation (2.39), and the pressure values used in the momentum equations are no longer $2\Delta x$ apart, means the pressure checkerboarding is also eliminated. Then the pressure gradient terms in the momentum equations(2.30) turn

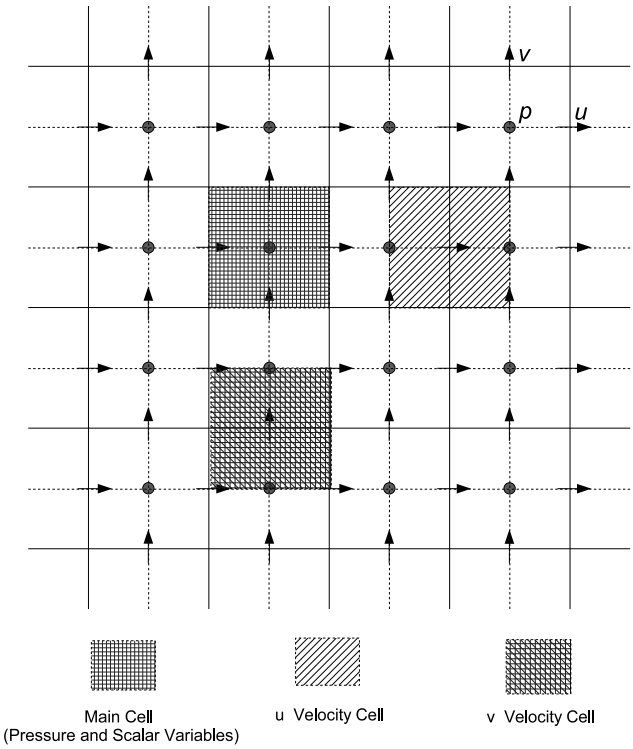

Figure 2.4: Staggered Grid Arrangement

out to be,

For u-momentum equation

$$(p_P - p_E)\Delta y$$
$$(p_W - p_P)\Delta y \tag{2.44}$$

For v-momentum equation

$$(p_P - p_N)\Delta x$$
$$(p_S - p_P)\Delta x \tag{2.45}$$

It can be noticed that u and v control volumes overlap each other and the main control volumes, but this is of no consequence other than complex indexing which cause difficulty in coding. This problem arises especially in complex geometries, however there exist studies [48] in the literature which successfully use staggered arrangement in these geometries.

2.5 Solution Methods

In the previous sections discretization process of the continuity and momentum equations are examined. Next step is to study the available methods that can be used to solve the discretized equations. Solution method not only affects the accuracy of the solution but also the computation time and above all, it determines whether a solution can be obtained or not. Sequential solution methods are used in this study. As explained in Section 1.5, sequential solution methods solve the continuity and two momentum equations one-by-one. In sequential solvers each discrete equation has to be associated with a particular unknown. The discrete u-momentum equation is solved to get the u-velocities and similarly v-momentum equation is used for the v-velocities. However, the third unknown, pressure, can not be obtained using the remaining continuity equation. As it is known, pressure does not appear in the continuity equation. For incompressible flows, the constant density that appears in the continuity equation can not be linked to pressure. To use sequential solvers it is therefore necessary to find a way to introduce pressure into the continuity equation. This is the basic idea of the SIMPLE algorithm, as it will be explained in the next section.

2.5.1 SIMPLE Algorithm

The SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm is essentially a guess and correct method [13]. The primary idea behind SIMPLE is to convert the discrete continuity equation into a new equation called as the pressure correction equation that can be used to solve for pressure. To create this new equation, SIMPLE method forms a relation between the velocity terms of the continuity equation and pressure gradient terms of the discrete momentum equations.

To have a general understanding, SIMPLE algorithm works as follows; It starts with initial guesses of pressure and velocity fields. u- and v-momentum equations are solved using these guessed values. Of course there is no guarantee that the resulting velocity field will be divergence free to satisfy the continuity equation. Next, the pressure correction equation is solved

to obtain pressure corrections that can be used to correct the pressure and the velocity fields. The velocity field will now be divergence free, but the corrected variables will not satisfy the momentum equations. The iteration between the continuity and momentum equations continue till pressure and velocity fields that satisfy all three equations are obtained.

The SIMPLE algorithm will be explained using the sample computational domain given in Figure 2.5. In this staggered grid, (i,J) nodes store the u-velocity (u), (I,j) nodes store the v-velocity (v) and (I,J) nodes store the pressure (p). Using Equation (2.30) and the indexing shown in Figure 2.5 discrete u- and v-momentum equations can be rewritten as follows

$$\begin{aligned} a_{i,J}u_{i,J} &= \sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + b_{i,J} \qquad \text{u-momentum} \\ a_{I,j}v_{I,j} &= \sum a_{nb}v_{nb} + (p_{I,J} - p_{I,J+1})A_{I,j} + b_{I,j} \qquad \text{v-momentum} \end{aligned} \tag{2.46}$$

where the final b terms represent the constant part of the source terms, $S_C\Delta x\Delta y$, and

$$\begin{aligned} A_{i,J} &= y_{i,j} - y_{i,j-1} \\ A_{I,j} &= x_{i,j} - x_{i-1,j} \end{aligned} \tag{2.47}$$

The SIMPLE algorithm starts with initial guesses of the pressure and velocity fields, p^*, u^*, v^*. Discrete momentum equations are solved using these guessed fields as follows

$$\begin{aligned} a_{i,J}u^*_{i,J} &= \sum a_{nb}u^*_{nb} + (p^*_{I,J} - p^*_{I+1,J})A_{i,J} + b_{i,J} \qquad \text{u-momentum} \\ a_{I,j}v^*_{I,j} &= \sum a_{nb}v^*_{nb} + (p^*_{I,J} - p^*_{I,J+1})A_{I,j} + b_{I,j} \qquad \text{v-momentum} \end{aligned} \tag{2.48}$$

Note that the initial u^* and v^* guesses are necessary to calculate the convective fluxes, F. Since the pressure field, p^*, is only a guess or a prevailing iterate, the velocity field, u^* and v^*, obtained by solving Equation (2.48) will not be divergence free and therefore will not satisfy the continuity equation (Equation (2.38)). Let's assume that we have the following corrections (shown with primes) that provide a divergence free velocity field

$$\begin{aligned} u &= u^* + u' \\ v &= v^* + v' \end{aligned} \tag{2.49}$$

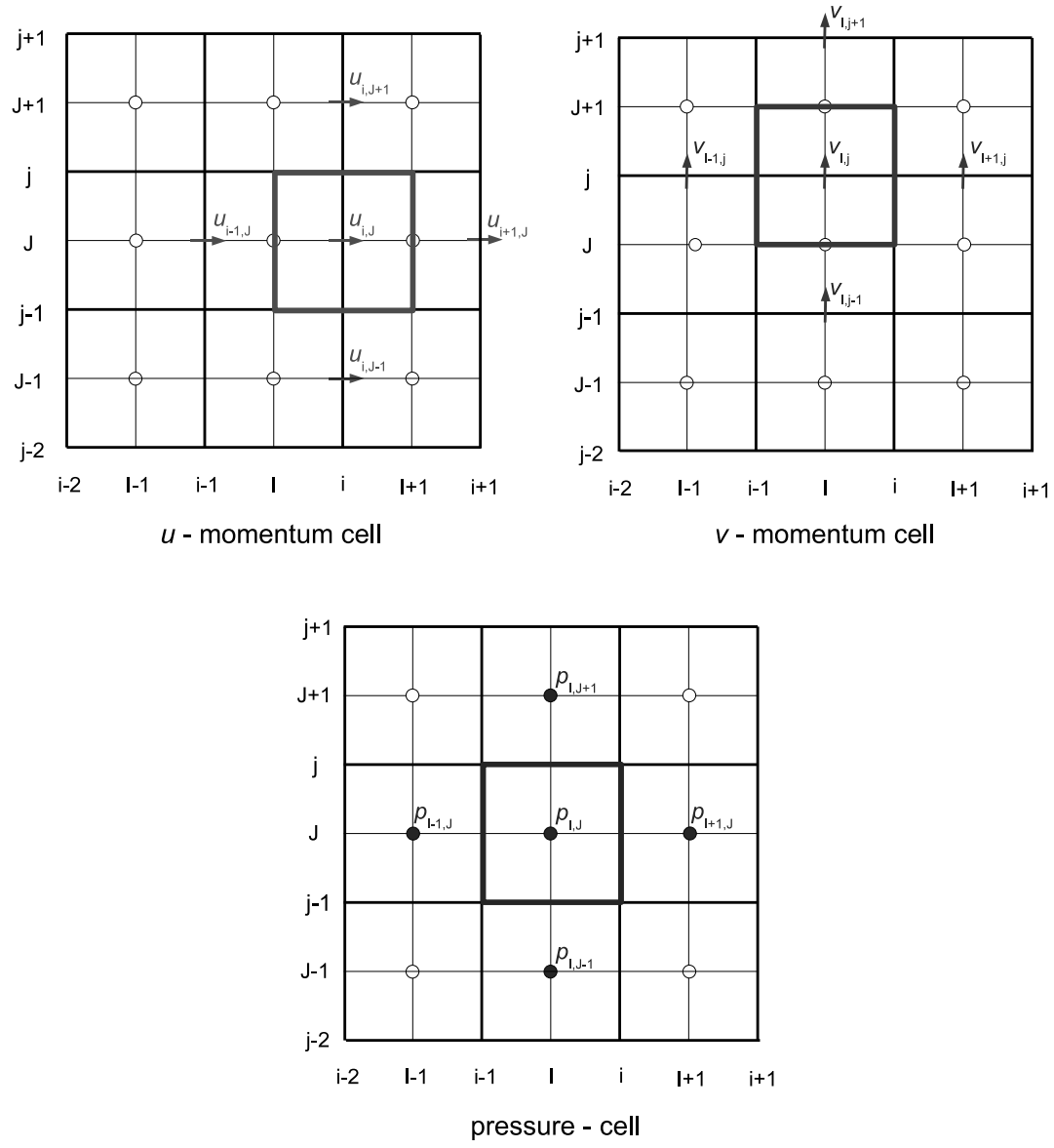

Figure 2.5: Staggered Grid Arrangement

Correspondingly, the pressure field can be corrected to satisfy the momentum equations as

$$p = p^* + p' \tag{2.50}$$

Subtracting Equation (2.48) from Equation (2.46) and using the previously defined corrections yields

$$\begin{aligned} a_{i,J}u'_{i,J} &= \sum a_{nb}u'_{nb} + (p'_{I,J} - p'_{I+1,J})A_{i,J} \\ a_{I,j}v'_{I,j} &= \sum a_{nb}v'_{nb} + (p'_{I,J} - p'_{I,J+1})A_{I,j} \end{aligned} \tag{2.51}$$

As the main approximation of the SIMPLE algorithm, $\sum a_{nb}u_{nb}^{'}$ and $\sum a_{nb}v_{nb}^{'}$ terms of Equation (2.51) are dropped to get (a justification for this simplification will be discussed later)

$$\begin{aligned} u_{i,J}^{'} &= d_{i,J}(p_{I,J}^{'} - p_{I+1,J}^{'}) \\ v_{I,j}^{'} &= d_{I,j}(p_{I,J}^{'} - p_{I,J+1}^{'}) \end{aligned} \tag{2.52}$$

where

$$\begin{aligned} d_{i,J} &= \frac{A_{i,J}}{a_{i,J}} \\ d_{I,j} &= \frac{A_{I,j}}{a_{I,j}} \end{aligned} \tag{2.53}$$

Corrected velocities can now be expressed as

$$\begin{aligned} u_{i,J} &= u_{i,J}^{*} + d_{i,J}(p_{I,J}^{'} - p_{I+1,J}^{'}) \\ v_{I,j} &= v_{I,j}^{*} + d_{I,j}(p_{I,J}^{'} - p_{I,J+1}^{'}) \end{aligned} \tag{2.54}$$

Now it is time to obtain the necessary pressure corrections.

2.5.1.1 Pressure Correction Equation

Before studying the derivation of the pressure correction equation, it is necessary to rewrite the continuity equation (Equation (2.38)) with the indexing given in Figure 2.5.

$$(\rho uA)_{i,J} - (\rho uA)_{i-1,J} + (\rho vA)_{I,j} - (\rho vA)_{I,j-1} = 0 \tag{2.55}$$

As stated before, the starred velocities u^* and v^* obtained by solving Equation (2.48) do not satisfy the continuity equation, so that

$$(\rho u^{*}A)_{i,J} - (\rho u^{*}A)_{i-1,J} + (\rho v^{*}A)_{I,j} - (\rho v^{*}A)_{I,j-1} \neq 0 \tag{2.56}$$

Using Equation (2.54) in Equation (2.55) provides

$$\begin{aligned} &\rho\Big[A_{i,J}\Big(u_{i,J}^{*} + d_{i,J}(p_{I,J}^{'} - p_{I+1,J}^{'})\Big) - A_{i-1,J}\Big(u_{i-1,J}^{*} + d_{i-1,J}(p_{I-1,J}^{'} - p_{I,J}^{'})\Big) + \\ &A_{I,j}\Big(v_{I,j}^{*} + d_{I,j}(p_{I,J}^{'} - p_{I,J+1}^{'})\Big) - A_{I,j-1}\Big(v_{I,j-1}^{*} + d_{I,j-1}(p_{I,J-1}^{'} - p_{I,J}^{'})\Big)\Big] = 0 \end{aligned} \tag{2.57}$$

which can be rearranged to get the following discrete pressure correction equation

$$a_{I,J}p_{I,J}^{'} = a_{I+1,J}p_{I+1,J}^{'} + a_{I-1,J}p_{I-1,J}^{'} + a_{I,J+1}p_{I,J+1}^{'} + a_{I,J-1}p_{I,J-1}^{'} + b_{I,J}^{'} \tag{2.58}$$

where

$$
\begin{aligned}
a_{I+1,J} &= \rho d_{i,J} A_{i,J} \\
a_{I-1,J} &= \rho d_{i-1,J} A_{i-1,J} \\
a_{I,J+1} &= \rho d_{I,j} A_{I,j} \\
a_{I,J-1} &= \rho d_{I,j-1} A_{I,j-1} \\
a_{I,J} &= a_{I+1,J} + a_{I-1,J} + a_{I,J+1} + a_{I,J-1} \\
b'_{I,J} &= -\rho u^*_{i,J} A_{i,J} + \rho u^*_{i-1,J} A_{i-1,J} - \rho v^*_{I,j} A_{I,j} + \rho v^*_{I,j-1} A_{I,j-1}
\end{aligned} \tag{2.59}
$$

It is important to note that the term $b'_{I,J}$ of Equation (2.59) is the negative of Equation (2.56). This term is called as the mass source and it can be thought as a measure of mass imbalance in the continuity equation due to the use of wrong (starred) velocities. When the prevailing iterate velocities, u^* and v^* exactly satisfy the continuity equation, $b'_{I,J}$ term will be zero and the pressure correction equation will yield a constant value, which, due to Equation (2.52), results in no velocity correction. Therefore the convergence is obtained once the velocities predicted by the momentum equations satify the continuity equation.

At this point it is important to realize the reason for dropping the $\sum a_{nb} u'_{nb}$ and $\sum a_{nb} v'_{nb}$ terms from Equation (2.51). Assume for a moment that these terms are retained. Then they would have to be expressed in terms of the pressure and velocity corrections at the neighbors of corresponding u- and v-cells. These neighbors would, in turn, bring their neighbors. Consequently, the resulting pressure correction equation would involve all the grid points in the computational domain, which makes the solution of the pressure correction equation very expensive.

Now a new question arises whether the converged solution given by SIMPLE does contain any error due to the omission explained in the previous paragraph. If we were solving a pressure equation rather than a pressure correction equation then the omission of any term would be unacceptable, since the result would not be the true solution of the governing discrete equa-

tions. However, we are working with a pressure correction equation and as far as the final answers are considered the omission of $\sum a_{nb}u_{nb}^{'}$ and $\sum a_{nb}v_{nb}^{'}$ terms is of no consequence. This is easily seen if what happens in the final iteration is considered. In the final iteration before convergence, the velocity and pressure fields of the previous iteration can be denoted as u^*, v^*, p^*. Since this velocity field corresponds to a converged solution the mass source, $b^{'}$, turns out to be zero. This means pressure correction equation will yield a constant value that can be taken as zero. If $p^{'} = 0$ at all grid points then $u = u^*$, $v = v^*$ and $p = p^*$ thus the pressure correction equation plays no role in the final iteration so that the converged solution is uninfluenced by any approximations made in deriving this equation.

Although dropping the $\sum a_{nb}u_{nb}^{'}$ and $\sum a_{nb}v_{nb}^{'}$ terms does not affect the converged results, it has consequences for the rate of convergence. By dropping these terms, the entire burden of velocity corrections is placed upon the pressure correction, Equation (2.52). As a result, corrected velocity field will satisfy the continuity equation but the resulting pressure field is over-corrected which is prone to divergence unless some under relaxation is used

$$p = p^* + \alpha_p p^{'} \tag{2.60}$$

where α_p is the pressure under-relaxation factor that has a value between 0 and 1. As the under-relaxation factor gets closer to 1, convergence speed increases, but the chance of divergence also increases. On the other hand, as the value gets closer to 0, the solution becomes more stable, but now the convergence slows down. There exist no general guidelines for the determination of an optimum under-relaxation factor, hence each problem should be considered seperately [6].

Most of the time a stable solution would also require under-relaxation for the momentum equations. Using Equation (2.46), the unknown u-velocity at node P can be written as

$$u_{i,J} = \frac{\sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + b_{i,J}}{a_{i,J}} \tag{2.61}$$

The Change between this u-velocity of the current itearitон and the one for the previous iteration becomes

$$\frac{\sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + b_{i,J}}{a_{i,J}} - u^*_{i,J} \tag{2.62}$$

This difference can be under-relaxed to update the current value of the u-velocity at node P as

$$u_{i,J} = u^*_{i,J} + \alpha_u \left[\frac{\sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + b_{i,J}}{a_{i,J}} - u^*_{i,J} \right] \tag{2.63}$$

which can be rearranged to give

$$\frac{a_{i,J}}{\alpha_u}u_{i,J} = \sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + b_{i,J} + \left[(1-\alpha_u)\frac{a_{i,J}}{\alpha_u} \right] u^*_{i,J} \tag{2.64}$$

where α_u is the under-relaxation parameter of the u-momentum equation. Similarly the under-relaxation of the v-momentum equation provides

$$\frac{a_{I,j}}{\alpha_v}v_{I,j} = \sum a_{nb}v_{nb} + (p_{I,J} - p_{I,J+1})A_{I,j} + b_{I,j} + \left[(1-\alpha_v)\frac{a_{I,j}}{\alpha_v} \right] v^*_{I,j} \tag{2.65}$$

As mentioned before, the choice of the under-relaxation factors is critical for an efficient analysis. The effect of these parameters on the convergence characteristics will be investigated in the next chapter.

2.5.1.2 Sequence of Operations of the SIMPLE Algorithm

The overall procedure that the SIMPLE algorithm follows is given below:

1. Initialize the pressure and velocity fields (p^*,u^*,v^*).
2. Solve the discrete momentum Equations (2.64) and (2.65) to obtain u^* and v^*.
3. Solve the pressure correction equation (2.58) to obtain $p^{'}$.
4. Correct the pressure and velocity fields using the pressure (2.50) and velocity (2.54) correction equations.
5. If necessary, solve the discrete equations for other scalar variables such as temperature, species concentration or turbulence quantities, using the continuity-satisfying velocity field.

6. If the solution is converged based on a predetermined convergence tolerance, stop. If not, go to the second step.

2.5.2 SIMPLER Algorithm

There have been a number of studies in the literature which aim to accelerate the convergence of the SIMPLE algorithm. One of such attempts is the SIMPLER (SIMPLE Revised) algorithm of Patankar [6]. The approximation introduced in the SIMPLE algorithm for the derivation of pressure correction equation leads an exaggerated pressure-correction field which requires under-relaxation, thus slows down the convergence rate. In other words, pressure correction equation is an efficient method when the velocity correction part is considered; however it is not that much successful in correcting the pressure.

To understand this argument, a very simple problem can be considered; one dimensional, constant density flow with a given inlet velocity boundary condition. In this problem, velocity is governed only by the continuity equation, and hence the final solution, the continuity satisfying velocity field, is obtained at the end of the first iteration. However, converged pressure field can be established after many iterations[6]. Consequently, while finding another way to obtain the pressure field, it would be appropriate to employ pressure correction equation only to correct the velocity field.

The pressure equation is obtained by re-arranging the momentum equations. First the momentum equations are written as,

$$\begin{aligned} u_{i,J} &= \frac{\sum a_{nb}u_{nb} + b_{i,J}}{a_{i,J}} + d_{i,J}(p_{I,J} - p_{I+1,J}) \\ v_{I,j} &= \frac{\sum a_{nb}v_{nb} + b_{I,j}}{a_{I,j}} + d_{I,j}(p_{I,J} - p_{I,J+1}) \end{aligned} \tag{2.66}$$

By defining pseudovelocities ($\widehat{u}$ and $\widehat{v}$) as,

$$\widehat{u}_{i,J} = \frac{\sum a_{nb} u_{nb} + b_{i,J}}{a_{i,J}}$$
$$\widehat{v}_{I,j} = \frac{\sum a_{nb} v_{nb} + b_{I,j}}{a_{I,j}} \tag{2.67}$$

The momentum equations become,

$$u_{i,J} = \widehat{u}_{i,J} + d_{i,J}(p_{I,J} - p_{I+1,J})$$
$$v_{I,j} = \widehat{v}_{I,j} + d_{I,j}(p_{I,J} - p_{I,J+1}) \tag{2.68}$$

It is easy to see the similarity between the Equations (2.68) and (2.54). Here the pseudovelocities ($\widehat{u}$ and $\widehat{v}$) appear in the place of prevailing iterate velocity field (u^* and v^*) and the pressure itself (p) takes the place of pressure correction ($p^{'}$). Then the same procedure, as in the SIMPLE algorithm, is followed to derive an equation for pressure.

Substituting Equation (2.68) into the discrete continuity equation (2.55), the following equation for pressure is obtained.

$$a_{I,J} p_{I,J} = \sum_{nb} a_{nb} p_{nb} + b_{I,J} \tag{2.69}$$

where

$$\begin{aligned}
a_{I+1,J} &= \rho d_{i,J} A_{i,J} \\
a_{I-1,J} &= \rho d_{i-1,J} A_{i-1,J} \\
a_{I,J+1} &= \rho d_{I,j} A_{I,j} \\
a_{I,J-1} &= \rho d_{I,j-1} A_{I,j-1} \\
a_{I,J} &= a_{I+1,J} + a_{I-1,J} + a_{I,J+1} + a_{I,J-1} \\
b_{I,J} &= -\rho \widehat{u}_{i,J} A_{i,J} + \rho \widehat{u}_{i-1,J} A_{i-1,J} - \rho \widehat{v}_{I,j} A_{I,j} + \rho \widehat{v}_{I,j-1} A_{I,j-1}
\end{aligned} \tag{2.70}$$

It can be noticed that the pressure equation is identical to the pressure correction equation (2.59) except the $b_{I,J}$ term. At this point it should be noted that; different than the pressure correction equation where the source term represent the mass imbalance, the source term of the pressure equation does not represent the mass source. Another important difference between

these equations is that the pressure equation involves no approximation. Thus, if a correct velocity field is used, the pressure equation would give the correct pressure at once.

2.5.2.1 Sequence of Operations of the SIMPLER Algorithm

The revised algorithm SIMPLER solution loop takes the following form:

1. Initialize the velocity field.
2. Compute the pseudovelocities ($\widehat{u}$ and $\widehat{v}$).
3. Solve the pressure Equation (2.69) which yields the pressure field.
4. Using this new calculated pressure field, and the momentum Equations (2.64), 2.65), calculate the velocity fields (u^*, v^*).
5. Solve the pressure correction Equation (2.58) hence obtain the $p^{'}$ field.
6. Correct the velocity fields using the velocity (2.54) correction equations. Do NOT correct the pressure.
7. Solve the discrete equations for other scalar variables (ϕ) such as temperature, concentration and turbulence quantities, using the continuity-satisfying velocity field for the convective terms.
8. If the solution is converged, stop. If not, go to the second step.

