SOLUTION TO INCOMPRESSIBLE NAVIER STOKES EQUATIONS BY USING FINITE ELEMENT METHOD

by

WANCHAI JIAJAN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN AEROSPACE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2010

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Brian Dennis for constantly motivating and encouraging me, and also for his invaluable advice during the course of my studies. I wish to thank my academic advisors Dr. Donald R. Wilson for their interest in my research and for taking time to serve in my thesis committee.

I would also like to extend my appreciation to Royal Thai Air Force for providing scholarship for my Master studies. I wish to thank all my colleagues and co-workers at