2.5.3 SIMPLEC Algorithm

The SIMPLEC (SIMPLE-Corrected) algorithm of Van Doormal [14] was developed as a remedy to the relatively crude approximation of the SIMPLE algorithm which is the omission of $\sum a_{nb}u_{nb}^{'}$ and $\sum a_{nb}v_{nb}^{'}$ terms in the derivation of pressure correction equation. The SIMPLEC algorithm attempts to approximate the omitted terms rather than completely neglecting them as,

$$\begin{aligned} \sum_{nb} a_{nb}u_{n}^{'}b &\approx u_{i,J}^{'}\sum_{nb} a_{nb} \\ \sum_{nb} a_{nb}v_{n}^{'}b &\approx v_{I,j}^{'}\sum_{nb} a_{nb} \end{aligned} \tag{2.71}$$

Thus, the velocity correction takes the form

$$u_{i,J}^{'} = d_{i,J}(p_{I,J}^{'} - p_{I+1,J}^{'})$$
$$v_{I,j}^{'} = d_{I,j}(p_{I,J}^{'} - p_{I,J+1}^{'}) \tag{2.72}$$

where

$$d_{i,J} = \frac{A_{i,J}}{a_{i,J} - \sum_{nb} a_{nb}}$$
$$d_{I,j} = \frac{A_{I,j}}{a_{I,j} - \sum_{nb} a_{nb}} \tag{2.73}$$

The rest of the procedure is same as the SIMPLE algorithm except for the fact that the pressure correction equation does not need under relaxation anymore. Whereas, the momentum equations have to be under-relaxed to prevent the denominator of Equation (2.73) to be zero. Since, if the momentum equations are not under-relaxed, then the $a_{i,J}$ term will be equal to $\sum_{nb} a_{nb}$ term.

Although this algorithm converges faster than the SIMPLE algorithm and does not have the extra computational cost as in the SIMPLER algorithm, it shares the same disadvantage with the SIMPLE algorithm. This method would destroy the initially good guessed velocity field if it is not accompanied by a good pressure field guess, since there exist no equation dedicated only to solve the pressure field as in the SIMPLER algorithm.

A comparative study between these algorithms on different test cases and the superiority of each algorithm over the others are explained in the next chapter.

2.5.3.1 Sequence of Operations of the SIMPLEC Algorithm

The sequence of operations of the SIMPLEC algorithm is exactly same as the SIMPLE algorithm; the only difference is at the fourth step where different velocity correction Equation (2.72) is used.

2.6 Unsteady Flows

The solution procedure of unsteady flows is very similar to the one followed for steady problems. As usual, the process starts with the discretization of the governing equations. However, for the unsteady flow calculations the transient terms should also be discretized. Therefore, Equations (2.3) and (2.4) are integrated over not only a control volume but also a finite time step.

2.6.1 Discretization of the Equation

Replacing the volume integrals of diffusive and convective terms with the surface integrals, using the Gauss Divergence Theorem, and changing the order of integration for the transient term, the unsteady u-momentum Equation (2.3) takes the following form.

$$\int_{CV}\Big(\int_t^{t+\Delta t}\frac{\partial(\rho u)}{\partial t}dt\Big)dV+\int_t^{t+\Delta t}\Big(\int_A(\rho\vec{V}u)\cdot d\boldsymbol{A}\Big)dt=$$

$$\int_t^{t+\Delta t}\Big(\int_{CV}-\vec{i}\cdot\nabla p dV\Big)dt+\int_t^{t+\Delta t}\Big(\int_A(\mu\nabla u)\cdot d\boldsymbol{A}\Big)dt+\int_t^{t+\Delta t}\Big(\int_{CV}S_u dV\Big)dt \quad (2.74)$$

There are several different schemes available to discretize the time derivative. First-order, backward differencing scheme, used in this study, provides the following discretized form,

$$\int_{CV}\Big(\int_t^{t+\Delta t}\frac{\partial(\rho u)}{\partial t}dt\Big)dV=\rho\frac{(u-u^o)}{\Delta t}\Delta V \quad (2.75)$$

where, the superscript 'o' refers to the u-velocity at time t. No superscript is used for the u-velocity at time level $t+\Delta t$.

The remaining terms of the u-momentum equation can only be calculated if an assumption is made to approximate the variation of u-velocity with time. It is possible to use u-velocities at time step t only, u-velocities at time step $t+\Delta t$ only or a combination of both can be used. It is possible to generalize the approach using a weighting parameter (θ) that takes a value between 0 and 1. Then,

$$\int_t^{t+\Delta t}u dt=\left[\theta u+(1-\theta)u^o\right]\Delta t \quad (2.76)$$

Hence, the final form of the discrete equation depends on the value of θ. $\theta = 0$ is called as explicit discretization where only the u-velocities that belong the previous time level t are used to evaluate the velocities at the new time level $t + \Delta t$.

$$\int_t^{t+\Delta t} u dt = u^o \Delta t \tag{2.77}$$

$0 < \theta < 1$ provides implicit discretization where velocities of both previous (u^o) and new (u) time levels are used in the calculation. The special case of $\theta = 1/2$ is called as the Cranck-Nicolson scheme where both time levels have equal dominance in the calculations [49].

$$\int_t^{t+\Delta t} u dt = \frac{1}{2}(u^o + u)\Delta t \tag{2.78}$$

The extreme case of $\theta = 1$ is called as the fully implicit scheme. In this scheme only the velocities that belong the new time level are used in the calculations. Thus, at each time level a system of algebraic equations must be solved.

$$\int_t^{t+\Delta t} u dt = u \Delta t \tag{2.79}$$

The fully implicit method is recommended for general purpose CFD computations due to its superior stability [50]. It is also the scheme used in this study. Now, the differences between the steady and unsteady momentum equations and the respective extra calculations can be explained. The whole discretization process of unsteady u-momentum equation will not be repeated. Substituting the Equations (2.75) and (2.79) into Equation (2.74), the discrete unsteady u-momentum equation takes the following form

$$(a_{i,J} + \frac{\rho \Delta V}{\Delta t}) u_{i,J} = \sum a_{nb} u_{nb} + (p_{I,J} - p_{I+1,J}) A_{i,J} + \frac{\rho \Delta V}{\Delta t} u^o_{i,J} + b_{i,J} \tag{2.80}$$

Depending on the discretisation scheme used $a_{i,J}$ and a_{nb} coefficients can be calculated from the previously derived equations for steady u-momentum equation.

Incompressible, unsteady flow calculations with fully implicit formulation can be performed by employing the SIMPLE, SIMPLER or SIMLEC algorithms that are already covered. The changes in the discretized form of the momentum equations due to the transient term are described above. The transient term does not have any effect in the final discrete continuity

equation for incompressible flows. Therefore, pressure correction equations of steady and unsteady problems are same. The only point to be paid attention in the unsteady calculations is, at each time level these algorithms should be applied until a specified convergence level is achieved.

2.6.2 Time Marching Approach for the Solution of Steady Problems

Time marching approach is an alternative way to obtain steady-state results [51]. In this method time-dependent equations is solved and steady-state results is obtained by marching in time. Since, only the steady-state results are of interest, it is not necessary to obtain converged results at each time level as in the unsteady calculations. Therefore, at each iteration, obtaining only partially converged results is enough. At successive time steps, the results approach to the steady-state values which is the idea of the method.

The application of time marching approach is quite easy due to the similarity between the steady under-relaxed and the unsteady momentum equations. Corresponding equations for these two different approaches are repeated below

$$\frac{a_{i,J}}{\alpha_u}u_{i,J} = \sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + \left[(1-\alpha_u)\frac{a_{i,J}}{\alpha_u}\right]u^*_{i,J} + b_{i,J}$$

$$(a_{i,J} + \frac{\rho\Delta V}{\Delta t})u_{i,J} = \sum a_{nb}u_{nb} + (p_{I,J} - p_{I+1,J})A_{i,J} + \frac{\rho\Delta V}{\Delta t}u^o_{i,J} + b_{i,J}$$

As seen from the above equations, the unsteady term and the under-relaxation practice have similar effects on the discretized equations. The following relation can be written between the relaxation parameter α_u and the time step Δt

$$(1-\alpha_u)\frac{a_{i,J}}{\alpha_u} = \frac{\rho\Delta V}{\Delta t} \quad \text{or} \quad \alpha_u = \frac{a_{i,J}}{a_{i,J} + \rho\Delta V/\Delta t} \tag{2.81}$$

It should be noted that in this method, the under-relaxation factor (α_u) depends on the central coefficient ($a_{i,J}$), which in turn depends on the velocities. Therefore, the relaxation parameter varies spatially (cell to cell) and evolves with the solution. This unique feature enables us to solve a flow by marching in time using the steady momentum equations.

Although it is problem dependent, generally time marching formulation shows better performance than iterative formulation by decreasing the total computational time [51]. A comperative study of these approaches are presented in Chapter 4.

2.7 Boundary Conditions

Implementation of initial and boundary conditions is an essential point for all numerical algorithms so that this process should be well understood. The initializing process is a straight forward task where the initial values of all flow variables need to be specified at all nodes in the computational domain. Thus, this topic is not discussed any furter. Implimentation of the boundary condition, on the other hand, requires special attention. In this subsection the most common three boundary condition types, namely; inlet, outlet and wall, are discussed in detail. Also the method of blocked-off region, as it is called by Patankar [6] is covered in this section.

2.7.1 Inlet

At an inlet boundary, all the flow variables have to be prescribed. Due to the staggered grid arrangement that is used in this study, $u-$ & $v-$velocity and pressure nodes are all stored in different locations. Figure 2.6 has an inlet and an outlet which are perpendicular to the x-direction and emphasize three $u-$momentum cells; in the vicinity of inlet, outlet and wall boundaries. Figures 2.7 and 2.8 are similar to the Figure 2.6, but they are emphasizing $v-$ velocity and pressure correction (any other scalar) cells in the same domain. The cells labeled as "1" are the first computational nodes where discrete momentum and pressure correction equations are solved. The grids extend outside the physical boundary (I=0) and along that boundary (i=0) are used to store the inlet values of the flow variables. Thus, for the computational cells in the vicinity of the inlet, all neighbouring nodes remain active, shown as squares, means no special treatment is necessary for these cells. Another important point is the treatment of $p^{'}$ at boundaries where velocities are prescribed, as it is the case for the inlet boundary. When the velocity is specified at a boundary, a zero gradient on $p^{'}$ normal to that boundary gives the

desired result [13]. As stated by Patankar [6] when the velocity boundary conditions are applied correctly, it implicitly asserts the correct condition on $p^{'}$; then there is no requirement for an explicit application of $p^{'}$ boundary condition. Finally, since the velocity in the inlet boundary is specified there is no need to make a velocity correction here,i.e.,

$$u_{i=0,J} = u^{*}_{i=0,J} \tag{2.82}$$

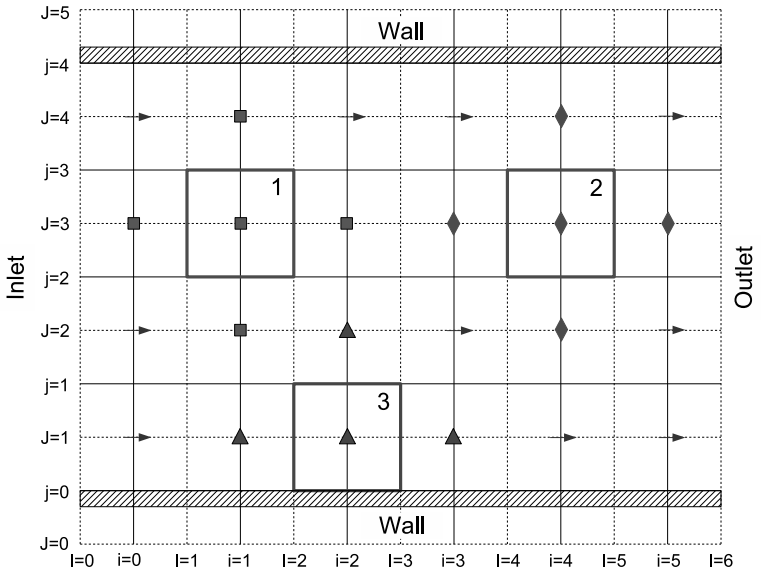

Figure 2.6: Computational Domain With Three Emhasized $u-$momentum Cells in the Vicinity of Inlet, Outlet and Wall Boundaries

2.7.2 Outlet

The Neumann boundary condition is used as the outlet boundary condition where the gradients of all flow variables, except pressure, are zero in the flow direction. The location of the outlet boundary condition is essential. It has to be selected far away from the geometrical disturbance, generally the area of interest, where the flow reaches a fully developed state. When the flow reaches fully developed state no further change occurs in the downstream of the flow. It is a common practice to locate the outlet surface perpendicular to the flow direction then take gradients in the direction normal to the outlet surface equal to zero. The last computational $u-$ & $v-$momentum and pressure cells, as displayed in diamond, in the vicinity of outlet boundary

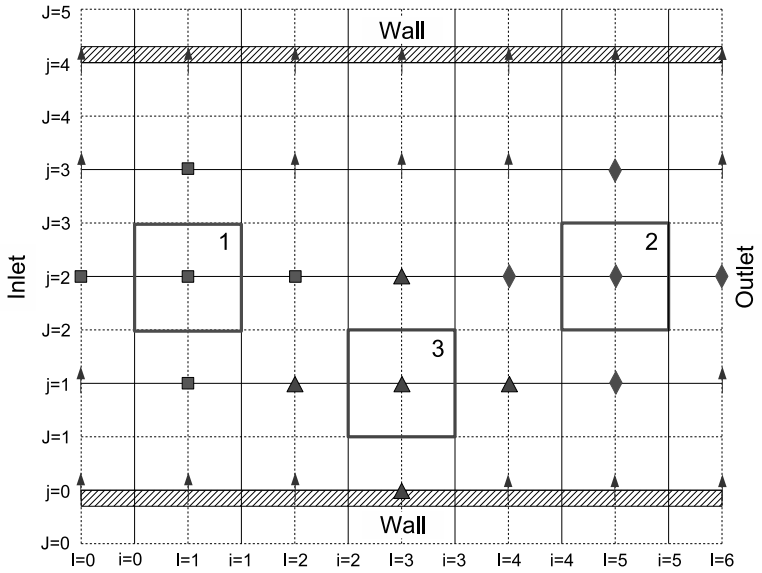

Figure 2.7: Computational Domain With Three Emhasized $v-$momentum Cells in the Vicinity of Inlet, Outlet and Wall Boundaries

condition are shown in Figures 2.6, 2.7 and 2.8 respectively.

It is seen from Figure 2.6 that the last column where the computational $u-$velocity nodes exist is i=4. Thus, the $u-$velocity nodes at i=5 are not calculated through out the iterations. As a result, before the next iteration, where the u-momentum equation will be solved, the values of u-velocities at the nodes on i=5 are updated by extrapolation from the interior on the assumption of zero gradient at the outlet plane. Similar extrapolation is performed for column I=6 to determine the values of v-velocities and pressures; this time the interior column on which the values are extrapolated is I=5. These extrapolations can be summarized as,

$$\begin{aligned} u_{i=5,J} &= u_{i=4,J} \\ v_{I=6,j} &= v_{I=5,j} \\ p_{I=6,j} &= p_{I=5,j} \end{aligned} \tag{2.83}$$

Similar to the inlet, no explicit boundary condition for $p^{'}$ is required; the appropriate boundary treatment of $p^{'}$ is implicitly assured. Also no velocity correction is needed for the nodes on the outlet boundary.

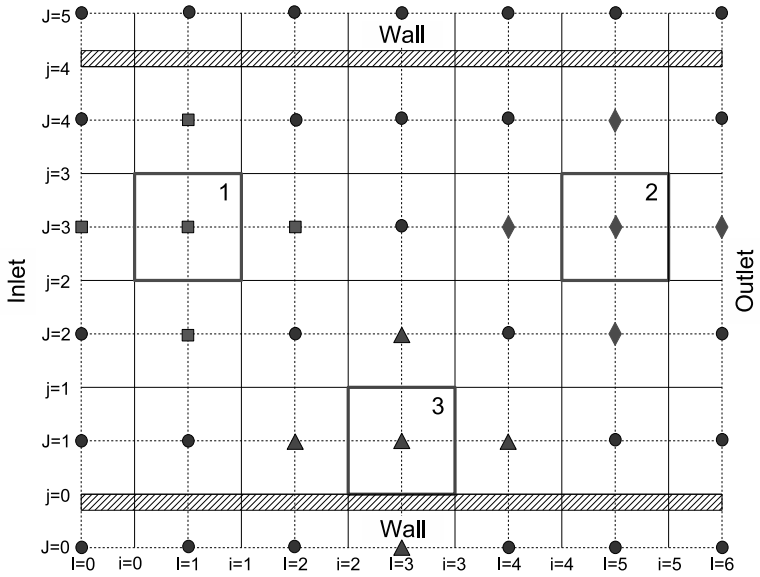

Figure 2.8: Computational Domain With Three Emhasized Pressure (and Any Scalar Variable) Cells in the Vicinity of Inlet, Outlet and Wall Boundaries

2.7.3 Wall

Wall is the most commonly encountered boundary condition in confined flow problems. The no-slip boundary condition, where both components of the velocity are equal to zero, is the appropriate condition at solid walls. Depending on the direction of the wall either $u-$ or $v-$ momentum cells require special attention. For the case that is discussed here where the walls are in horizontal pozition, as shown in Figure 2.7, the $v-$ cells at the row j=1 can be evaluated without modification like any other v-cell in the domain. The $u-$cells, on the other hand, at the wall boundaries can not be accounted in that level of simplicity. The presence of the wall have to be taken into and added to the equation as a source term. It is assumed that the velocity varies linearly with distance from the wall in the laminar flow as illustrated in Figure 2.9-a. Thus, the discretised u-momentum equation (Equation 2.46) for the boundary node which can be restated by taking the velocity of the south cell (ghost-cell) is equal to the negative of the center cell.

$$\underbrace{a_{i,J}\mathbf{u_{i,J}}}_{center} = \underbrace{a_{i+1,J}u_{i+1,J}}_{east} + \underbrace{a_{i-1,J}u_{i-1,J}}_{west} + \underbrace{a_{i,J+1}u_{i,J+1}}_{north} - \underbrace{a_{i,J-1}\mathbf{u_{i,J}}}_{wallside}$$

$$+(p_{I,J} - p_{I+1,J})A_{i,J} + b_{i,J} \tag{2.84}$$

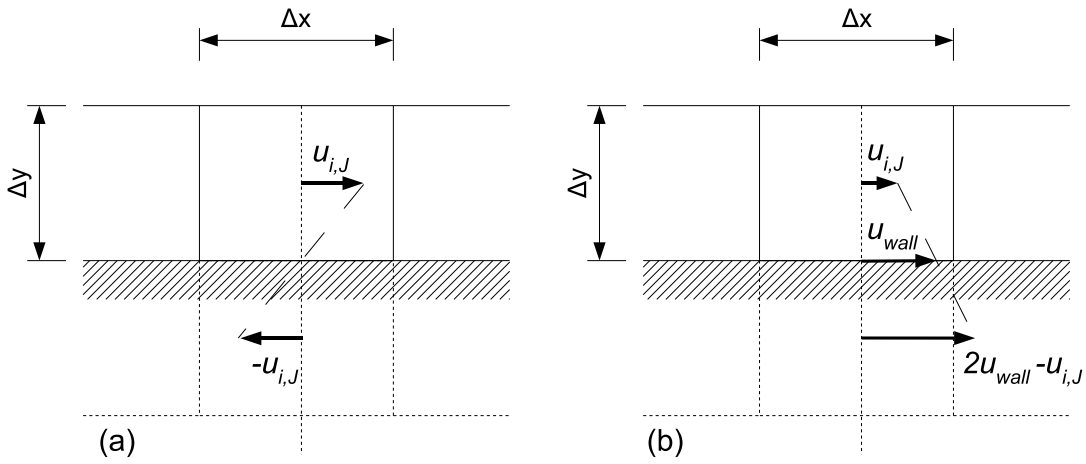

Figure 2.9: u-momentum Cells in the Vicinity of Stationary (a) and Sliding (b) Boundaries.

The coefficient of the wallside cell ($a_{i,J-1}$) is composed of convective and diffusive terms. Since the velocity of the south cell is zero, there exists no convective term and the diffusive term will be $\mu\Delta x/\Delta y$. Finally, collecting the common terms together, the u-momentum cell in the vicinity of the wall will be:

$$\underbrace{(a_{i,J}+\frac{\mu\Delta x}{\Delta y})u_{i,J}}_{center}=\underbrace{a_{i+1,J}u_{i+1,J}}_{east}+\underbrace{a_{i-1,J}u_{i-1,J}}_{west}+\underbrace{a_{i,J+1}u_{i,J+1}}_{north}$$

$$+(p_{I,J}-p_{I+1,J})A_{i,J}+b_{i,J} \tag{2.85}$$

It is seen that the only difference between the u-momentum equations of an inner and a boundary cell is the coefficient of center cell.

Sliding wall boundary condition is very similar to stationary wall boundary condition. Depending on the direction of the wall movement one of the velocity components is non zero, as illustrated in the Figure 2.9-b. In this case, based on the linear velocity profile assumption, the u-velocity at the south cell (ghost-cell) is found by extrapolation using the velocity of the wall and the first inner cell. Thus, the u-momentum takes the following form.

$$\underbrace{a_{i,J}\mathbf{u_{i,J}}}_{center}=\underbrace{a_{i+1,J}u_{i+1,J}}_{east}+\underbrace{a_{i-1,J}u_{i-1,J}}_{west}+\underbrace{a_{i,J+1}u_{i,J+1}}_{north}+\underbrace{2u_{wall}-a_{i,J-1}\mathbf{u_{i,J}}}_{wallside}$$

$$+(p_{I,J}-p_{I+1,J})A_{i,J}+b_{i,J} \tag{2.86}$$

Extracting the coefficient of the wallside cell and collecting the common terms together, the equation becomes,

$$\underbrace{\big(a_{i,J}+\frac{\mu\Delta x}{\Delta y}\big)u_{i,J}}_{center}=\underbrace{a_{i+1,J}u_{i+1,J}}_{east}+\underbrace{a_{i-1,J}u_{i-1,J}}_{west}+\underbrace{a_{i,J+1}u_{i,J+1}}_{north}$$

$$+(p_{I,J}-p_{I+1,J})A_{i,J}+b_{i,J}+\frac{\mu u_{wall}\Delta x}{\Delta y/2} \tag{2.87}$$

In the sliding wall case, not only the coefficient of the central cell is changed but also a new term is added to the right handside source.

2.7.4 Blocked-Off Regions

The solver that is developed through out this research is working on Cartesian structured grids which is limited in handling irregular shaped computational domains. The blocked-off region method of Patankar [6] can be a perfect tool to remove this weakness. In this method, by rendering the inactive control volumes, the desired irregular computational domain is obtained. The block-off operation is simply the assignment of known values of relevant variables to inactive control volumes. Recalling the u- and v- momentum equations and inserting two new source terms; one to the coefficient of central cell (S_P) and one to the righthand side source (S_C). The equations take the following form,

$$\begin{aligned}
\big(a_{i,J}S_{P,u}\big)u_{i,J}&=\sum a_{nb}u_{nb}+(p_{I,J}-p_{I+1,J})A_{i,J}+b_{i,J}+S_{C,u} \qquad \text{u-momentum}\\
\big(a_{I,j}S_{P,v}\big)v_{I,j}&=\sum a_{nb}v_{nb}+(p_{I,J}-p_{I,J+1})A_{I,j}+b_{I,j}+S_{C,v} \qquad \text{v-momentum}
\end{aligned} \tag{2.88}$$

At this point it is seen that if large enough source terms (S_P,S_C) are employed, as shown in the Equation 2.89, all the other terms in the discretization equations will become negligible.

$$\begin{aligned}
S_{C,u} &= 10^{30}u_{desired}\\
S_{C,v} &= 10^{30}v_{desired}\\
S_{P,u} &= 10^{30}\\
S_{P,v} &= 10^{30}
\end{aligned} \tag{2.89}$$

Thus, the u- & v- velocities in the node of interest (central node) can be calculated in the following form,

$$u_{i,J} = \frac{S_{C,u}}{S_{P,u}} = u_{desired}$$
$$v_{I,j} = \frac{S_{C,v}}{S_{P,v}} = v_{desired} \tag{2.90}$$

It is obvious that, if $u_{desired}$ and $v_{desired}$ are set equal to zero, then internal obstacles (stationary solid boundaries) can be created inside the calculation domain using this method.

The major disadvantage of this method is its inefficiency in CPU time and storage. Although the values of flow variables are known at these inactive cells, it is still necessary to store and perform trivial computations for that zone.

CHAPTER 3

VIRTUAL FLOW LAB SOFTWARE

The flow solver developed in this study is embeded into a Graphical User Interface (GUI) environment [52] and the combined software is called as Virtual Flow Lab (VFL) [53]. This chapter provides a general overview of its Graphical User Interface (GUI) and presents its capabilities.

Most CFD codes written as part of an academic study include not more than a bare flow solver. In such an approach the user first needs to draw the problem geometry and prepare the mesh using a mesh generation software. Then an input file that includes the solver parameters, boundary and initial conditions, etc. is prepared. The generated mesh and this input file are fed into the solver to obtain the numerical results. These results need to be analyzed using a visualization software. This is definitely not a user-friendly approach. It is also not economical due to the need of separate pre- and post-processing software. The learning curve of such a software bundle is too steep to be used for educational purposes.

VFL, on the other hand, is written to be a complete CFD package, which combines all necessary sub-components in a single, user friendly enviroment. VFL is an open source software. It is distributed under the GPL license [45], which means that it can be downloaded freely from the internet. The source codes can also be downloaded freely and if necessary they can

be modified by the users according to their specific needs. It is written in C++ language. User interfaces are designed with Qt [46], a C++ user interface library freely available from Trolltech. It is a platform independent software which can be compiled and run under different computer architectures without any or very little source code change. It is successfully tested under Linux and Windows operating systems. It is a multi-lingual software, which means that the user interface can be used in different languages. Using the tools provided by Trolltech, it is possible to translate the whole user interface from one language to another in just few hours. The VFL project is still under development and the capabilities of the current version will be presented in the following paragraphs.

The opening screenshot of VFL is given in Figure 3.1. It is composed of a main menu bar, operation toolpad, graphical window and communication box. Main menu bar is used for operations like opening/saving a problem, setting general user interface options, capturing a screenshot, accessing the help files. Operation toolpad is composed of six tabs, namely Geometry, Blocks/Faces, Mesh, BC/IC, Solve and Visualize. The task of each tab can be followed easily by its name. Graphical window is used to draw and display the problem geometry and the generated mesh, select blocked-cells and control points and visualize the results. Communication box is used to provide feedback about the operations performed. Warning and error messages can be followed from this window. Once the solution starts, it is also used to watch the convergence and the variation of variables at the control points.

Figure 3.2 shows a screenshot taken during the creation of a problem geometry. The user can draw the geometry of the problem by using elementary geometrical entities such as lines and circular arcs. After drawing a closed geometry mesh points on the boundaries of the problem can be selected using the Blocks/Faces tab. Then a multi-block structured mesh can be generated using either algebraic of differential mesh generation methods as demonstrated in Figure 3.3 [54].

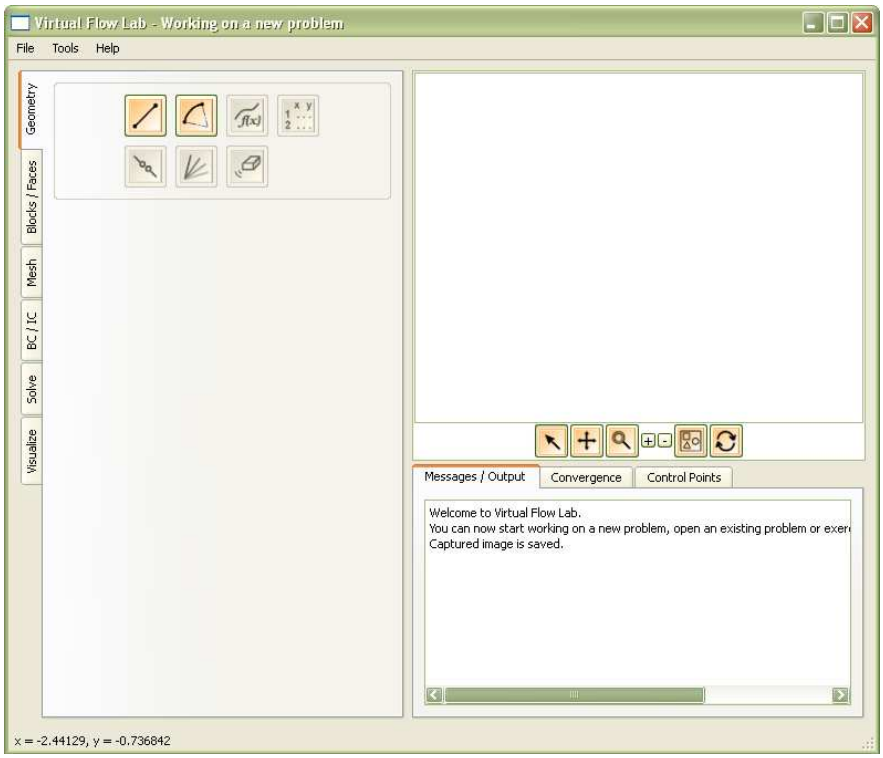

Figure 3.1: Opening Screenshot of VFL

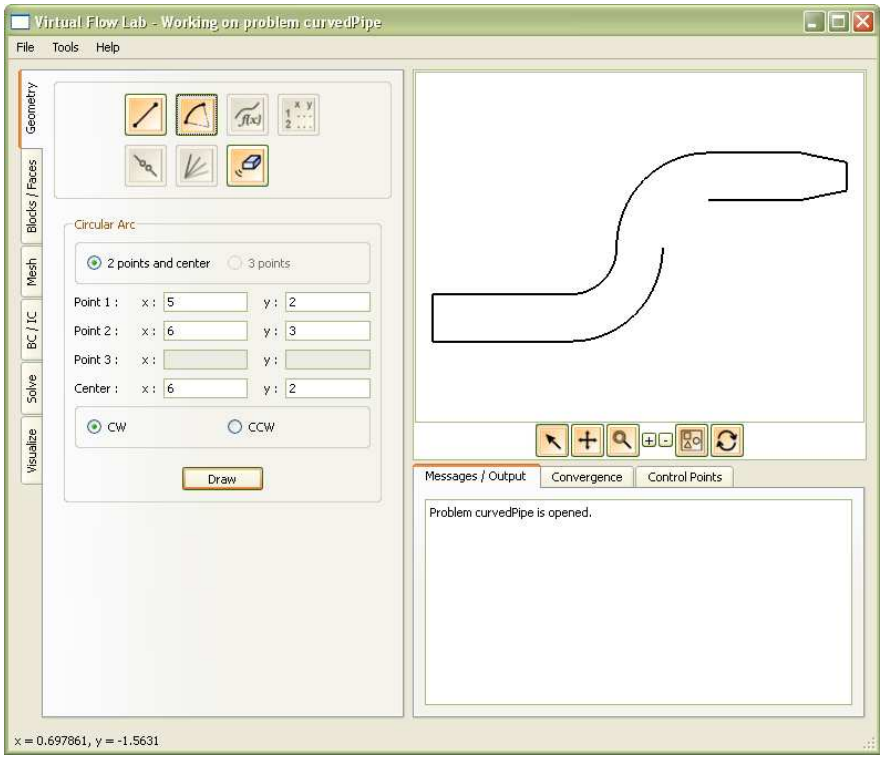

Figure 3.2: A Screenshot of VFL Taken During the Creation of a Problem Geometry

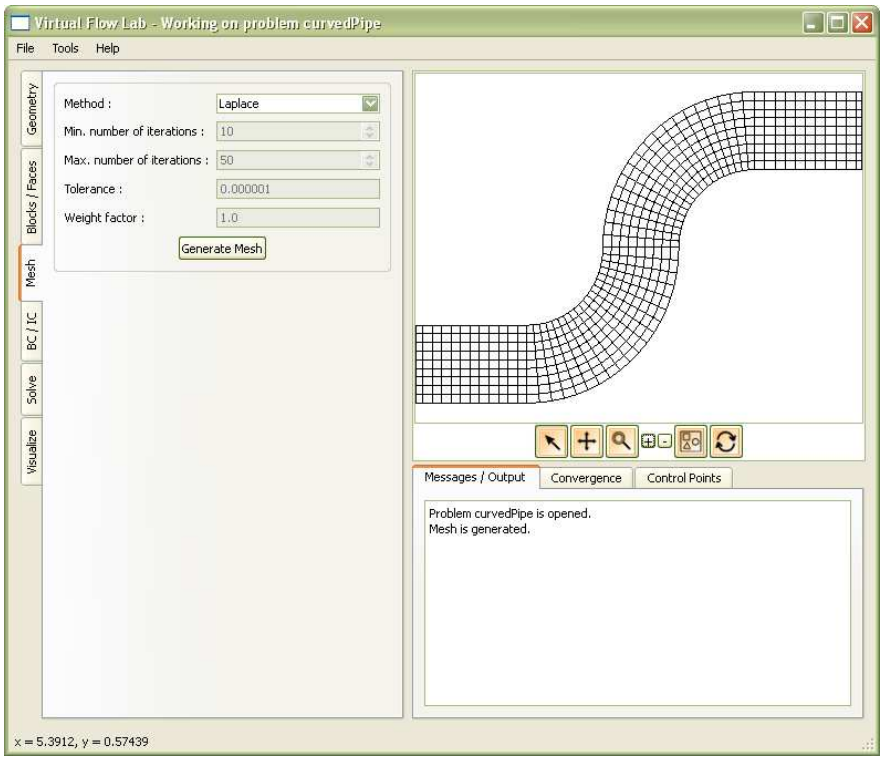

Figure 3.3: A Screenshot of VFL Taken During Mesh Generation

As shown in Figure 3.4, BC/IC tab is used to specify boundary and initial conditions. The current version of the software supports three most commonly used boundary condition types: inflow, outflow and wall. Specification of fluid properties (e.g. density, kinematic viscosity), solver parameters (e.g. relaxation parameters, time step size, convergence tolerance, etc.) and the convergence monitoring capability of the software can be seen in Figure 3.5. As explained in Chapter 2 the incompressible flow solver is based on the SIMPLE algorithm of Patankar [6]. Its variants SIMPLER and SIMPLEC are also available. Space discretization can be made using one of the central, upwind, power-law or hybrid choices. It is possible to perform steady or unsteady simulations.

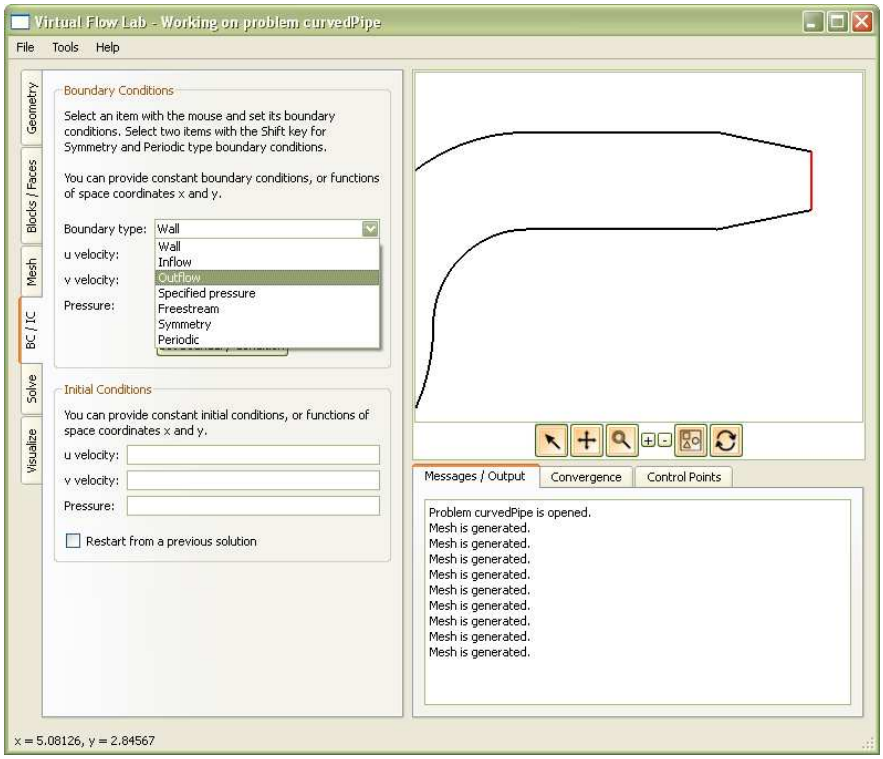

Figure 3.4: A Screenshot of VFL Taken During the Specification of Boundary and Initial Conditions

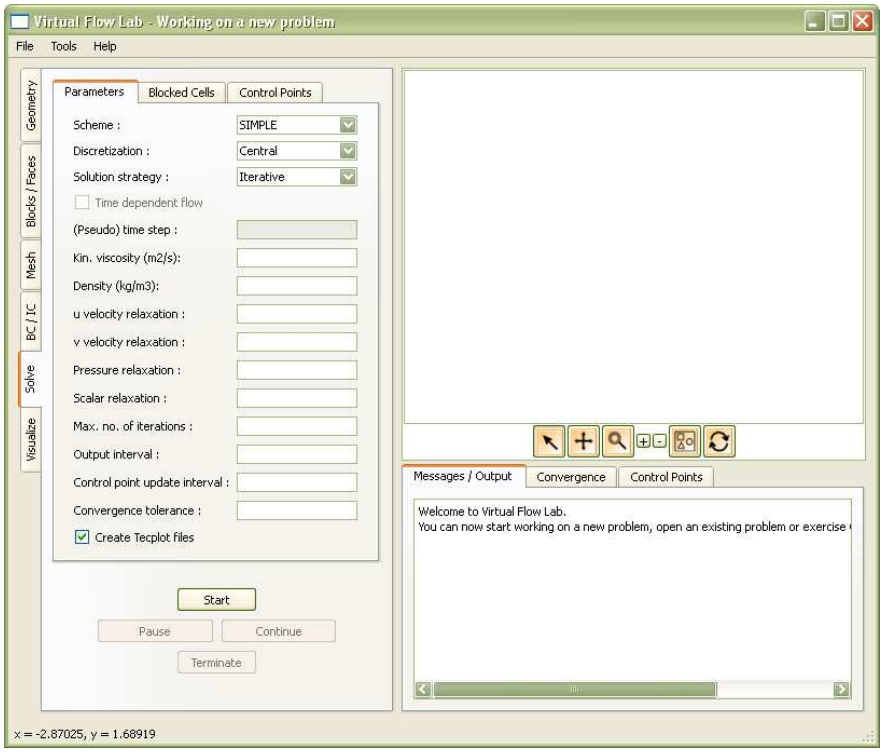

Figure 3.5: A Screenshot of VFL Showing the Convergence Monitoring of a Running Simulation

As seen in Figure 3.6 it is very simple to create complex geometries by selecting proper blocked cells in a Cartesian mesh, which will be treated as walls in a simulation by using the block-off region method of Patankar [6]. Other than the real-time convergence monitoring, it is also possible to select desired number of control points and visualize the convergence of the solution by following the variations of flow properties at these points.

While running a simulation, the results can be post-processed as seen in Figure 3.7. It is possible to draw contour plots of flow variables, place streamlines and probe the properties of any point in the flow field. The user can also save the contents of the graphical window and convergence and control point monitoring windows as images.

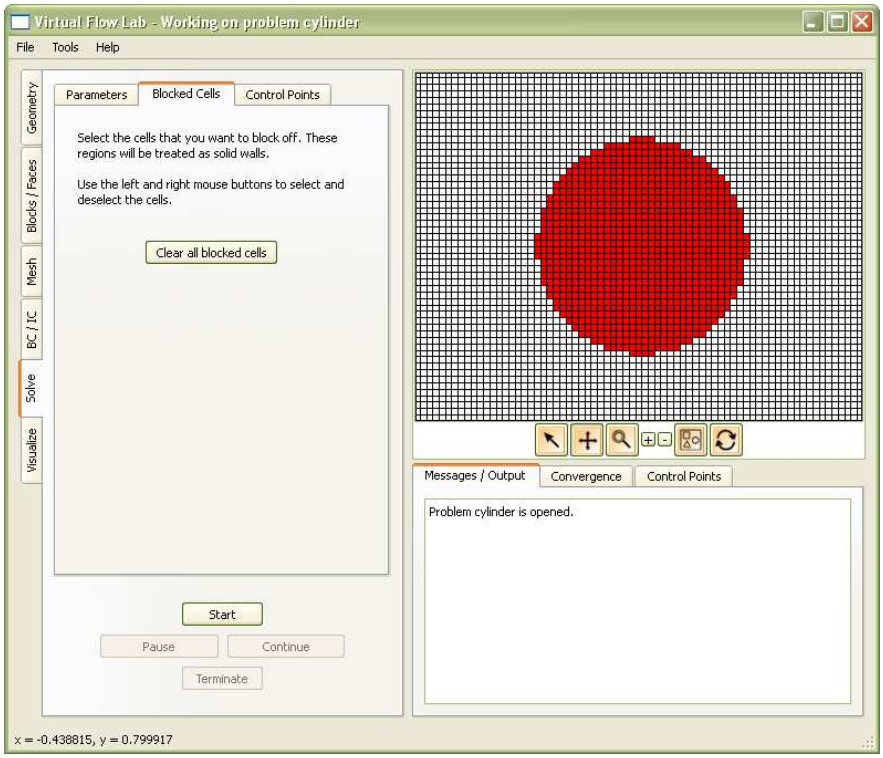

Figure 3.6: A Screenshot of VFL Taken During the Selection of Blocked Cells

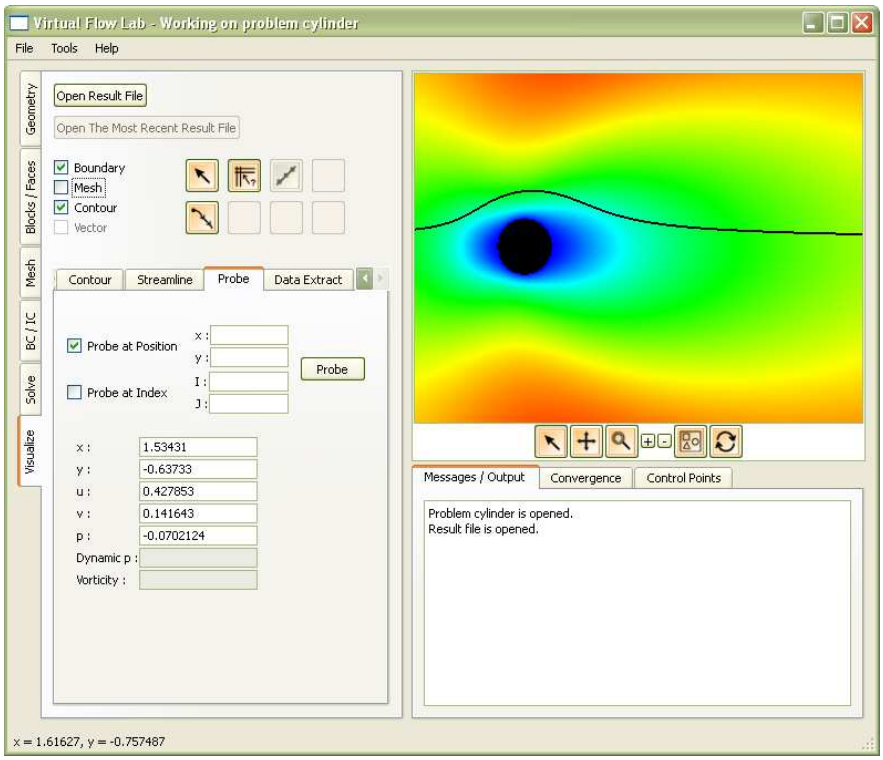

Figure 3.7: A Screenshot of VFL Showing Its Post-Processing Capabilities

CHAPTER 4

VALIDATION of the FLOW SOLVER

The flow solver developed in this study is validated by solving a number of test problems. Several factors that affect the accuracy and the efficiency of the solutions, such as solution methods, discretization schemes, relaxation parameters, grid sizes, etc. are examined in detail. The studied test problems are as follows; 1. Lid-driven square cavity flow, which is a standard validation test case due to its simple geometry and boundary conditions, 2. Back facing step flow, which is another popular validation case for an internal flow with recirculating regions, 3. Unsteady confined flow over a square cylinder, a widely refered benchmark problem. Obtained results are compared with experimental and computational results available in the literature.

The reported computing times are the CPU seconds obtained on a Hewlett-Packard Compac dc7100 personal computer with an Intel ®, Pentium ® 4 3.00Ghz processor and 512 MB of memory. The iterations are performed until the mass residual (the largest cell mass source term, b term in the pressure correction equation (2.59)) drops below 10^{-9} in the domain. This convergence criteria always satisfies that u and v velocity residuals in the domain are also smaller than 10^{-9}. Unless noted otherwise, this convergence criteria is applied to all runs in this work as suggested by Patankar [6]. Numerical studies, which are presented in the tables, are not labeled; whereas, experimental studies are labeled as "E".

4.1 Problem 1. Lid-Driven Cavity

The numerical solution of the flow in a two-dimensional square cavity with the top wall sliding at a constant velocity is one of the most popular benchmark problems for testing new solution techniques and flow solvers. Since the early work by Burggraf [55], this problem has served over and over again as a standard problem for the validation of Navier-Stokes codes, in spite of the singularities at the two top corners where the velocity is discontinuous. It provides a good test case because there is no primary flow direction, a structured Cartesian grid fits well into the simple problem domain and the boundary conditions are of single type.

Generally, two sets of data are presented in the lid-driven cavity analyses. The first one is the u-velocity and the v-velocity profiles along the vertical and horizontal center lines, respectively and their extrema, as shown in Figure 4.1-a. The other set of data is the locations of primary, secondary and for some cases ternary vortices, as shown in Figure 4.1-b.

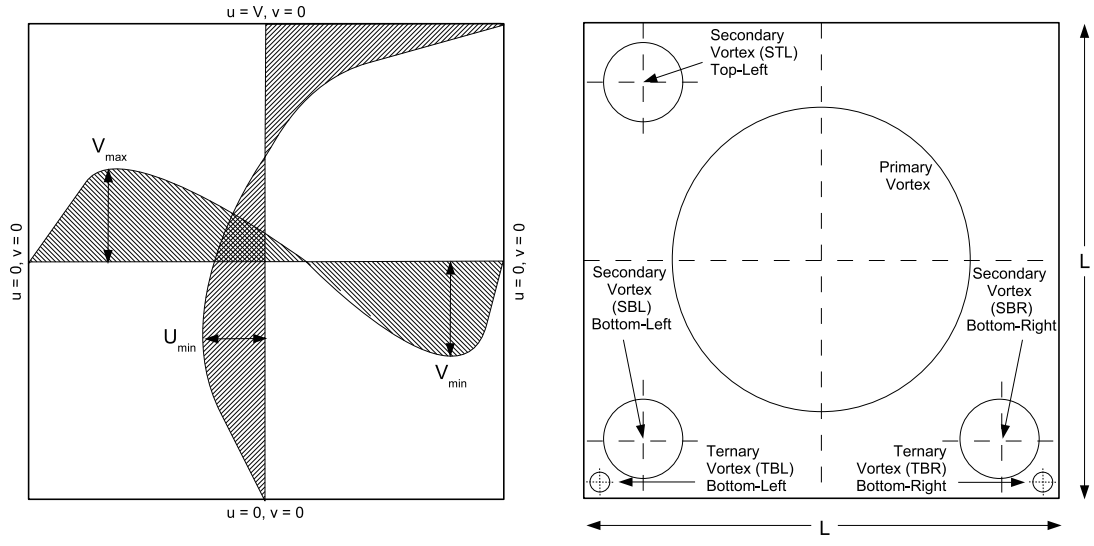

Figure 4.1: (a) Velocity Profiles Along the Centerlines, (b) Definition of the Vortices and Corresponding Indexing for Problem 1

4.1.1 Mesh Independency Study

The first set of runs are performed for the mesh independency study to see the affect of the grid size on the accuracy of the solution. For this purpose, uniformly spaced grids of 8×8, 16×16, 32×32, 64×64, 128×128, 256×256 are used to calculate the flow field for a Reynolds number of $Re = UL/\nu = 100$. For this set of runs steady flow equations are solved using central differencing scheme and the SIMPLE algorithm.

Calculated velocity profiles along the centerlines of the cavity are shown in Figures 4.2 and 4.3. With the increasing number of grid points the computed extrema of u and v velocities along the centerlines are approaching to their correct values. Surprisingly, even an 8×8 mesh can capture the general characteristics of the flow field.

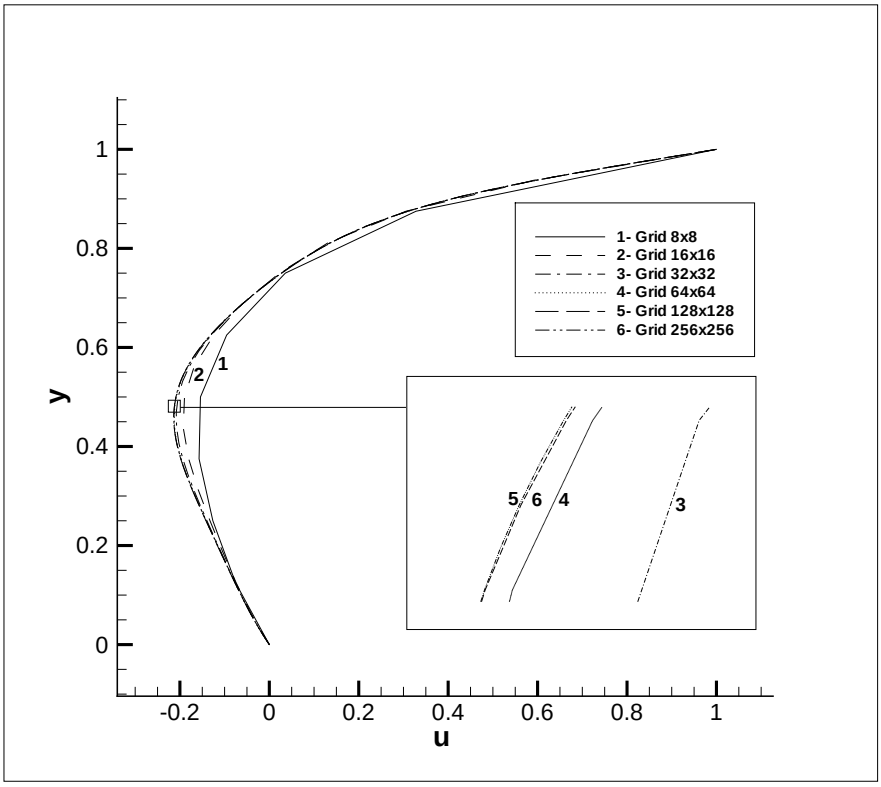

Figure 4.2: Comparison of u-velocity Along the Vertical Centerline Using Different Mesh Sizes (Re=100), Problem 1

But for a more solid comparison the location of the vortices and the extrema of the velocities along the centerlines should also be taken into account. These data are presented in Table 4.1

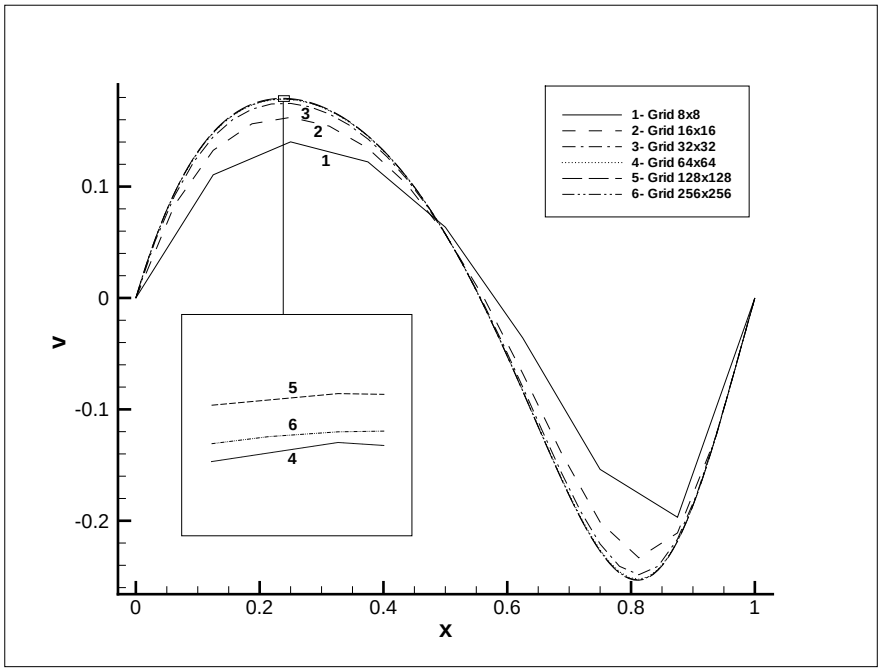

Figure 4.3: Comparison of v-velocity Along the Horizontal Centerline Using Different Mesh Sizes (Re=100), Problem 1

and 4.2, respectively.

Table 4.1: Center Locations of the Vorticies, Re=100, Problem 1

	Primary Vortex	Secondary Vortex Bottom Right	Secondary Vortex Bottom Left
Grid 8 ×8	(0.6237, 0.7190)	(N/A)	(N/A)
Grid 16 ×16	(0.6215, 0.7389)	(N/A)	(N/A)
Grid 32 ×32	(0.6167, 0.7380)	(0.9462, 0.0622)	(N/A)
Grid 64 ×64	(0.6154, 0.7375)	(0.9438, 0.0619)	(0.0343, 0.0347)
Grid 128 ×128	(0.6159, 0.7373)	(0.9427, 0.0618)	(0.0344, 0.0348)
Grid 256 ×256	(0.6162, 0.7374)	(0.9426, 0.0616)	(0.0342, 0.0344)
Ghia et al. [56]	(0.6172, 0.7344)	(0.9453, 0.0625)	(0.0313, 0.0391)
Bruneau et al. [57]	(0.6172, 0.7344)	(0.9453, 0.0625)	(0.0313, 0.0391)
Vanka [58]	(0.6188, 0.7375)	(0.9375, 0.0563)	(0.0375, 0.0313)
Goyon [59]	(0.6172, 0.7343)	(0.9453, 0.0625)	(0.0312, 0.0390)
Schreiber et al. [60]	(0.6167, 0.7417)	(0.9417, 0.0500)	(0.0333, 0.0250)

It is seen from the tables that the results are in agreement with the ones in the literature. It can be noted from the Table 4.1 that mesh sizes less than 64×64 fail to reveal the secondary vortices, and for higher Reynolds numbers, where ternary vortices start to appear, this inability will be more obvious. Another drawback of coarse mesh is about the convergence. As the Re number increases it becomes more difficult, if possible at all, to get converged results. For ex-

Table 4.2: Characteristic Values for the Driven Cavity, Re =100, Problem 1

	y_{min}	U_{min}	x_{max}	V_{max}	x_{min}	V_{min}
Grid 8 ×8	0.3749	-0.1571	0.2500	0.1401	0.8750	-0.1969
Grid 16 ×16	0.4375	-0.1924	0.2500	0.1621	0.8125	-0.2332
Grid 32 ×32	0.4687	-0.2082	0.2500	0.1747	0.8125	-0.2485
Grid 64 ×64	0.4530	-0.2126	0.2344	0.1787	0.8125	-0.2525
Grid 128 ×128	0.4609	-0.2136	0.2343	0.1792	0.8125	-0.2534
Grid 256 ×256	0.4576	-0.2136	0.2344	0.1792	0.8125	-0.2529
Ghia et al. [56]	0.4531	-0.2109	0.2344	0.1753	0.8047	-0.2453
Bruneau et al. [57]	0.4531	-0.2106	0.2344	0.1786	0.8125	-0.2521
Vanka [58]	0.4578	-0.2130	-	-	-	-
Deng et al. [61]	-	-0.2132	-	0.1790	-	-0.2534
Botella et al. [62]	0.4581	-0.2140	0.2370	0.1796	0.8104	-0.2538

ample, irrespective of the solution algorith that is employed or the relaxation parameters that are used, it is impossible to get a converged result for Re = 1000 with a 8×8 mesh.

Choosing the appropriate mesh size is a kind of optimization problem between the CPU time and accuracy. The CPU time and the corresponding converged iteration numbers are given in the Table 4.3. For the sake of controlled numerical experimentation, only one parameter, namely the mesh sizes, are changed; while all the other solution parameters are kept the same. As a result, the presented iteration numbers and the CPU times are not necessarily the optimum values; but can still be used for comparison. As expected as the mesh gets finer, both the convergence iteration number and the CPU time increase.

Table 4.3: CPU Time and Convergence Iteration Number, Re = 100, Problem 1

	CPU Time (s)	Convergence Iteration Number
Grid 8 ×8	0.14	95
Grid 16 ×16	0.42	242
Grid 32 ×32	2.63	737
Grid 64 ×64	26.75	2304
Grid 128 ×128	318.05	6909
Grid 256 ×256	3683.16	18558

4.1.2 Comparative Study of Different Reynolds Numbers

In this set of numerical experiments, the lid driven cavity problem is solved for three different Reynolds number, namely, Re = 100, Re = 400 and Re = 1000, and the results are compared with the studies in the literature. Based on the discussion made in the previous section, the uniform mesh size of 128×128 is chosen.

u- and v-velocity contours and streamtraces for Re = 100 are given in Figures 4.4, 4.5 and 4.6, respectively. Similar plots for Re = 1000 can be seen in Figures 4.7 , 4.8 and 4.9. It can be seen that in both case two secondary vortices appear which are on the left and right bottom corners. It can also be noted from Re = 1000 case that a third secondary vortex will appear on the left-top corner, if Reynolds number is continue to be increased. Another fact that can be figured out is that, as the Reynolds number increases, the center location of the primary vortex moves downward.

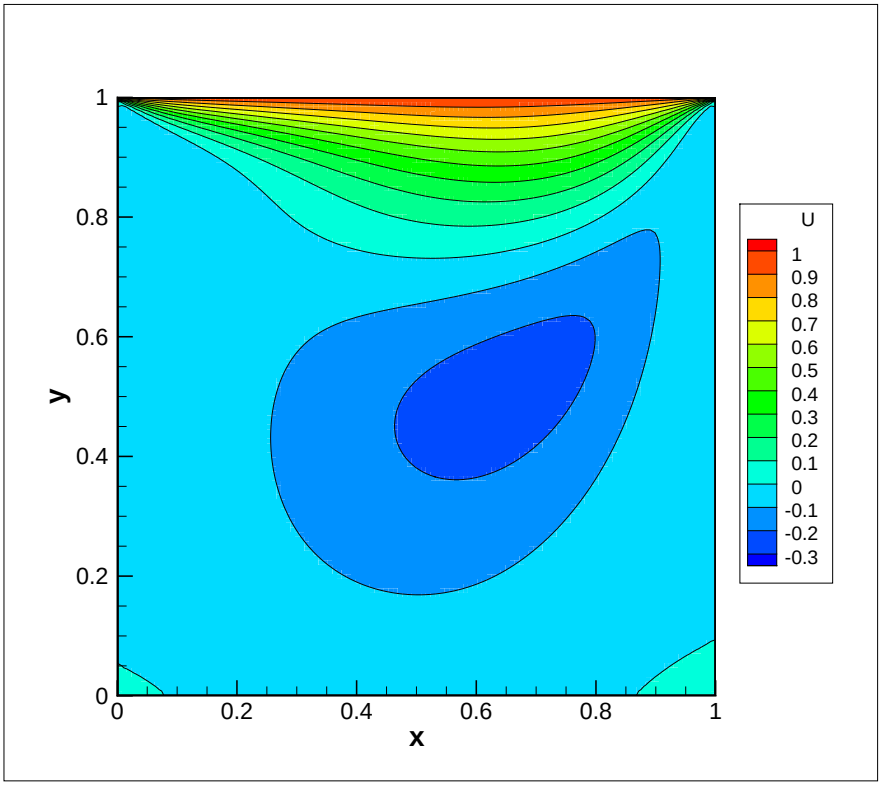

Figure 4.4: u-velocity Contour, Re = 100, Problem 1

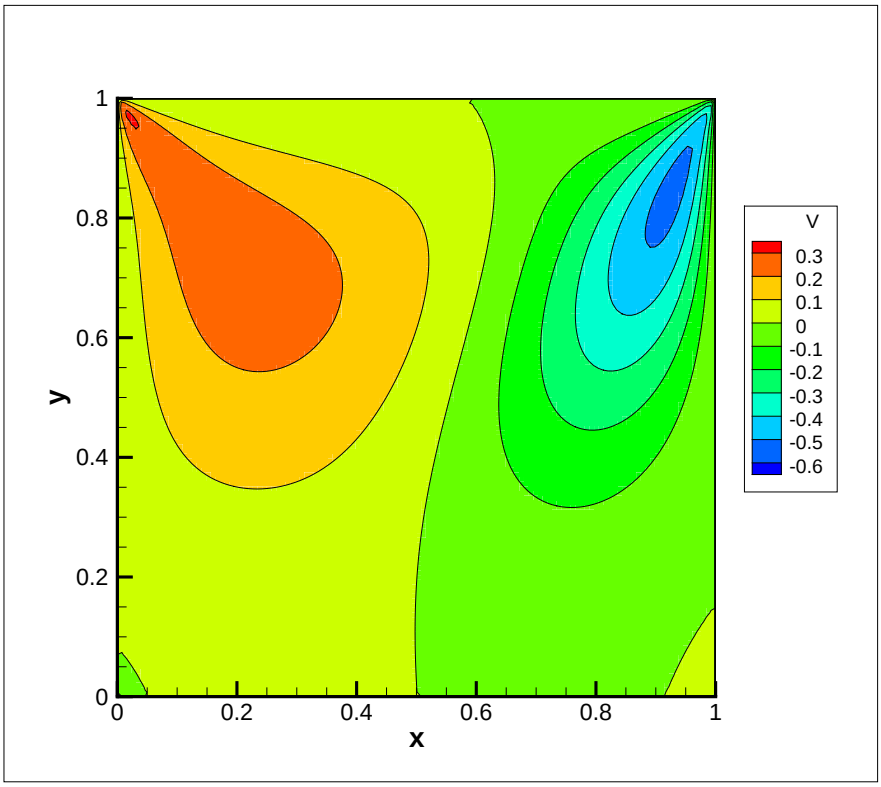

Figure 4.5: v-velocity Contour, Re = 100, Problem 1

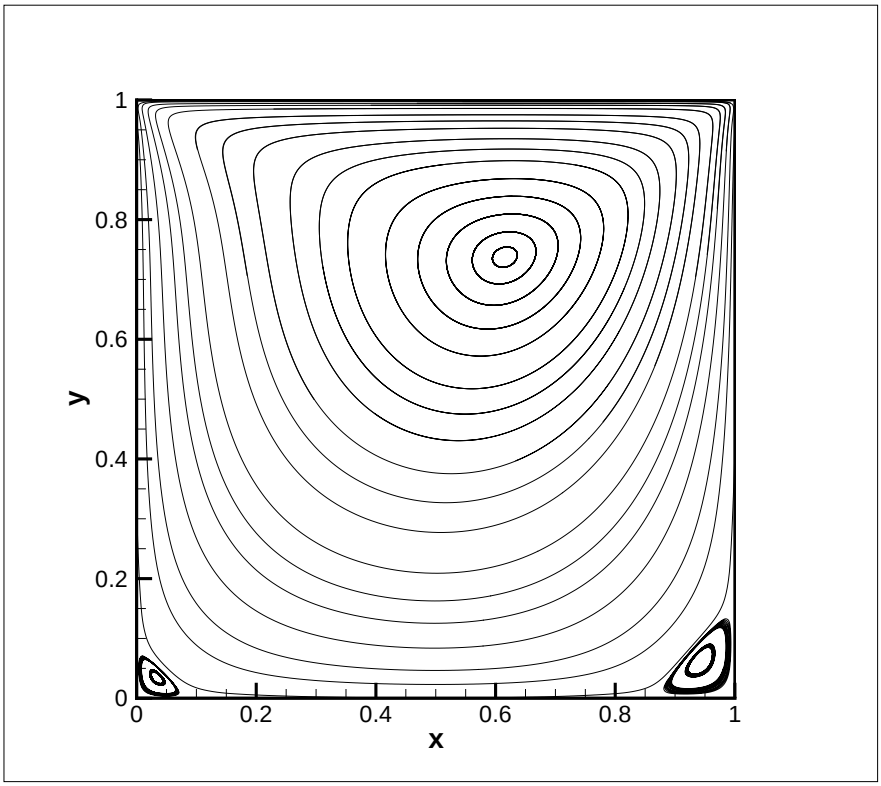

Figure 4.6: Streamtraces, Re = 100, Problem 1

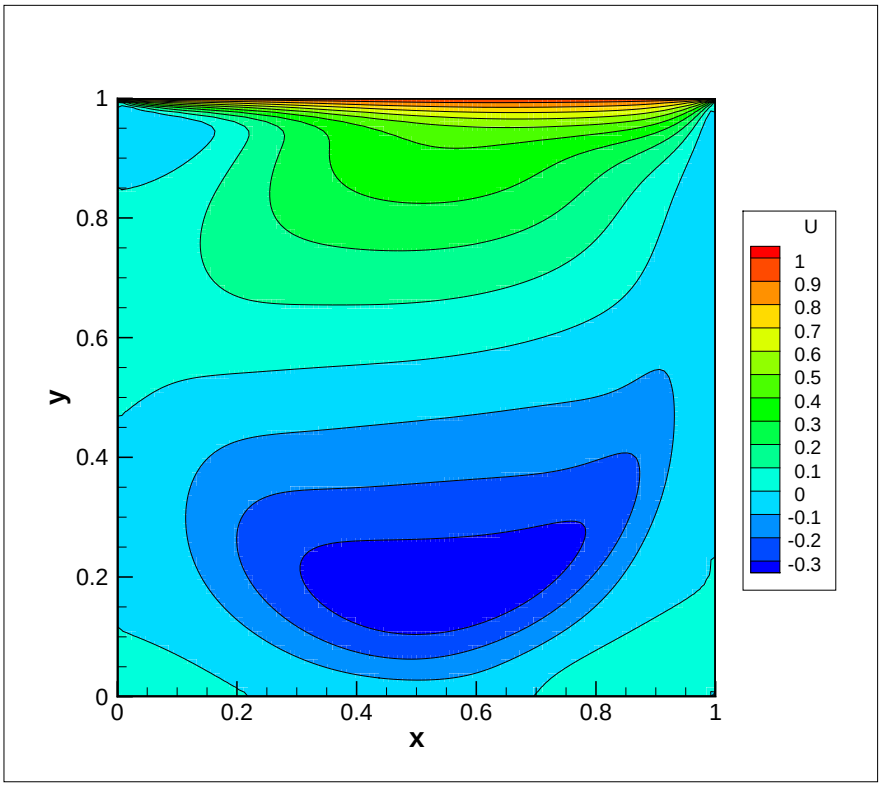

Figure 4.7: u-velocity Contour, Re = 1000, Problem 1

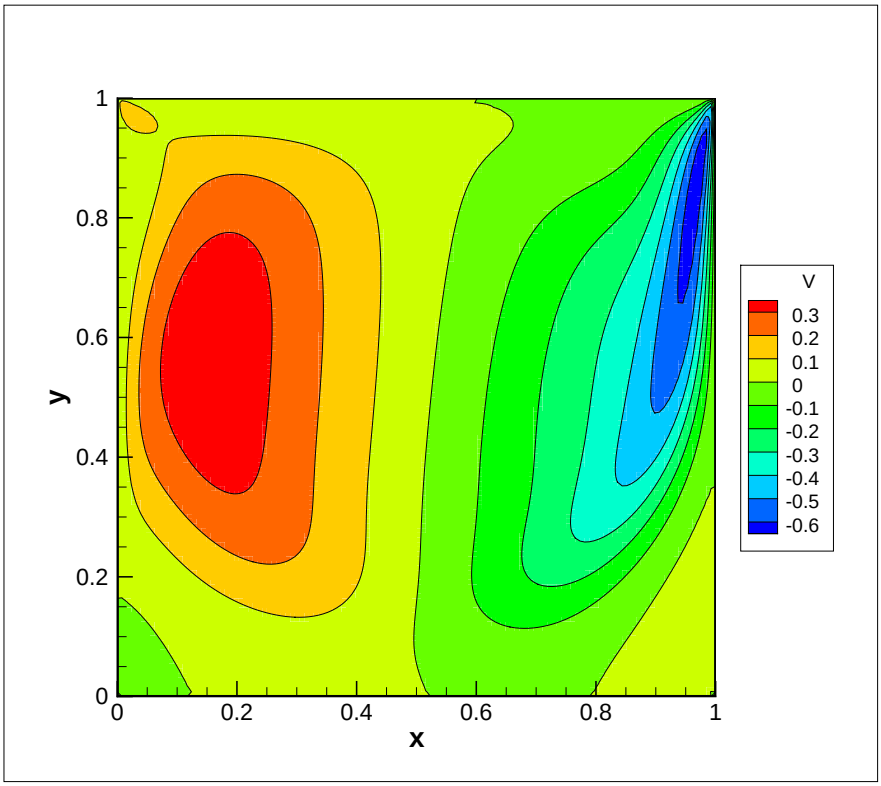

Figure 4.8: v-velocity Contour, Re = 1000, Problem 1

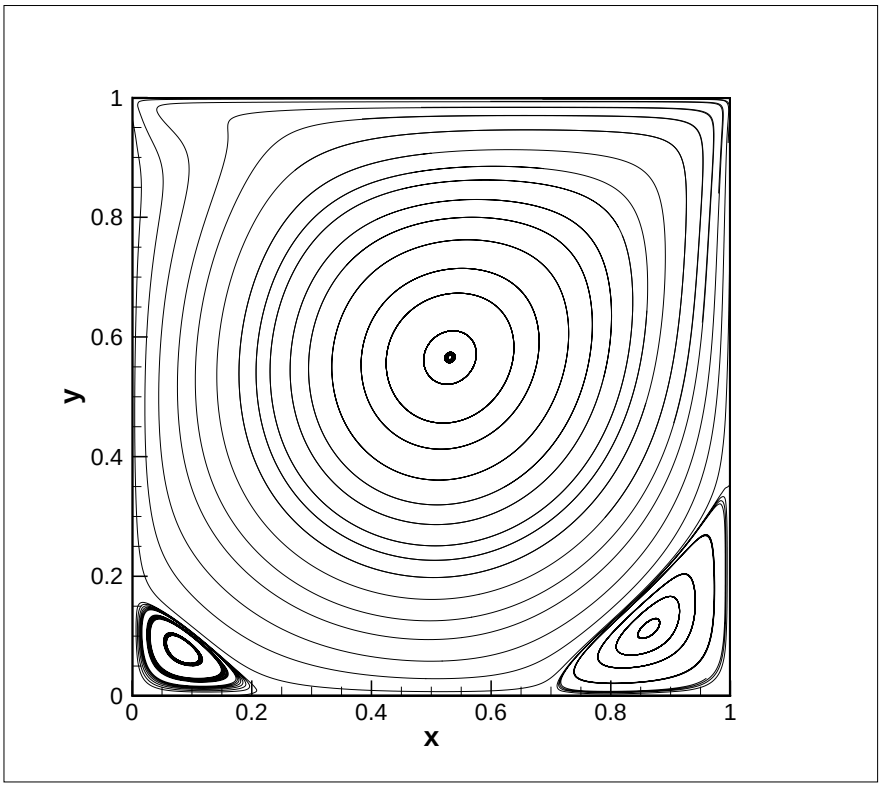

Figure 4.9: Streamtraces, Re = 1000, Problem 1

u-velocities along the vertical centerline and the v-velocities along the horizontal centerline are given in Figures 4.10 and 4.11, respectively. The extrema of the velocity profiles along the centerlines are compared with the available data in literature in Table 4.4. Results are satisfactorily close to the values in the literature.

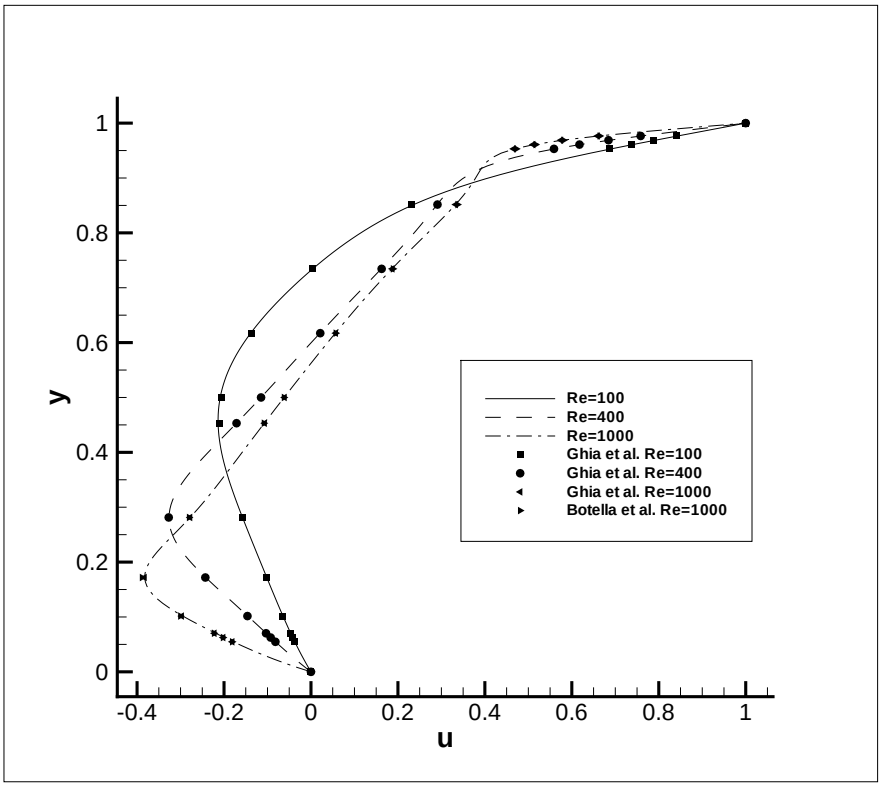

Figure 4.10: Comparison of u-velocity Along Vertical Line Through Geometric Center for Different Reynolds Numbers, Problem 1

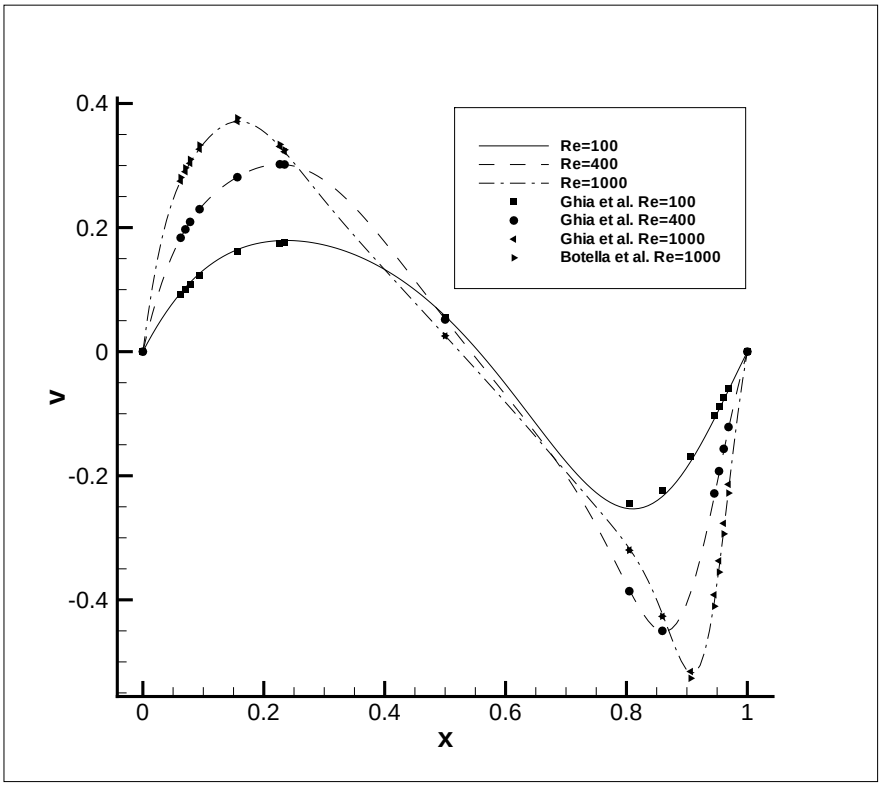

Figure 4.11: Comparison of v-velocity Along Horizontal Line Through Geometric Center for Different Reynolds Numbers, Problem 1

Table 4.4: Characteristic Values, Re = 400 & Re = 1000, Problem 1

	y_{min}	U_{min}	x_{max}	V_{max}	x_{min}	V_{min}
Re = 400						
Present Study	0.2813	-0.3265	0.2266	0.3019	0.8594	-0.4511
Ghia et al. [56]	0.2813	-0.3273	0.2266	0.3020	0.8594	-0.4499
Deng et al. [61]	-	-0.3275	-	0.3027	-	-0.4527
Re = 1000						
Present study	0.1719	-0.3824	0.1563	0.3710	0.9063	-0.5181
Ghia et al. [56]	0.1719	-0.3829	0.1563	0.3710	0.9063	-0.5155
Bruneau et al. [57]	0.1602	-0.3764	0.1523	0.3665	0.9102	-0.5208
Vanka [58]	0.1680	-0.3798	0.1563	0.3669	0.9102	-0.5186
Deng et al. [61]	-	-0.3851	-	0.3737	-	-0.5228
Zhang [63]	0.1699	-0.3901	0.1582	0.3785	0.9082	-0.5284

The locations of the primary and secondary vortices are another set of parameters that can be compared with the previous studies, Table 4.5 (This comparison for Re = 100 was done in the previous section). It can be noticed that the results are in agreement with each other. The data can also be observed from Figure 4.12.

Table 4.5: Center Locations of the Vorticies, Re = 400 & Re = 1000, Problem 1

	Primary Vortex	Secondary Vortex Bottom Right	Secondary Vortex Bottom Left
Re = 400			
Present study	(0.5544,0.6057)	(0.8865,0.1227)	(0.0509,0.0470)
Ghia et al. [56]	(0.5547,0.6055)	(0.8906,0.1250)	(0.0508,0.0469)
Vanka [58]	(0.5563,0.6000)	(0.8875,0.1188)	(0.0500,0.0500)
Re = 1000			
Present study	(0.5312,0.5652)	(0.8634,0.1124)	(0.0832,0.0779)
Ghia et al. [56]	(0.5313,0.5625)	(0.8594,0.1094)	(0.0859,0.0781)
Bruneau et al. [57]	(0.5313,0.5586)	(0.8711,0.1094)	(0.0859,0.0820)
Vanka [58]	(0.5438,0.5625)	(0.8625,0.1063)	(0.0750,0.0813)
Goyon [59]	(0.5312,0.5625)	(0.8671,0.1171)	(0.0859,0.0781)
Schreiber et al. [60]	(0.5286,0.5643)	(0.8643,0.1071)	(0.0857,0.0714)
Botella et al. [62]	(0.5308,0.5652)	- , -	- , -
Zhang [63]	(0.5313,0.5664)	(0.8633,0.1133)	(0.0820,0.0781)

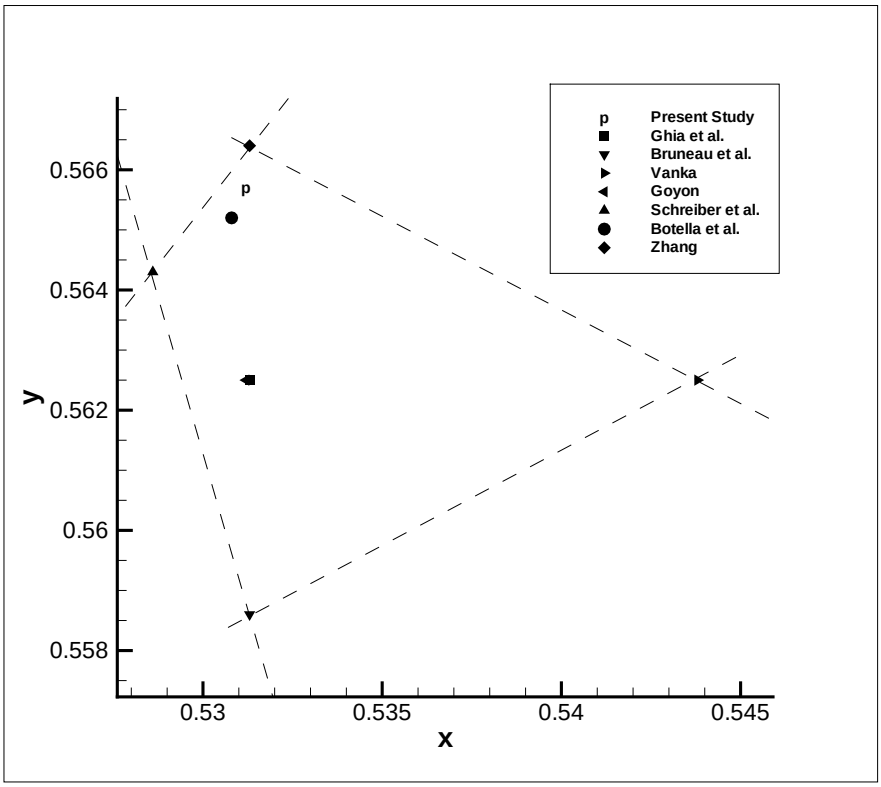

Figure 4.12: Center Location of Primary Vortex, Re = 1000, Problem 1

4.1.3 Algorithm Study

In this subsection the three possible solver algorithms,namely the SIMPLE, SIMPLER and SIMPLEC algorithms, which the program is capable of using, are compared with each other. This set of runs are performed for Re = 100 using the central differencing scheme on a uniform mesh size of 128×128. Another important set of parameters which should be specified at this point is the relaxation parameters which are given in Table 4.6.

For a meaningful comparison, different relaxation parameters should be employed with dif-

Table 4.6: Relaxation Parameters Used in the Algorithm Comparison Runs, Re = 100, Problem 1

	u Relaxation Parameters	v Relaxation Parameters	*p-correction* Relaxation Parameters
SIMPLE	0.8	0.8	0.5
SIMPLEC	0.9	0.9	1
SIMPLER	0.9	0.9	N/A

ferent algorithms, since the main advantage of the SIMPLEC and SIMPLER algorithms over SIMPLE is that these algorithms enable the use of larger relaxation parameters. The relaxation parameters given in the Table 4.6 are the limiting values, any relaxation parameter bigger than the one given in the table, cause divergence for that case. Thus, these parameters can be thought as the optimum values which are appropriate for a comparison purpose. Note that pressure correction term need not to be underrelaxed for SIMPLEC algorithm, so the *p-correction* term, Equation (2.60), is always equal to 1. Whereas, in SIMPLER algorithm a relaxation parameter for pressure correction is not needed at all, so it is not specified.

The residual for mass imbalance term versus the iteration number is given in Figure 4.13. As expected, it is shown that the SIMPLER algorithm performs better than SIMPLE. This is primarly because the SIMPLER algorithm does not require a good pressure guess (which is difficult to provide). Rather, it generates the pressure field from a good guess of velocity field (which is easier to guess) itself. However, the SIMPLER algorithm solves for two pressure variables; the actual pressure and the pressure correction. Thus, this algorithm involves more computational effort per iteration than SIMPLE. The iteration numbers and the corresponding CPU times are given in the Table 4.7. It can be noticed from the table that the additional pressure equation in the SIMPLER algorithm increases the computational effort by about 25% per iteration. Although the computational load per iteration increases, the total computation time decreases about 35% due to the faster convergence rate. The SIMPLEC algorithm also performs better than SIMPLE, however it is not as efficient as SIMPLER. The total reduction in the CPU time is approximately 10% in this case and the computational load per iteration is almost the same as SIMPLE. It should be kept in mind that, although the trend about the iteration numbers and the CPU times are agree with the values in literature, there can be some differences in percentages. Since the aim of this study is to develop a single program that involves many features, it is inevitable to make some inefficient combinations.

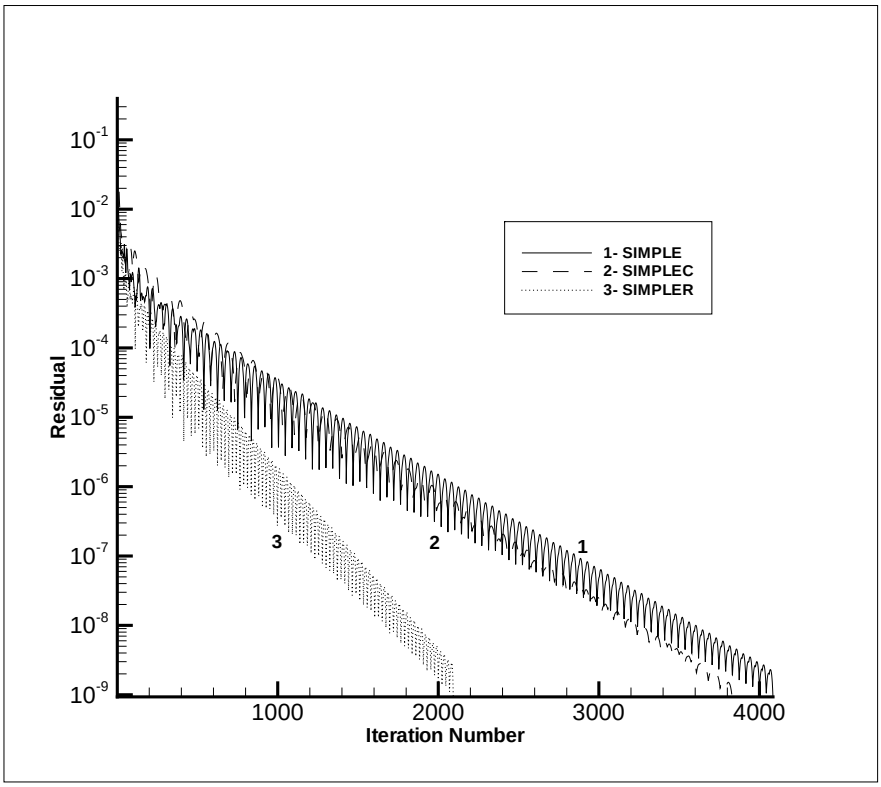

Figure 4.13: Residual for Mass Imbalance Versus Iteration, Re = 100, Problem 1

Table 4.7: Iteration Numbers and the CPU Times, Re = 100, Problem 1

	Convergence Iteration Numbers	CPU Time (s)	Time per Iteration (s)
SIMPLE	4084	197.38	0.0483
SIMPLEC	3828	180.33	0.0471
SIMPLER	2093	126.17	0.0603

4.1.4 Differencing Scheme Study

In this section, the four differencing schemes covered in the previous chapter are compared with each other for different Reynolds numbers. In Figures 4.14 and 4.15 the ability of the schemes to predict the flow at Re 100 are seen. Their extremas and convergence iteration numbers are given in the Table 4.8

The central differencing and hybrid differencing give exactly the same result since for low Reynolds number flows hybrid differencing is equal to the central differencing as mentioned in the second chapter. Also the best prediction of the flow is obtained by these two schemes since central differecing scheme is second order accurate. As expected, first order accurate upwind

differencing scheme has the worst prediction. However, both the extramas and the convergence iteration numbers of different schemes are close to each other. So it can be said that, for low Re number flows the differencing schemes are not very crucial.

The same data that is presented for Re = 100 is now given for Re = 1000 in Figures 4.16 and 4.17 and Table 4.9 . It is clear that as the Reynolds number increases the difference between the schemes becomes more obvious. Again the central differencing scheme offers the best approximation and the upwind differencing scheme offers the worst approximation. Among these four alternative differencing schemes, as long as the central differencing scheme gives converged results it should be the choice, since it is the only second order accurate scheme. However, due to its non-transportive nature, central differencing scheme fails to converge for high Reynolds number flows where the remaining three first order accurate differecing schemes still provide converged results. This is the reason (obtaining accurate results at high Reynold number flows) behind the development of high order upwind differencing schemes like QUICK (Quadratic upwind differecing scheme), but these schemes are not covered in this study.

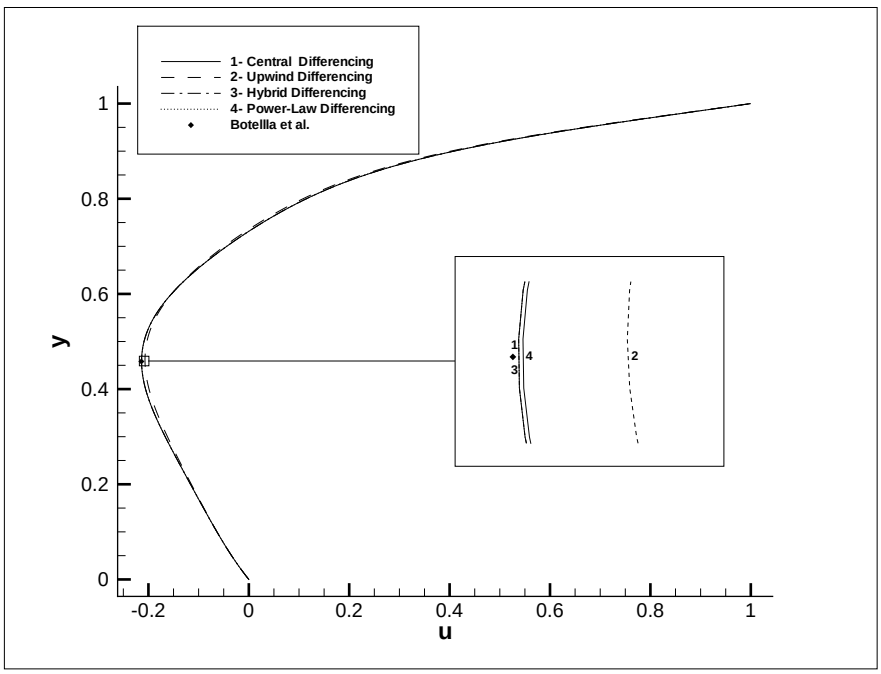

Figure 4.14: u-velocity Along the Vertical Centerline, Re = 100, Problem 1

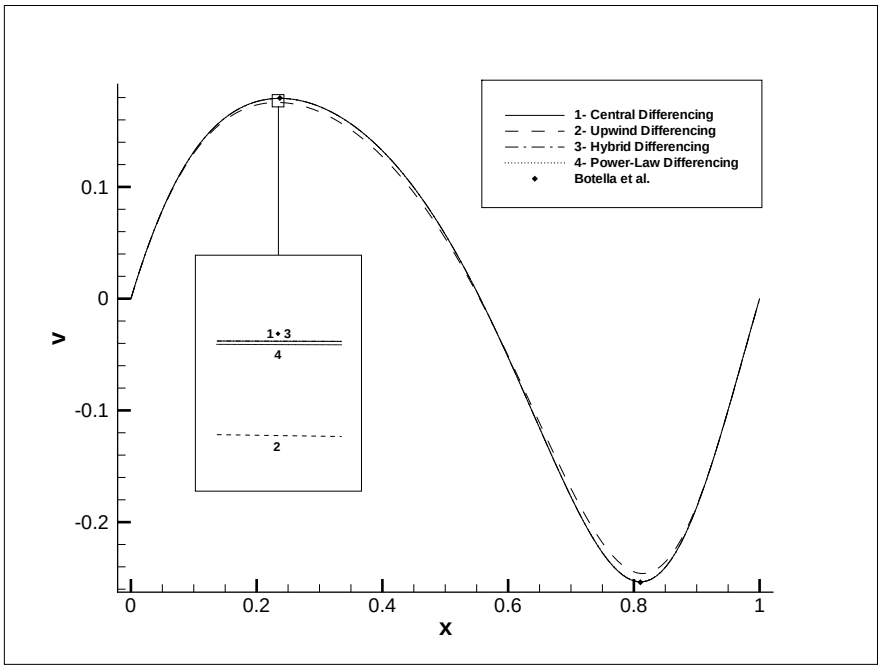

Figure 4.15: v-velocity Along the Horizontal Centerline, Re = 100, Problem 1

Table 4.8: Extrema and Convergence Iteration Number, Re = 100, Problem 1

Differencing Scheme	y_{min}	U_{min}	x_{max}	V_{max}	x_{min}	V_{min}	Convergence Iteration Number
Central	0.4610	-0.2137	0.2344	0.1793	0.8125	-0.2535	4084
Upwind	0.4610	-0.2069	0.2344	0.1756	0.8125	-0.2460	4054
Hybrid	0.4610	-0.2137	0.2344	0.1793	0.8125	-0.2535	4084
Power-Law	0.4610	-0.2134	0.2344	0.1792	0.8125	-0.2532	4000
Botella et al. [62]	0.4581	-0.2140	0.2370	0.1796	0.8104	-0.2538	–

Table 4.9: Extrema and Convergence Iteration Number, Re = 1000, Problem 1

Differencing Scheme	y_{min}	U_{min}	x_{max}	V_{max}	x_{min}	V_{min}	Convergence Iteration Number
Central	0.1719	-0.3824	0.1563	0.3710	0.9063	-0.5181	12020
Upwind	0.1719	-0.3099	0.1563	0.2975	0.9063	-0.4592	11452
Hybrid	0.1719	-0.3726	0.1563	0.3627	0.9063	-0.5095	11698
Power-Law	0.1719	-0.3552	0.1563	0.3449	0.9063	-0.4984	11580
Ghia et al. [56]	0.1719	-0.3829	0.1563	0.3710	0.9063	-0.5155	–

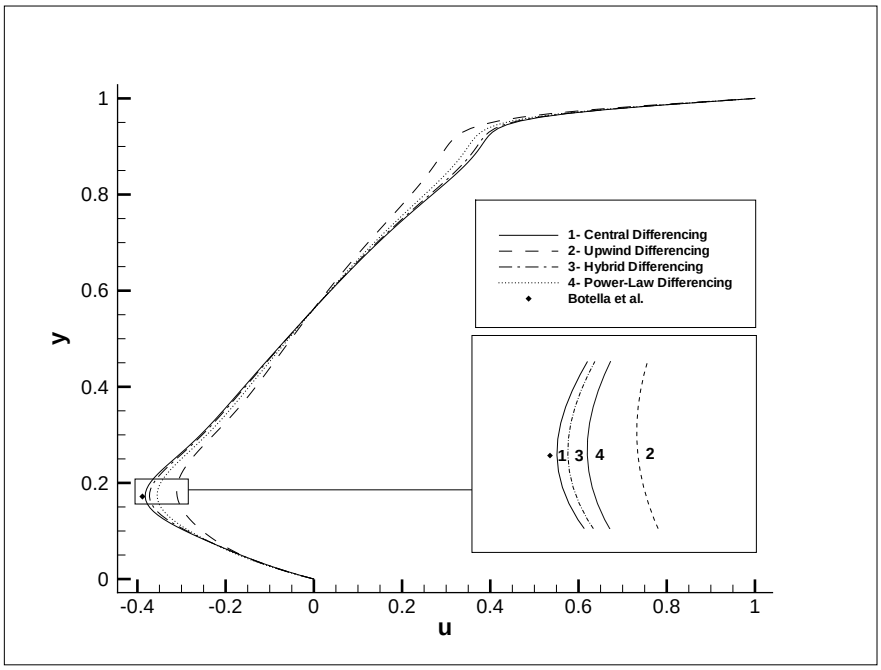

Figure 4.16: u-velocity Along the Vertical Centerline, Re = 1000, Problem 1

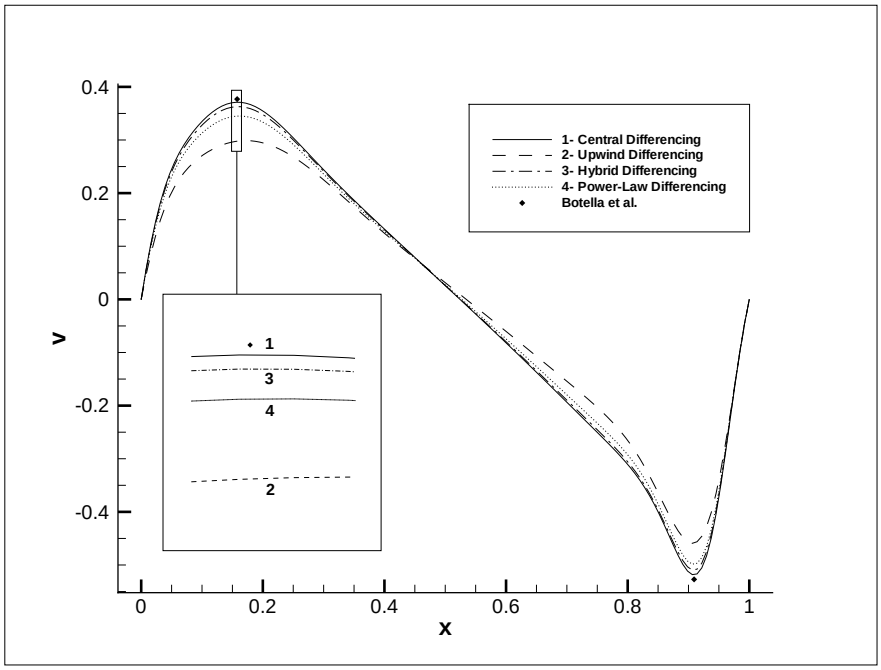

Figure 4.17: v-velocity Along the Horizontal Centerline, Re = 1000, Problem 1

4.1.5 Time Marching Approach

In this section, the superiority of time marching approach over the classical iterative approach will be demonstrated in terms of convergence rate. Similar to the algorithm comparison study, the optimum relaxation parameters and time step sizes are used. Thus, any time step larger than the ones listed in the Table 4.10 will result in divergence. It can be noticed from the table that the algorithms need neither u-relaxation nor v-relaxation parameter, since the relaxation parameters evolve with the solution procedure.

The results show that the time marching approach accelerate the convergence rate in all three algorithms, as shown in Figure 4.18. However, as stated in Table 4.11, the accelaration rate between these algorithms are different. Especially, the SIMPLE algorithm drastically improves.

Table 4.10: Relaxation Parameters and Time Step Size Used in the Time Marhcing Approach Comparison Runs, Re = 100, Problem 1

	u Relaxation Parameters	v Relaxation Parameters	*p-correction* Relaxation Parameters	Time Step Size (s)
SIMPLE	N/A	N/A	0.5	0.0001
SIMPLEC	N/A	N/A	1	0.0002
SIMPLER	N/A	N/A	N/A	0.0002

Table 4.11: Iteration Numbers, CPU Times and Percent Acceleration in the Convergence Rate for Both Iterative and Time Marhcing Approaches, Re = 100, Problem 1

	Convergence Iteration Numbers	CPU Time (s)	Improvement in Convergence Iteration Number (%)	Improvement in CPU Time (%)
SIMPLE Iterative	4084	197.38	-	-
SIMPLE Time Marching	3229	153.38	22.3	20.9
SIMPLEC Iterative	3828	180.33	-	-
SIMPLEC Time Marching	3501	167.56	7.1	8.5
SIMPLER Iterative	2093	126.17	-	-
SIMPLER Time Marching	1885	116.83	7.4	9.9

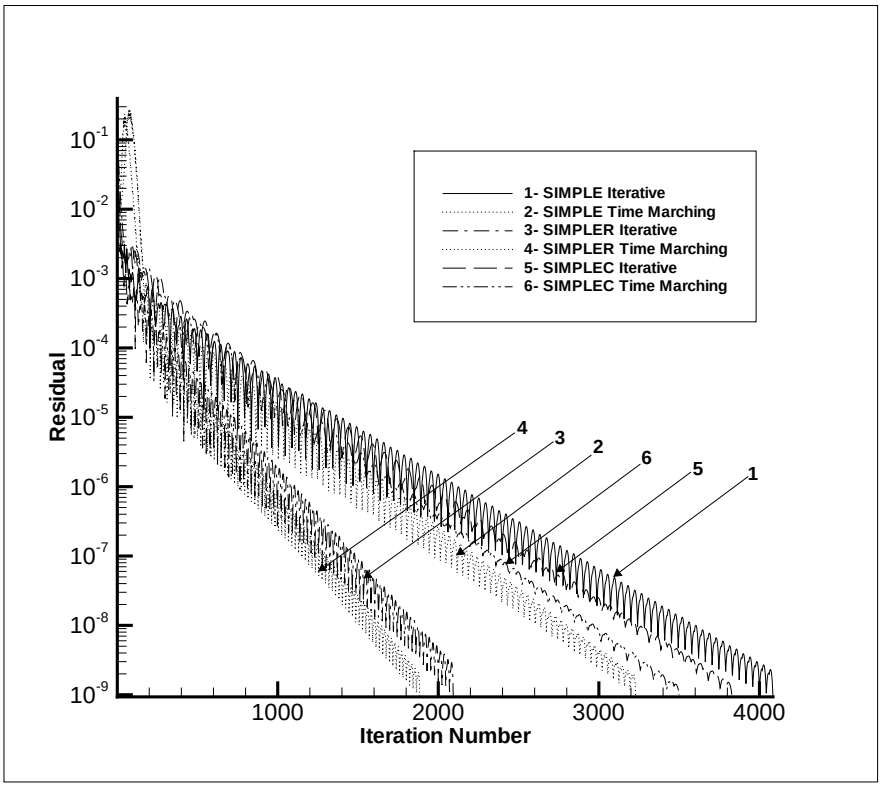

Figure 4.18: Rate of Convergence of Mass Imbalance Term for Different Algorithms With Both Approaches (Iterative and Time Marhcing), Re = 100, Problem 1

4.2 Problem 2. Backward-Facing Step

The separation of the flow and its subsequent reattachement to a solid surface occurs in variety of engineering applications, like airfoils at large angles of attack, channel flows with sudden expansion and turbine blades. However, due to the complexity of the flow physics, the numerical and experimental methods are still far from being perfect for the characterization of the flow for these complex geometries. In order to obtain a deeper undertanding of fluid flow with separation and reattachement, the two-dimensional flow past a backward-facing step is a highly referred test case.

Geometry and the corresponding indexing of the problem is shown in the Figure 4.19. The

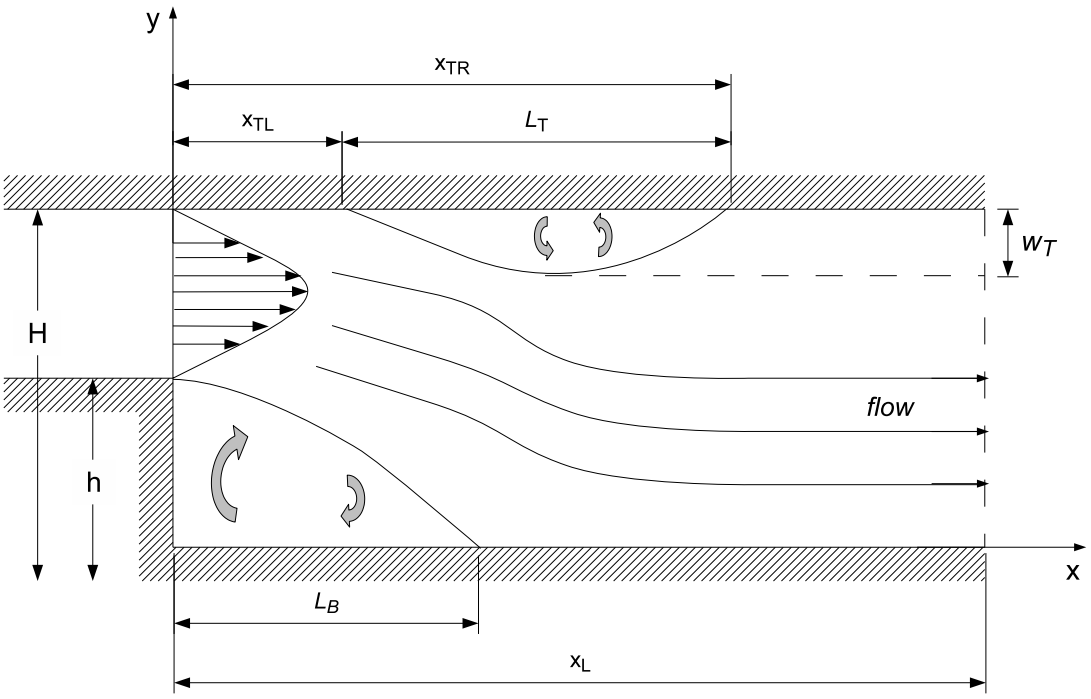

Figure 4.19: Geometry of the Flow Over a Backward-Facing Step in a Channel

expansion ratio of the inlet channel height, $H - h$, to the outlet height, H, is 1:2. The Reynolds number, $Re = U_{avrg}H/\nu$, is based on the mean inlet velocity U_{avrg}, the channel height H, and the kinematic viscosity ν. At the inflow boundary condition a parabolic fully developed profile,

given in Equation (4.1), is used.

$$u = 6U_{avrg}(H - y))(y - h)/(H - h)^2$$
$$v = 0 \tag{4.1}$$

There exists diversity in literature for the proper location of the outflow boundary condition. In all of the studies, the location of the outlet is assumed to be selected far away from the geometrical disturbances where the flow reaches a fully developed state with no further change ocurring in the flow direction. But depending on the method and boundary condition that is used, the "far away" differs from study to study; in some references [64] it is as short as 30h and for some others [65] it is as long as 140h. Also, there exists a diversity in the choice of outflow boundary condition. In some studies the outflow boundary condition is taken to be a parabolic velocity profile [65] as it is in inlet. Although different boundary conditions are used, the most common one is the Neumann boundary condition where the gradient of the flow variables (except pressure) in the direction normal to the outlet surface is taken to be zero. In this case outlet velocities are found by extrapolation, but the order of the extrapolation is also a varying parameter between studies. In some studies [66] even fourth order extrapolation were used where the four velocity positions upstream of the outflow boundary are involved. In this work only first order extrapolation is used in the outlet boundary condition as recommended by Peric [67] which lead quite satisfactory results.

In this set of numerical experiments, the backward-facing step problem is investigated for three different Reynolds numbers which are 300, 600 and 800. Re = 300 is the upper limit of the low Reynolds number flows which behaves like an open backward-facing step flow where the flow seperates at the step and reattaches further downstream. For higher Reynolds number flows (e.g. Re = 600) the adverse pressure gradient is strong enough to cause separation along the upper boundary which reattaches further downstream. Re = 800, the most referred Re number for this benchmark problem, is considered to be the upper limit of steady and stable laminar flows confirmed by Barton [66], Gartling [68] and Gresho et al. [69]. All these analyses are performed on two different mesh densities. The first one, medium mesh size, is composed of

uniformly spaced elements of 80×300. The fine meshed domain consists of as many as 100×500 cells which are clustered in the region where vorticies appear.

All of the results stated in this section are obtained using the following solver parameters: SIMPLE algorithm, central differencing scheme, iterative approach. No additional work has been done to use neither the same nor the optimum relaxation parameters in the analysis. The characteristic values of the backward-facing step flow at Re = 300, Re = 600 and Re = 800 are listed in Tables 4.12, 4.13 and 4.14, respectively.

Table 4.12: Characteristic Values, Re = 300, Problem 2

	Expansion Ratio	L_B	L_T	X_{TL}	X_{TR}
Barton, Central [64]	2	3.570	N/A	N/A	N/A
Barton, Hybrid [64]	2	3.540	N/A	N/A	N/A
Current study, medium	2	3.517	N/A	N/A	N/A
Current study, fine	2	3.565	N/A	N/A	N/A

Table 4.13: Characteristic Values, Re = 600, Problem 2

	Expansion Ratio	L_B	L_T	X_{TL}	X_{TR}
Barton, Central [64]	2	5.360	3.715	4.360	8.075
Barton, Hybrid [64]	2	5.160	3.695	4.130	7.825
Morinava et al. [65]	2	5.370	3.782	4.330	8.112
Current Study, medium	2	5.316	3.539	4.452	7.991
Current Study, fine	2	5.370	3.669	4.477	8.146

The relaxation parameters, convergence iteration numbers and the corresponding computational load of the above runs are given in Table 4.15 for comparison.

It is seen from the above tables that both the locations and the lengths of the separation and reattachement zones which are calculated in this study quite agree with the previous studies. It can be seen that the reattachment length in the below boundary increases almost linearly with Reynolds number, the slight non-linearity is due to the viscous drag along the upper boundary.

Table 4.14: Characteristic Values, Re = 800, Problem 2

	Expansion Ratio	L_B	L_T	X_{TL}	X_{TR}
Marinova [65]	2	6.091	5.651	4.821	10.472
Gartling [68]	2	6.100	5.630	4.850	10.480
Sayma et al. [70]	2	5.605	6.230	4.200	10.430
Srinivasan et al. [71]	2	6.220	5.160	5.090	10.250
Barton [64]	2	5.495	5.500	4.275	9.775
Keskar et al. [72]	2	6.096	5.625	4.853	10.479
Gresho [69]	2	6.100	5.630	4.860	10.490
Armaly et al. [73] (E)	1.94	7.000	5.700	4.300	10.000
Lee et al. [74] (E)	2	6.000	5.500	4.800	10.300
Kim et al. [12]	2	6.000	5.750	-	-
Sohn [75]	2	5.750	4.700	4.700	9.400
Current Study, medium	2	6.022	5.424	4.905	10.330
Current Study, fine	2	6.086	5.555	4.946	10.501

Table 4.15: Relaxation Parameters Used in the Analysis of Problem 2 and the Corresponding Convergence Iteration Numbers and CPU Times.

	u and v- velocity Relaxation Parameters	Pressure Relaxation Parameters	Convergence Iteration Numbers	CPU Time (sec)
Current study, medium, Re = 300	0.4	0.5	13222	973
Current study, medium, Re = 600	0.4	0.5	16237	1181
Current study, medium, Re = 800	0.3	0.3	29076	2123
Current study, fine, Re = 300	0.4	0.5	20728	3461
Current study, fine, Re = 600	0.4	0.5	25996	4064
Current study, fine, Re = 800	0.3	0.3	85601	13427

u and v velocity contours and streamtraces of the current numerical studies are given in Figures 4.20 , 4.21 and 4.22 respectively. It can be noted that as Reynolds number increases both vortices get larger and larger. It is also seen that for Re = 300 case, the confined flow behaves like an open backward-facing step flow where the flow seperates at the step and reattaches further downstream.

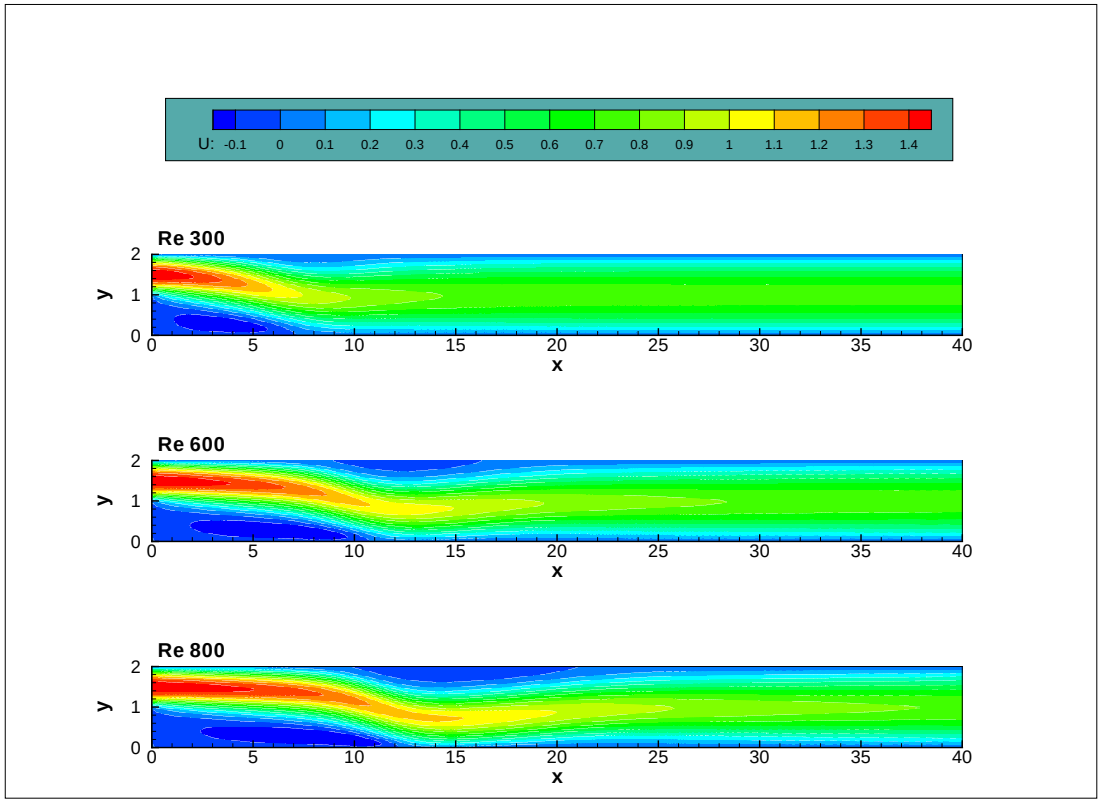

Figure 4.20: u-velocity Contours of Problem 2

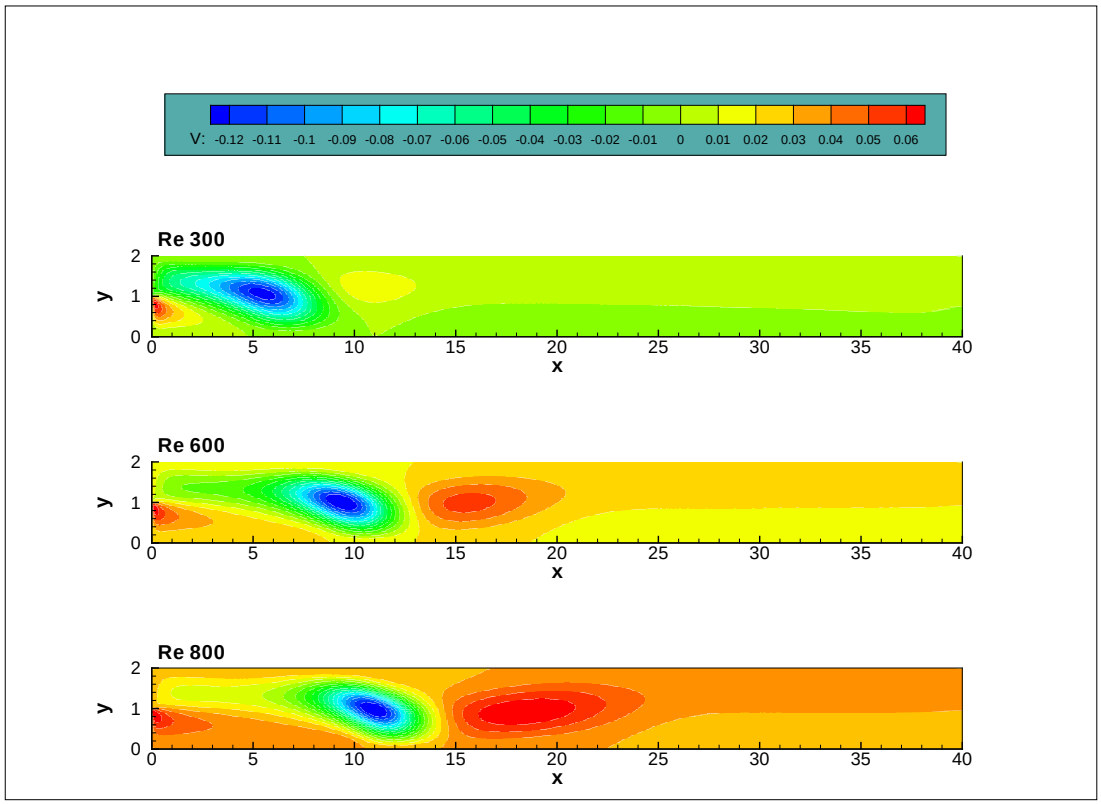

Figure 4.21: v-velocity Contours of Problem 2

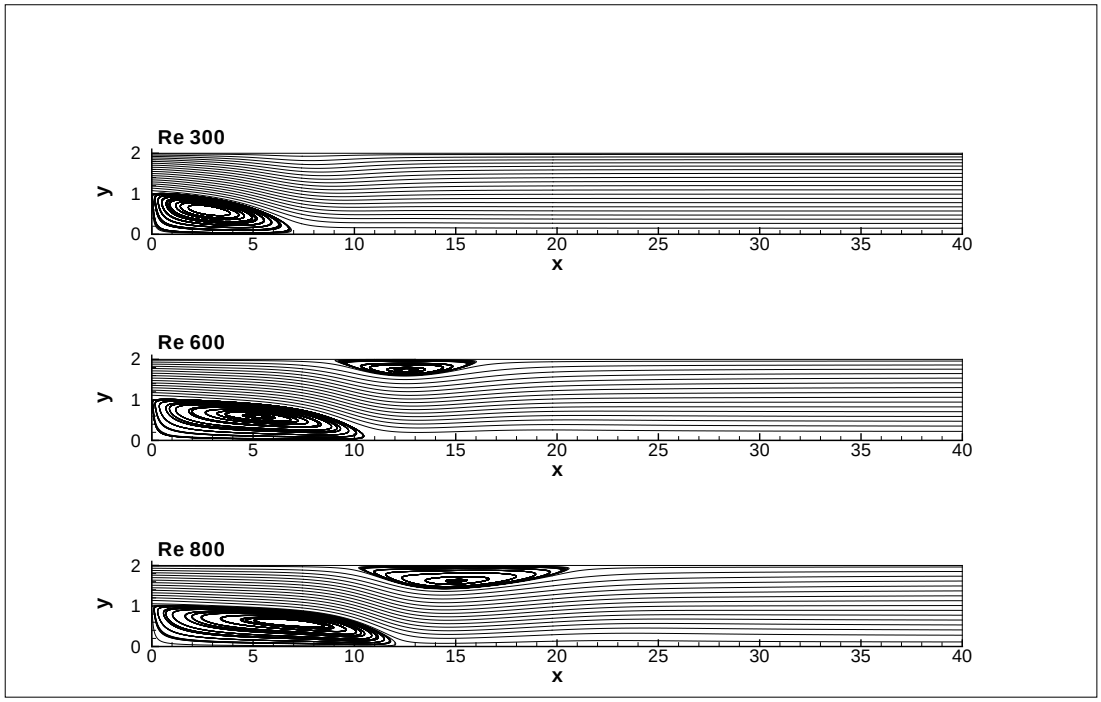

Figure 4.22: Streamtraces for Problem 2

4.3 Problem 3. Confined Flow Past a Rectangular Cylinder

The unsteady flows around bluff bodies such as circular and rectangular cylinders [1] and flat plates are of direct relevence to many practical applications (e.g. road vehicles, heat exchangers, chimneys, suspension bridges) and receiving a great deal of attention due to its importance as a design parameter for both energy efficient and structurally reliable systems. Other than its practical importance, this test case enables us to check the two capabilities (time-dependent solution and blocked-cell method) of the solver that could not be controlled with the previous benchmark problems.

In this set of numerical experiments the laminar two dimensional viscous flow past a square cylinder is investigated for various Reynolds numbers. The problem definition is given in Figure 4.23.

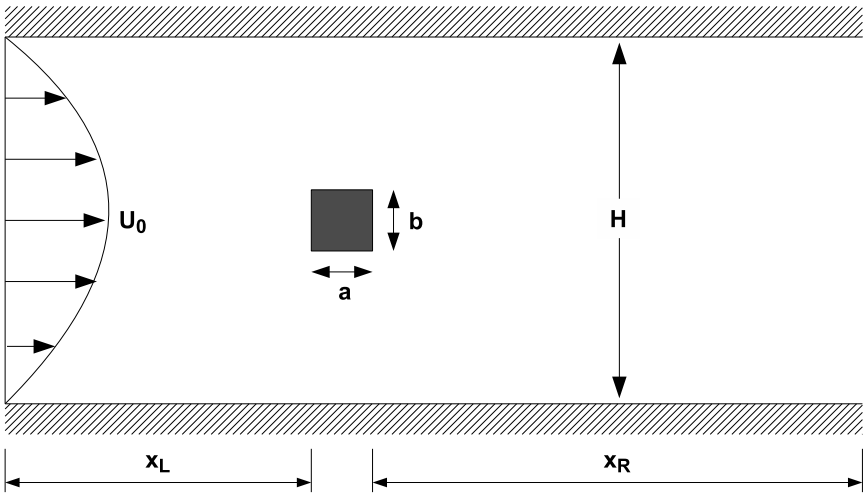

Figure 4.23: Configuration Definition of Flow Past a Square Cylinder

The Reynolds number for this flow is defined as $Re = U_0 b/\nu$. In the analyses, all the lengths are nondimensionalised by height of the square cylinder (b) and all the velocities by velocity magnitude at the centerline of the channel inlet (U_0), respectively. Thus, in Figure 4.23 $b = U_0 = 1$ by definition. Then "a" becomes the aspect ratio (AR) of the rectangle, and "H^{-1}" becomes the blockage ratio (BR) . The non-dimensional geometric parameters used in the following analysis

[1] In this context, which is 2D, cylinder refers to section. Thus, rectangular cylinder means a 2D rectangular section.

are, $BR = 1$, $AR = 1$, $x_L = 7$, $x_R = 18$.

The surface boundary conditions used in this study are that normal and tangential velocities are zero along the channel walls. A parabolic (fully-developed) inlet velocity profile is employed at $x = 0$. Finally, for the outflow the Neumann boundary condition is used where the gradient of the flow variables (except pressure) in the direction normal to the outlet surface is taken to be zero.

The numerical solution is accomplished on a variably spaced staggered mesh size of 300×120, as shown in Figure 4.24. The variable spacing is such that the mesh cells are concentrated in the areas around the rectangle where zones of separation and vorticies are distinct.

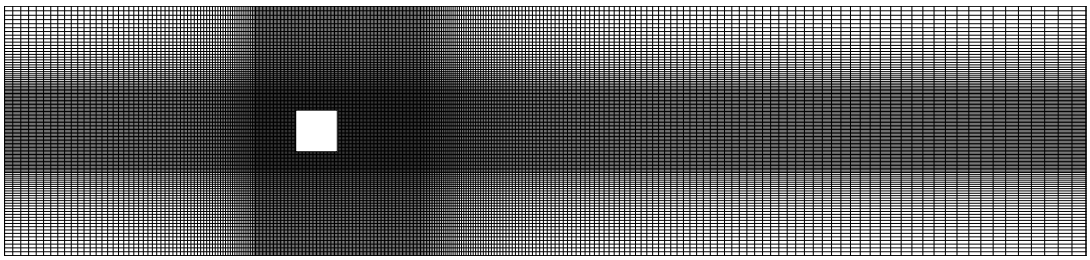

Figure 4.24: The 300×120 Non-Uniform Grid, Problem 3

For all the analyses listed in this section the SIMPLE algorithm and first order upwind differencing are employed. As the Reynolds number increases, the flow direction becomes more and more important. Thus, in this set of runs the upwind differencing is used due to its tranportiveness property. Also pressure relaxation parameter of 0.5 and time step size of 2.5×10^{-5} are used in the analyses. The analyses are performed for Reynolds numbers of 100, 150, 200, 250 and 300. The Strouhal numbers are calculated using

$$St = \frac{fb}{U_0} \tag{4.2}$$

where f is the shedding frequency. To determine the shedding frequency a control point, which is $1.5a$ behind the center of the square, is located. Then, using the data gathered from this point, the velocity profiles are drawn and the periodicity is figured out which leads to shedding

frequency. In Table 4.16, the Reynolds numbers and corresponding Strouhal numbers for the test cases are given. As given in Figure 4.25, the results are in agreement with the previous studies of many researchers, Breuer et al. [76], Galletti et al. [77], Mukhopadhyay [78], Roy et al. [79] and Davis et al. [80].

Table 4.16: Strouhal Numbers for Different Reynolds Numbers, Problem 3

	Strouhal Number
Current study, Re 100	0.1285
Current study, Re 150	0.1380
Current study, Re 200	0.1386
Current study, Re 250	0.1371
Current study, Re 300	0.1348

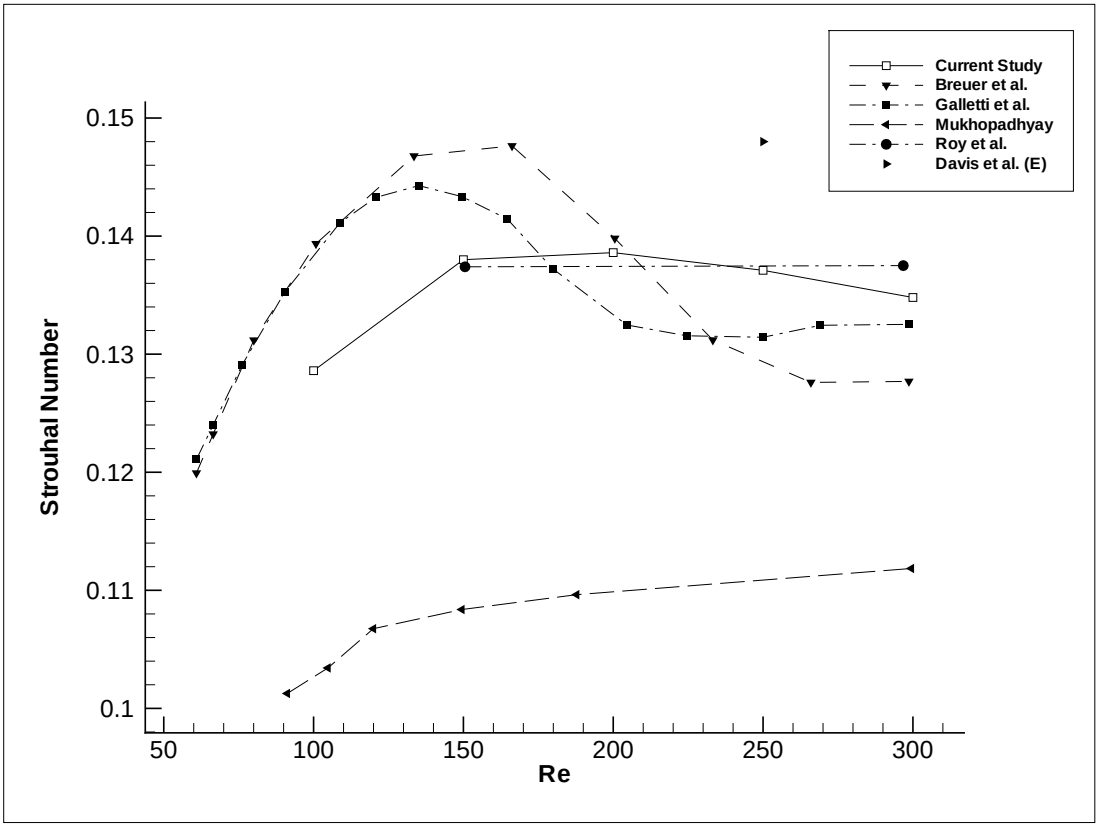

Figure 4.25: Comparison of Strouhal Numbers With Various Reynolds numbers, Problem 3

The instantaneous streamlines in Figure 4.26 (a-h) show the detailed views of the laminar vortex shedding near the square cylinder at Re 200, for eight successive moments of time which span

over the whole period (i.e., Figure 4.26 (a) is repeated after Figure 4.26 (h) for the next cycle of vortex shedding). It is quite easy to follow the *Karman vortex street* that is formed behind the cylinder. The critical points of the streamline patterns, e.g. centers and saddles, described by Perry et al. [81] are also shown in the figure. Similar to the study of Eaton [82] on circular cylinders, it is observed in Figure 4.26 (e-h) that while a vortex is shedding from the top of the cylinder, fluid from the below of the cylinder is drawn up into the recirculation region. The vortex that is forming on the top of the rear face of the cylinder grows until it reaches the point where it breaks off, as shown in Figure 4.26 (g). Then the shedding process is repeated from the other side. Another distinguishing feature of the streamline patterns which is the instantaneous *alleyways* as called by Perry are also observed. The alleyway is a path along which fluid is drawn from above or below the cylinder into the recirculating region. These appear in Figures 4.26 (e) and 4.26 (h), for instance, where an instantaneous alleyway carries fluid from below the cylinder around the forming vortex and up between the shedding vortex and the cylinder to the top of the recirculation region. The results shown in Figure 4.26 conform to Sharma et al's [83] picture of laminar vortex shedding. Also the instantaneous u- & v-velocity, pressure and vorticity contours are presented in Figure 4.27. As expected u and v velocities are oscillating due to vortex shedding. It can also be noted from the vorticity contour plot that direction of vortices on the top and bottom sides are opposite.

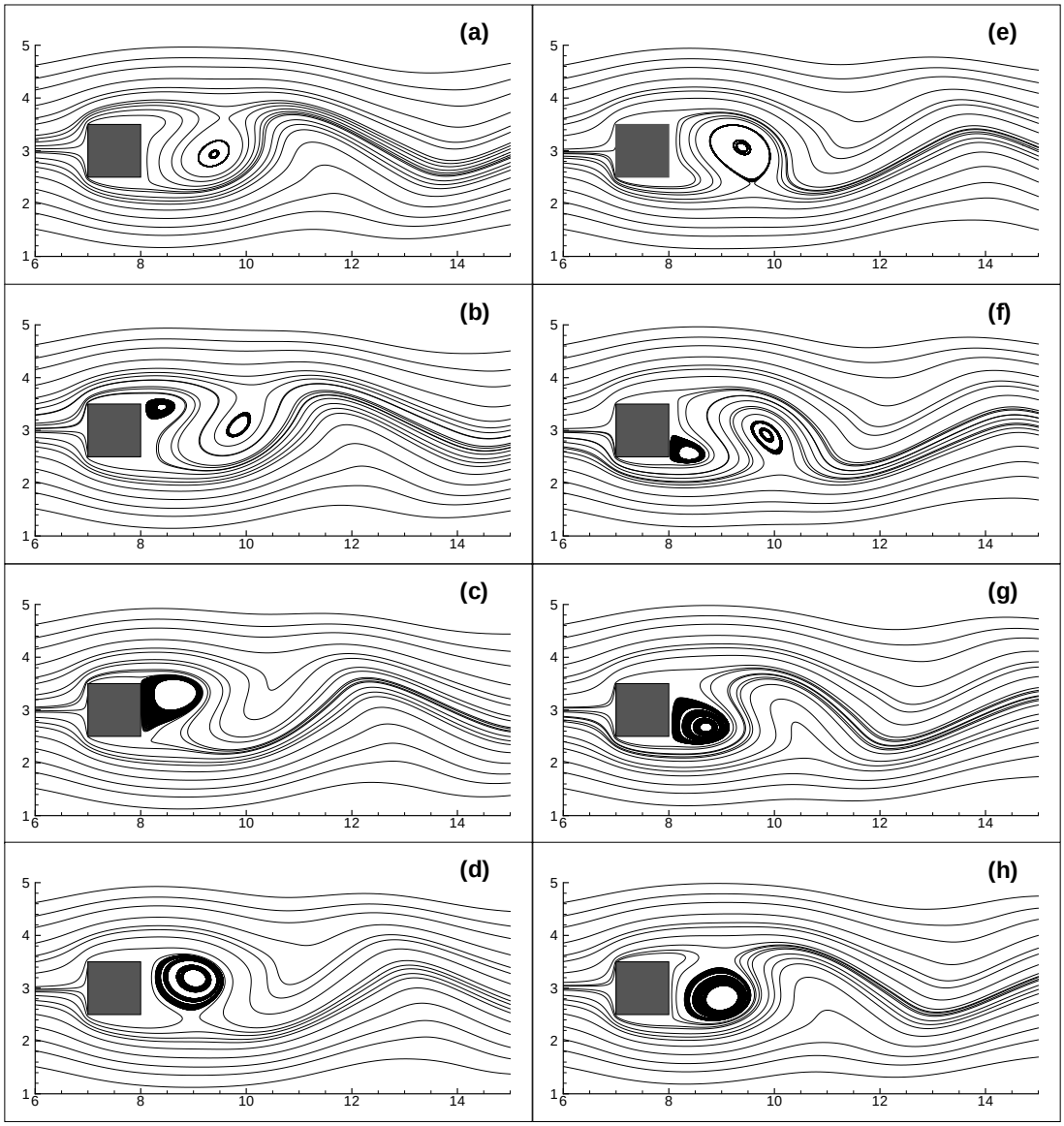

Figure 4.26: Instantaneous Streamlines Near the Square Cylinder, Separated by an Interval of One-Eight of the Time Period of Vortex Shedding (≈ 0.0045 sec), Re = 200, Problem 3

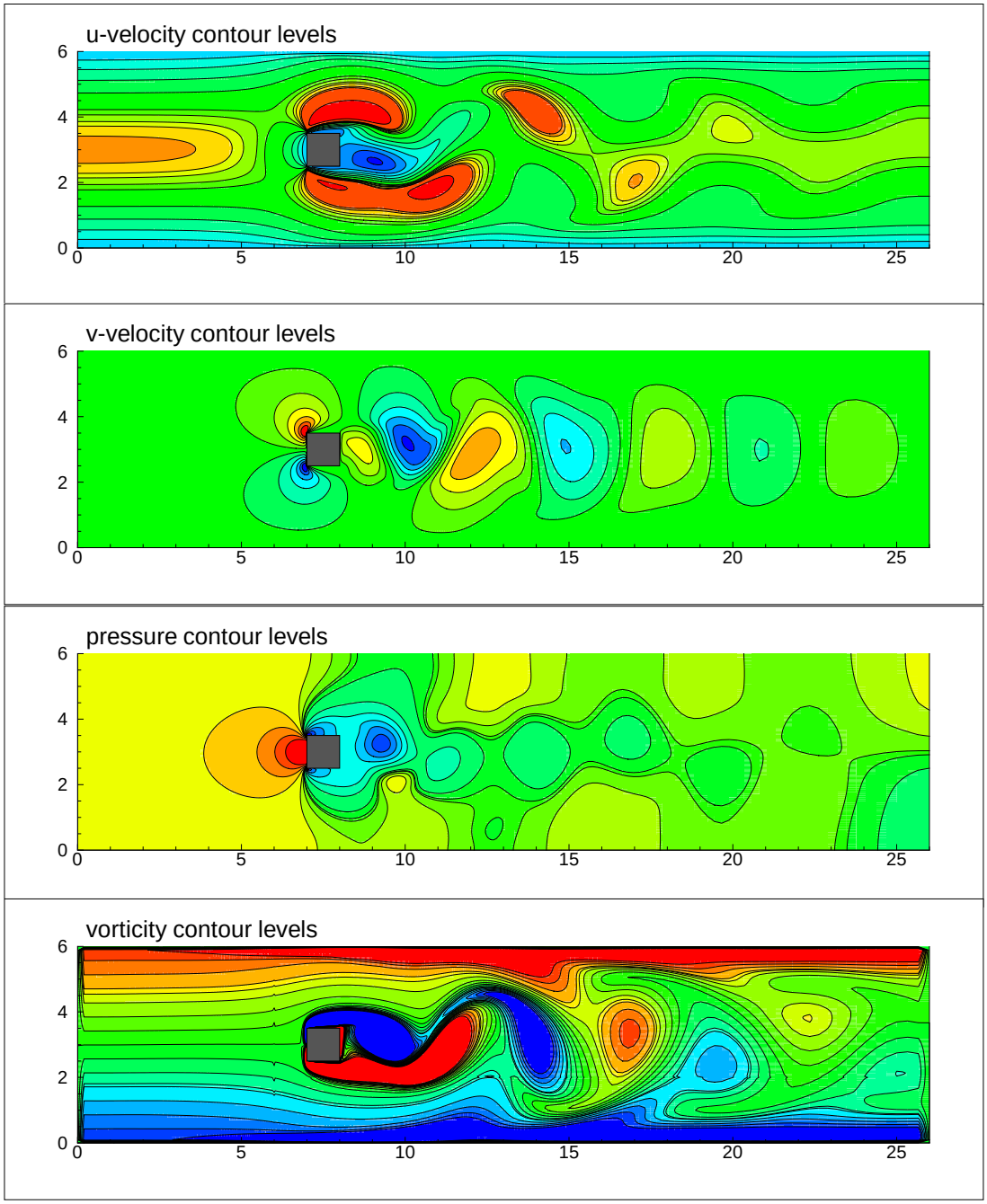

Figure 4.27: Contours of u and v-velocity, Pressure and Vorticity for Blockage Ratio of 1/6, Re = 300, Problem 3

4.4 Problem 4. Labyrinth Flow

In this sub-section the solver is tested with a more complex problem where many sharp turns exist in the flow field. The definition of this problem, which is called as Labyrinth flow in this study, is given in Figure 4.28. This geometry has direct relevance to many practical applications where the extended surface area is critical, like electronic cooling and heat exchangers. For the

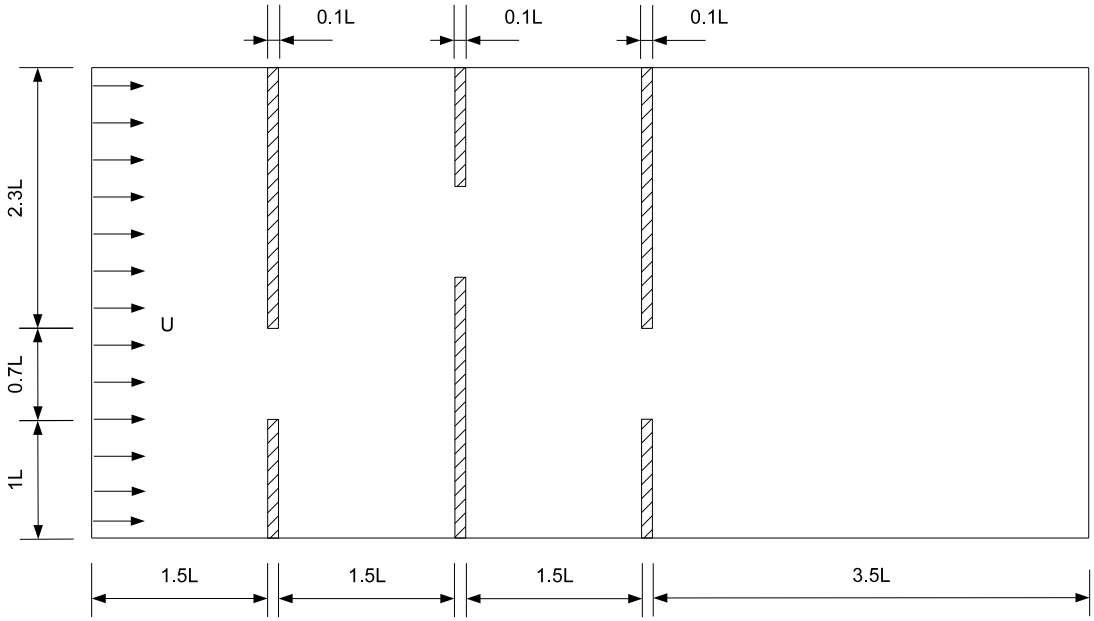

Figure 4.28: Definition of Problem 4

numerical solution, a uniformly spaced mesh size of 166×80 is employed, as shown in Figure 4.29. It is essential at this point to re-emphasize that the block-off cell method is used to form the walls of the Labyrinth. The surface boundary conditions, used in this case are that the normal and tangential velocities are zero along the channel wall. A uniform velocity profile is given as the inlet boundary condition and for the outlet, the Neumann boundary condition is specified where the gradients of velocities in the direction normal to the outlet surface is taken to be zero.

The problem is solved with both the solver developed in this research and with FLUENT. The parameters that are employed in both analyses with the corresponding convergence iteration numbers and CPU times are listed in the Table 4.17. The mass source term is used as the convergence criteria for both analyses where the mass imbalance has to be below the limit of

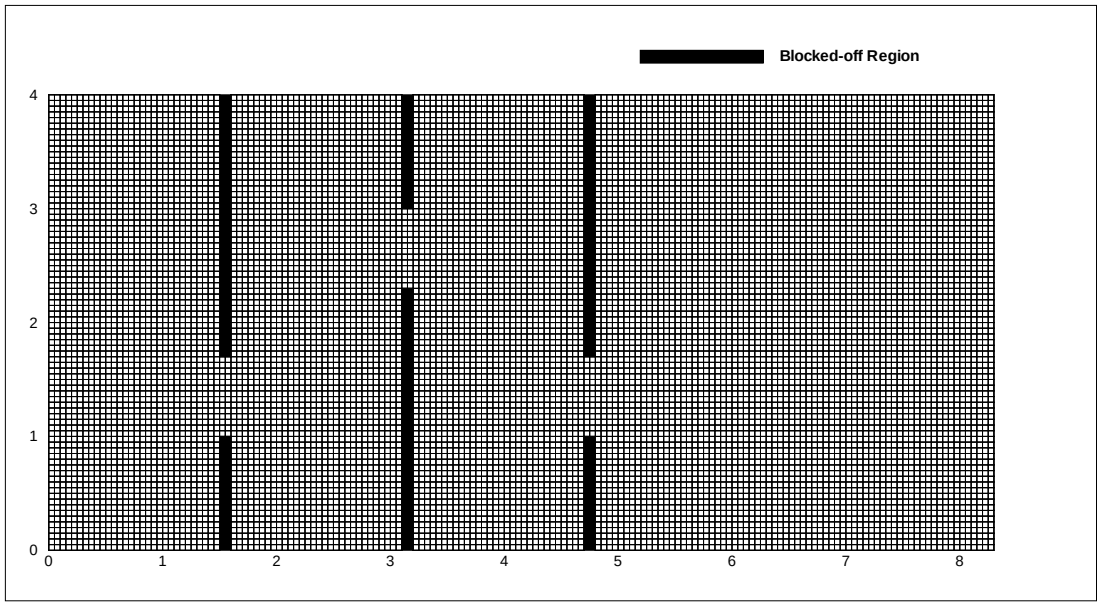

Figure 4.29: 166×80 Mesh is Used in Problem 4

10^{-9} in all of the computational cells.

Table 4.17: Solver Parameters Used in Problem 4 and Corresponding Results

	Algorithm	Differencing Scheme	Pressure Relaxation	Velocity Relaxation	Convergence Iteration Number	CPU Time (sec)
Current Study	SIMPLE	Central	0.3	0.7	2077	85.0
FLUENT	SIMPLE	2^{nd} Upwind	0.3	0.7	2862	263.2

Due to the geometry of the problem, in many locations both high gradients and recirculation regions are expected. The u- & v-velocity and pressure contours are given in Figure 4.30. As expected, both u and v components of the velocity increases in the narrow passages. The velocity magnitude contour and the comparison of the velocity magnitude contour with FLUENT result are given in Figure 4.31 and 4.32 respectively. The streamtraces of the problem are given in Figure 4.33, it can be noted that vortices appear in every turn of the flow field.

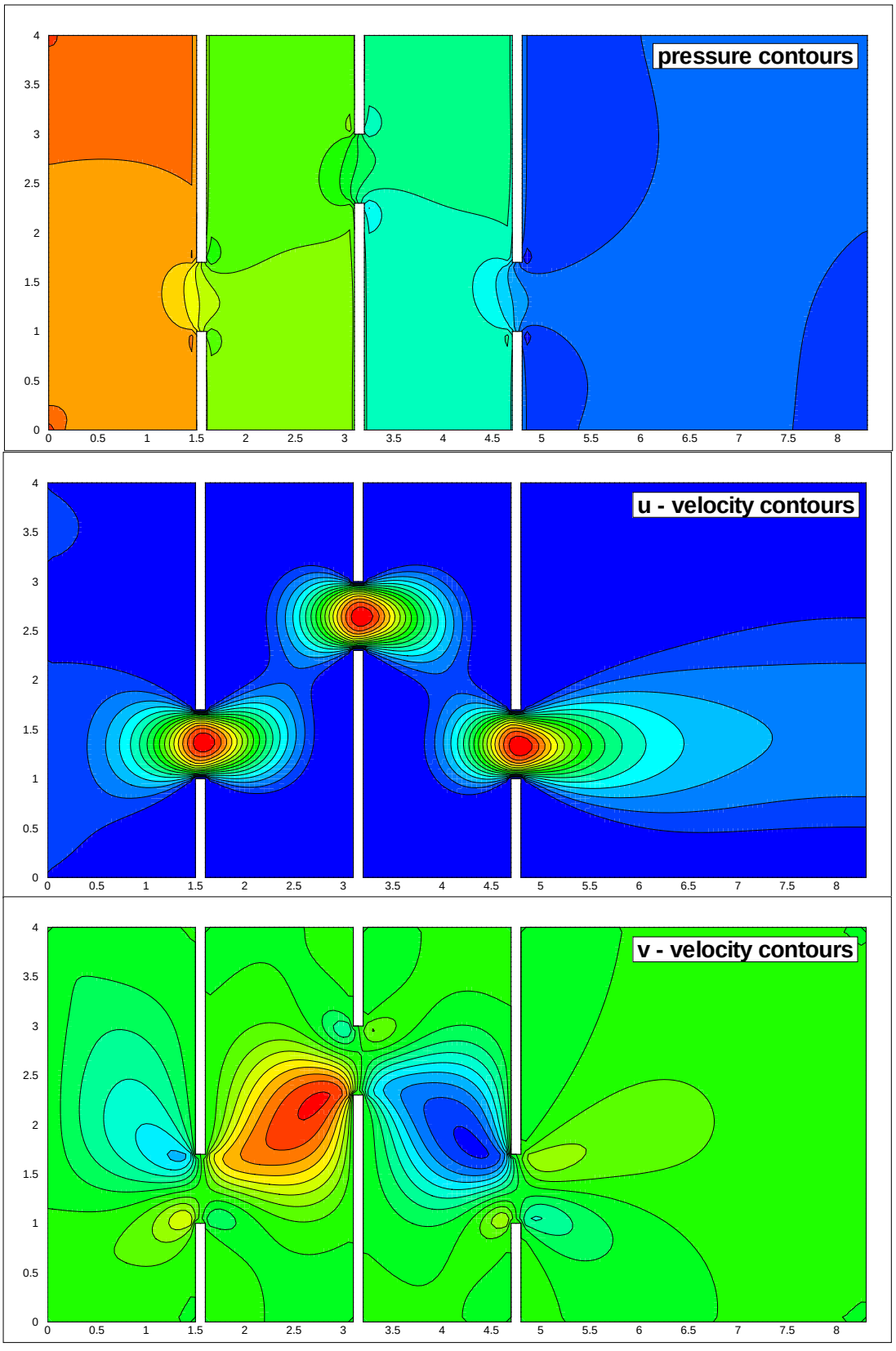

Figure 4.30: u-velocity, v-velocity and Pressure Contour of Problem 4

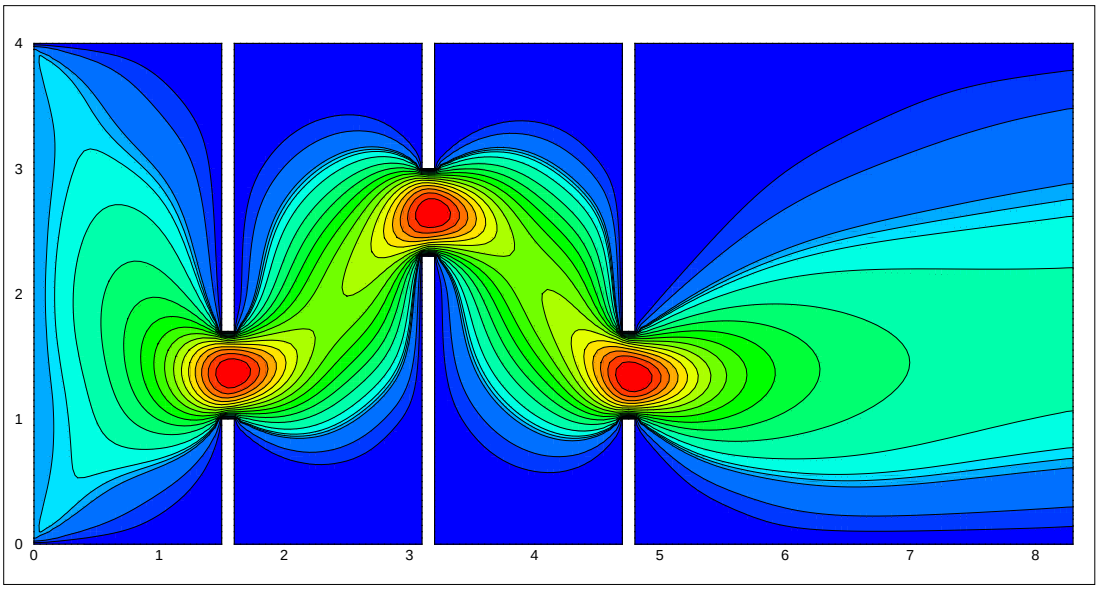

Figure 4.31: Velocity Magnitude Contours of Problem 4

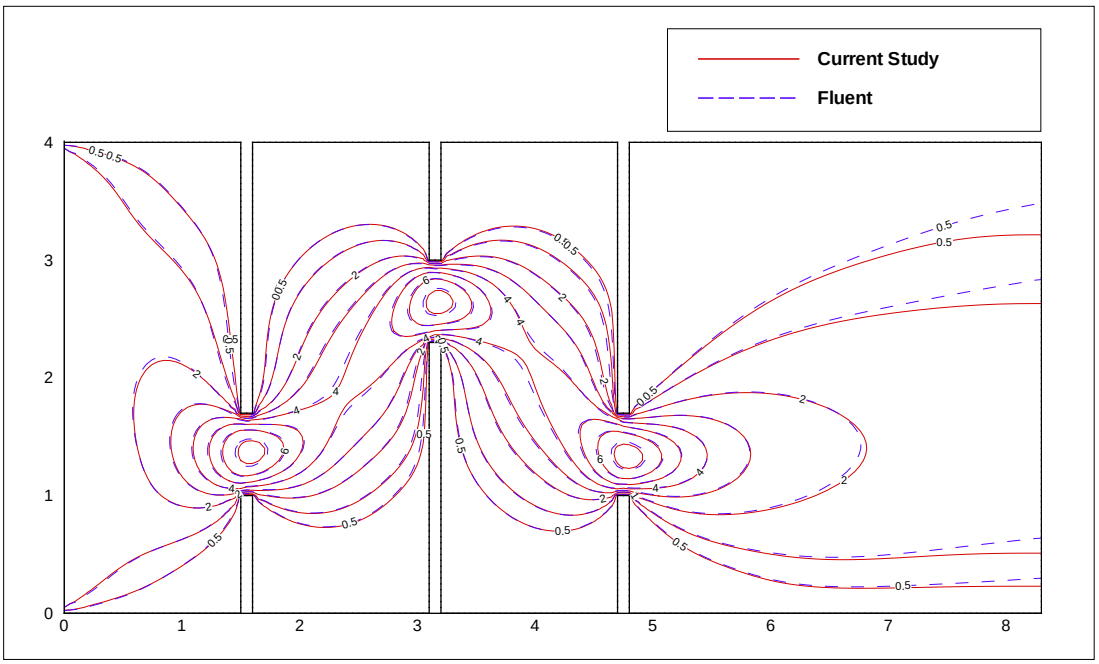

Figure 4.32: The Comparison of Velocity Magnitude Contours With FLUENT Result, Problem 4

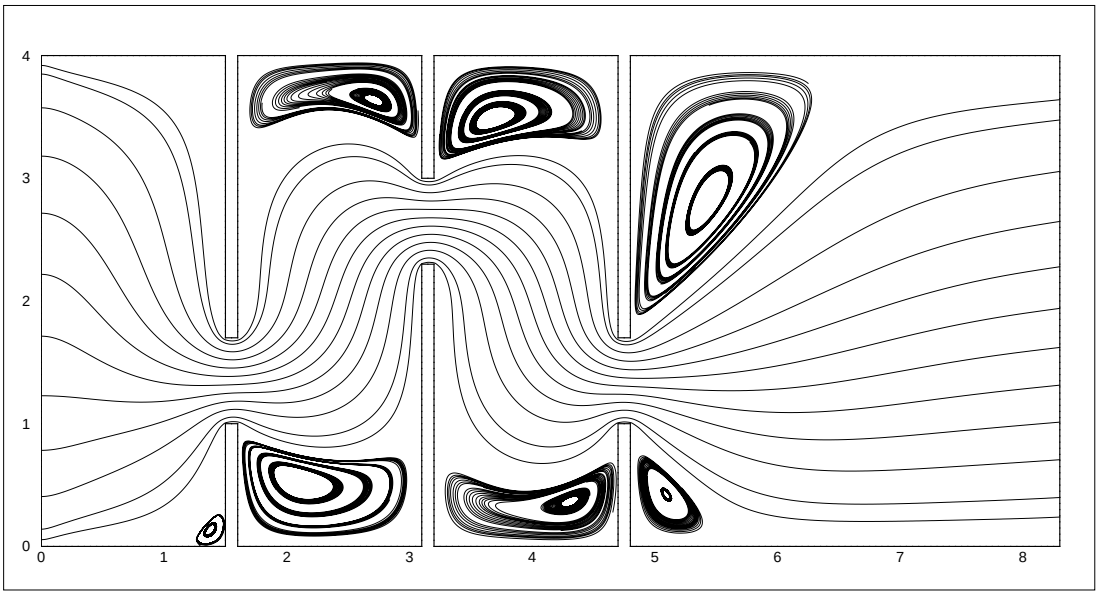

Figure 4.33: The Streamtraces of Problem 4

It is obvious from the Figure 4.32 that the result of the current study is in agreement with the results of FLUENT. The only deviation between the results appear close to the outlet section and it is due to the difference between the boundary conditions that are applied. In the FLUENT analysis, the zero gage pressure is employed as the outlet boundary condition whereas in the current solver, Neumann boundary condition is used. It is known that the gradients of the velocities in the direction normal to the outlet boundary have to be zero in Neumann boundary condition means the velocity magnitude contour are forced to take a horizontal shape at the exit plane. Although, the type of the outlet boundary condition has certain effects on the solution, it is essential to note that, this effect is limited to the region close to the outlet boundary.

CHAPTER 5

CONCLUSIONS

This study introduced a new Computational Fluid Dynamics software called Virtual Flow Lab (VFL). The main motivation behind developing VFL was to have an easy to learn and use, economical CFD software that can be used for educational purposes. Although VFL project is still in progress, current working version of it, as presented in this thesis, shows a great potential for fluid mechanics and CFD education. VFL's robust flow solver is also expected to serve as a starting point for future thermo-fluidic related researches. Main features of the current version of VFL are listed below

- It has a graphical user interface that enables basic pre-processing, solver parameter and boundary condition setting and post-processing steps of a typical CFD simulation. [1]

- It can generate multi-block, structured, orthogonal or non-orthogonal grids using a number of different algebraic and differential mesh generation techniques. [1]

- It can be used to study incompressible, laminar, steady or time-dependent problems on two-dimensional Cartesian grids.

- Its pressure based solver incorporates with the SIMPLE algorithm and its two variants, SIMPLER and SIMPLEC.

- Space discretization can be done using one of the central, upwind, power law or hybrid differencing schemes.

[1] These features are developed in another research study [52] and combined with the current one.

- Steady and time-dependent problems can be solved using either an iterative of time-marching approach.

- It implements the so called blocked-cell technique to extend the types of problems that can be studied on a Cartesian grid.

- It supports stationary or sliding wall, inflow and outflow type boundary conditions.

- Its post-processing capabilities include drawing contour plots, placing streamlines and probing the flow field. [1]

Virtual Flow Lab project is a work in progress and currently improvements are being made in the following areas

- Switching from staggered grid arrangement to colocated arrangement.

- Improving the solver to study problems on non-orthogonal grids.

- Modifying the pressure-based algorithm to have an all-speed flow solver that will enable the solution of both incompressible and compressible flows.

Future capabilities that VFL is planned to have in the future include

- An improved data structure to cover multi-block structured grids.

- Cutting cell detection algorithm to simulate problems with curved boundaries more accurately on Cartesian grids.

- Automatic grid refinement on structured grids with hanging nodes.

- A new solver for unstructured grids.

- A complete set of pre-prepared problem templates and help files to enhance the educational aspect of the project.

REFERENCES

[1] Kopal, Z., "*Tables of Supersonic Flow Around Cones*", Dept of Electrical Engineering, Center of Analysis, Massachusetts Institute of Technology, Cambridge, 1947.

[2] Wendt, J. F., "*Computational Fluid Dynamics an Introduction*", 2nd edition, Springer, 1995.

[3] Ozisik, M. N., "*Finite Difference Methods in Heat Transfer*", CRC Press, 1994.

[4] Clough, R.W., "*The Finite Element Analysis in Plane Stress Analysis*", Proc. 2nf ASCE Conf. on Electronic Computation, Pittsburg, PA, September. (Poland Lewis), 1960.

[5] Zienkiewicz, O. C., Cheung, K., "*Finite Elements in the Solution of Field Problems*",Engineer, vol. 200, pp. 507-510, 1965.

[6] Patankar, S. V., "*Numerical Heat Transfer and Fluid Flow*",Series in Computational Methods in Mechanics and Thermal Sciences, McGraw-Hill, New York, 1980.

[7] Harlow, F. H., Welch, J. E., "*Numerical Calculation of Time Dependent Viscous Incompressible Flow with Free Surface*",Pyhsics of Fluids, vol. 8, no. 12,pp.2182-2189, 1965.

[8] Roach,P. J., "*Computational Fluid Dynamics*", Hermosa Publishers, Albuquerque,New Mexico, 1972.

[9] Ferziger, J. E., "*Incompressible Turbulent Flows*", Journal of Computational Physics, vol. 69, pp. 1-48, 1987.

[10] Kwak, D., "*Computation of Viscous Incompressible Flows*" von Karman Institute For Fluid Dynamics,Lecture Series 1989-04, 1989.

[11] Chorin, A. J., "*A Numerical Solution of Navier Stokes Equations*", Mathematical Computation, vol. 22, no. 104, pp. 745-762, 1968.

[12] Kim, J., Moin, P., "*Application of a Fractional Step Method to Incompressible Navier-Stokes Equations*", Journal of Computational Physics, vol. 59, pp. 308-323, 1985.

[13] Patankar, S. V., Spalding, D. B. "*A Calculation Procedure for Heat, Mass and Momentum Transfer in Three Dimensional Parabolic Flows*",International Journal of Heat and Mass Transfer, vol.15, pp.1787-1806, 1972.

[14] Van Doormal, J. P.,Raithby, G. D., "*Enhancements of the SIMPLE Method for Predicting Incompressible Fluid Flows*", Numerical Heat Transfer, vol. 7, pp. 147-163, 1984.

[15] Chorin, A. J., "*A Numerical Method for Solving Incompressible Viscous Flow Problems*", Journal of Computational Physics, Vol. 2, pp. 12-26, 1967.

[16] Peyret, R., "*Unsteady Evolution of Horizontal Jet in a Stratified Fluid*", Journal of Fluid Mechanics, vol. 78, pp. 49-63, 1976

[17] Rogers, S. E., Kwak, D., "*A Upwind Differencing Scheme for the Time Accurate Incompressible Navier Stokes Equations*", AIAA Paper 88-2583, Williamsburg, Virginia, 1988.

[18] Chen, K. H., Pletcher, R. H., "*Primitive Variable, Strongly Implicit Calculation Procedure vor Viscous Flows at All Speeds*", AIAA Journal, vol. 29, pp. 1241-1249, 1991.

[19] Chang, K. S., Sa, J.Y., "*Patterns of Vortex Shedding from an Oscillating Circular Cylinder*", AIAA Journal, vol. 30, pp.1331-1336, 1992.

[20] Justesen, P., "*A Numerical Study of Oscillating flow around a circular cylinder*", Journal of Fluid Mechanics, vol. 222, pp.157-196, 1991.

[21] Hafez, M.,Dacles, J. and Soliman, M. "*A Velocity Vorticity Method for Viscous Incompressible Flow Calculations*",11th Int. Conference on Numerical Methodss in Fluid Dynamics, Williamsburg, Virginia, 1988.

[22] Dacles, J. S., "*Numerical solution of Incompressible Navier-Stokes Equations Using a Velocity-Vorticity Formulation*",Doctoral Dissertation, University of California,Davis, California, 1991.

[23] Turkel, E., "*Preconditioned Method for Solvig the Incompressible and Low Speed Compressible Equations*", Journal of Computational Physics, vol. 72, pp. 277-298, 1987.

[24] Rubel, A., Volpe, G., "*Biharmonic Vector Stream Function Formulation and Multigrid Solution for a Three Dimensional Driven Cavity Stokes Flow*", AIAA Paper 89-1968-CP, 1989.

[25] P. Tamamidis, G. Zhang and D. N. Assanis, "*Comparison of Pressure-Based and Artificial Compressibility Methods for Solving 3D Steady Incompressible Viscous Flows*", Journal of Computational Physics, vol. 124, pp. 1-13, 1996.

[26] K. Karki, S.V. Patankar, "*Pressure Based Calculation Procedure for Viscous Flows at all Speeds in Arbitrary Configurations*", AIAA Journal, vol. 27, No 9, pp. 1167-1174, 1989.

[27] Hirsch, C., "*Numerical Computation of Internal and External Flows, vol.I. Fundamentals of Numerical Discretization*", Wiley Interscience Publication, 1988.

[28] Ferziger, J.H., Peric, M., "*Computational Methods for Fluid Dynamics*", third, rev. edition, Springer, 2002.

[29] Hoffmann, K.A., Chiang, S.T., "*Computational Fluid Dynamics, Volume I*",fourth edition, Engineering Education Systems, 2000.

[30] Stone, H.L., "*Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations*" ,SIAM Journal of Numerical Analysis, vol. 5, pp. 530-558, 1968.

[31] Hageman, L.A.,Young, D.M., "*Applied Iterative Methods*" ,Wiley, New York, 1981.

[32] Poussin, F.V., "*An Accelerated Relaxation Algorithm for Iterative Solution of Elliptic Equations*" SIAM Journal of Numerical Analysis, vol. 5, pp. 340-351, 1968.

[33] Brant, A., "*Multi Level Adaptive Solutions for Boundary Value Problems*" Mathematical Computation, vol. 31, pp. 333-390, 1977.

[34] Sivaloganathan, S.,Shaw, G.J., "*An Efficient Non-linear Multigrid Procedure for the Incompressible Navier Stokes Equations*", International Journal of Numerical Methods in Fluids, vol. 8, pp. 417-440, 1988.

[35] Becker, C., Ferziger, J. H., Peric, M. and Scheurer, G., "*Finite Volume Multigrid Solutions of the Two-Dimensional Incompressible Navier Stokes Equations*" Notes on Numerical Fluid Mechanics, vol. 23, pp. 37-47, 1988.

[36] Smith, K. M., Cope, W. K. and Vanka, S. P., "*Multigrid Procedure for Three Dimensional Flows on Non-orthogonal Collocated Grids*", International Journal of Numerical Methods in Fluids, vol. 17, pp. 887-904, 1993.

[37] Gjesdal, T., Lossius, M. E. H., "*Comparison of Pressure Correction Smoothers for Multigrid Solutions of Incompressible Flow*", International Journal of Numerical Methods in Fluids, vol. 25, pp. 393-405, 1997.

[38] Y. A. Cengel, J. M. Cimbala, "*Fluid Mechanics Fundementals and Applications*", McGraw-Hill, New York, 2006.

[39] NCFMF Fluid Mechanics Films,
http://web.mit.edu/fluids/www/Shapiro/ncfmf.html

[40] Hunter Rouse (IIHR) Fluid Mechanics Films,
http://www.iihr.uiowa.edu/products/dhrm.html

[41] D. Pines, "*Using Computational Fluid Dynamics to Excite Undergraduate Students about Fluid Mechanics*", ASEE Annual Conference and Exposition, Session: 1055, 2004.

[42] R. D. LaRoche, B. J. Hutchings, R. Muralikrishnan, "*FlowLab: Computational Fluid Dynamics (CFD) Framework for Undergraduate Education*", ASEE Annual Conference and Exposition, Session: 1258, 2002.

[43] R. Bhaskaran, L. Collins, "*Integration of Simulation into the Undergraduate Fluid Mechanics Curriculum using FLUENT*", ASEE Annual Conference and Exposition, Session: 1637, 2003.

[44] R. A. Pieritz, R. Mendes, R. F. A. F. Da Suva, C. R. Maliska, "*CFD Studio: An Educational Software Package for CFD Analysis and Design, Computer Applications in Engineering Education*", vol. 12 (1), pp. 20-30, 2004.

[45] GNU General Public License (GPL),
http://www.gnu.org/copyleft/gpl.html

[46] J. Blanchette, M. Summerfield , "*C++ GUI Programming with Qt 4*", Prentice Hall, 2006.

[47] Scarborough, J. B., "*Numerical Mathematical Analysis*", 4th Edition,John Hopkins University Press, Baltimore, MD, 1958.

[48] Karki, K. C., Patankar, S.V., "*Calculation Procedure For Viscous Incompressible Flows in Complex Geometries*",Numerical Heat Transfer, vol. 14, pp. 295-307, 1988.

[49] Crank, J., Nicolson, P. "*A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat-conduction Type*",, Proc. Cambridge Phil. Soc., vol. 43, pp. 50-67, 1947.

[50] Versteeg, H. K., Malalasekera, W., "*An Introduction to Computational Fluid Dynamics, The Finite Volume Method*", Prentice Hall, 1995.

[51] Jang, D. S., Jetli, R. and Acharya, S., "*Comparison of the Piso, Simpler, and Simplec Algorithm for the Treatment of the Pressure-Velocity Coupling in Steady Flow Problems*", Numerical Heat Transfer, vol. 10, pp. 209-238, 1986.

[52] Cüneyt Sert's Academic Website,
http://www.me.metu.edu.tr/cuneyt

[53] Sert, C., Nakiboglu, G., "*Use of Computational Fluid Dynamics (CFD) in Teaching Fluid Mechanics*", ASEE Annual Conference and Exposition, Session: 1366, 2007.

[54] Thompson, J. F., Soni, B. and Weatherrill, N. P. "*Handbook of Grid Generation*", CRC Press, 1998.

[55] Burggraf, O. R., "*Analytical and Numerical Studies of the Structure of Steady Separated Flows*", Journal of Fluid Mechanics, vol. 24, pp. 113-151, 1966.

[56] Ghia, U., Ghia, K. N. and Shin, C. T., "*High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and Multigrid Method*", Journal of Computational Physics, vol. 48, pp. 387-411, 1982.

[57] Bruneau,C. H., Jouron, C., "*An Efficient Scheme for Solving Steady Incompressible Navier-Stokes Equations*", Journal of Computational Physics, vol. 89, pp. 389-413, 1990.

[58] Vanka, S. P., "*Block-Implicit Multigrid Solution of Navier-Stokes Equations in Primitive Variables*", Journal of Computational Physics, vol. 65, pp. 138-158, 1986.

[59] Goyon, O., "*High-Reynolds Number Solutions of Navier-Stokes Equations Using Incremental Unknowns*", Computer Methods in Applied Mechanics and Engineering, vol. 130, pp. 319-335, 1996.

[60] Schreiber, R., Keller, H. B., "*Driven Cavity Flows by Efficient Numerical Techniques*",Journal of Computational Physics vol.49, pp. 310-333, 1983.

[61] Deng, G. B., Piquet, J., Queutey, P. and Visonneau, M., "*Incompressible Flow Calculations With a Consistent Physical Interpolation Finite Volume Approach* ", Computers Fluids vol.23, No. 8,pp. 1029-1047, 1994.

[62] Botella, O., Peyret, R., "*Benchmark Spectral Results on the Lid-Driven Cavity Flow*", Computers Fluids vol.27, No. 4,pp. 421-433, 1998.

[63] Zhang, L. B., "*Doctoral Thesis*", University of Paris-Sud, Orsay, 1987.

[64] Barton I. E., "*A numerical Study of Flow Over a Confined Backward-Facing Step*", International Journal for Numerical Methods in Fluids, vol 21, pp. 653-665, 1995.

[65] Marinova, R. S., Christov C. I., Marinov T. T., "*A Fully Coupled Solver for Incompressible Navier-Stokes Equations using Operator Splitting*", International Journal of Computational Fluid Dynamics, vol 17(5), pp. 371-385, 2003.

[66] Barton I. E., "*The Entrance Effect of Laminar Flow Over a Backward-Facing Step Geometry*", International Journal for Numerical Methods in Fluids, vol 25, pp. 633-644, 1997.

[67] Peric M., "*A Finite Volume of Eight Discretization Schemes for Two-Dimensional Convection-Diffusion Equations*",Ph.D. Thesis, Mechanical Engineering Department, Imperial College, London, 1985.

[68] Gartling, D. "*A test problem for outflow boundary conditions-flow over a backward-facing step* ", International Journal of Numerical Methods in Fluids vol.24, Issue 3, pp. 291-317, 1990.

[69] Gresho, P.M., Gartling, D.K., Torczynski, J.R., Cliffe, K.A., Winters,K.H., Garratt, T.G., Spence, A. and Goodrich, J.W. "*Is a steady viscous incompressible two-dimensional flow over a backwardfacing step at Re - 800 stable?*", International Journal of Numerical Methods in Fluids vol.17, pp. 501-541, 1993.

[70] Sayma, A. I., Betts, P. L. "*A Finite Element Model for the Simulation of Dense Gas Dispersion in the Atmosphere* ", International Journal of Numerical Methods in Fluids vol.11, pp. 953-967, 1997.

[71] Srinivasan, K., Rubin, S.G. "*Solution-Based Grid Adaptation through Segmented Multigrid Domain Decomposition*", Journal of Computational Physics vol.136, Number 2, pp. 467-493(27), 1997.

[72] Keskar, J. and Lyn, D.A. "*Computations of a laminar backwardfacing step flow at Re - 800 with a spectral domain decomposition method*",International Journal of Numerical Methods in Fluids vol.29, pp. 411-427, 1999.

[73] Armaly, B.F., Durst, F., Pereira, J.C.F. and Schonung, B. "*Experimental and theoretical investigation of backward-facing step flow*", Journal of Fluid Mechanics vol.127, pp. 473-496, 1983.

[74] Lee, T. and Mateescu D. "*Experimental and Numerical Investigation of 2-D Backward-Facing Step Flow*", Journal of Fluids and Structures vol.12, pp. 703-716, 1998.

[75] Sohn, J. "*Evaluation of FIDAP on some classical laminar and turbulent benchmarks*", International Journal of Numerical Methods in Fluids vol.8, pp. 1469-1490, 1988.

[76] Breuer, M., Bernsdorf, J., Zeiser, T. and Durst, F. "*Accurate computations of the laminar flow past a square cylinder based on two different methods: lattice-Boltzmann and finite-volume*", International Journal of Heat and Fluid Flow vol.21, pp. 186-196, 2000.

[77] Galletti, B., Bruneau, C. H. and Zannetti, L. "*Low-order modelling of laminar flow regimes past a confined square cylinder*", Journal of Fluid Mechanics vol. 503, pp. 161-170, 2004.

[78] Mukhopadhyay, A., Biswas, G. and Sundararajan, T. "*Numerical investigation of confined wakes behind a square cylinder in a channel*", International Journal of Numerical Methods in Fluids vol. 14, pp. 1473-1484, 1992.

[79] Roy, A., Bandyopadhyay, G. "*Numerical investigation of confined flow past a square cylinder placed in a channel*", All-India Seminar on Aircrafts and Trans-atmospheric Vehicles : Missions, Challenges and Perspectives, Kolkata, May 28-29, 2004.

[80] Davis, R. W., Moore, E. F. and Purtell, L. P. "*A Numerical-experimental study of confined flow around rectangular cylinders*", Physics of Fluids vol. 27, No. 1, January 1984.

[81] Perry, A. E., Chong, M. S. and Lim T. T. "*The Vortex Shedding process behind Two-Dimensional Bluff Bodies*", Journal Fluid Mechanics, vol. 116, pp. 77-90, 1982.

[82] Eaton, B. E. "*Analysis of Laminar Vortex Shedding behind a Circular Cylinder by Computer-Aided Flow Visualisation*", Journal Fluid Mechanics, vol. 180, pp. 117-145, 1987.

[83] Sharma, A., Eswaran, V. "*Heat and Fluid Flow Across a Square Cylinder in the Two-Dimensional Laminar Flow Regime*", Numerical Heat Transfer, Part A, vol. 45, pp. 247-269, 2004.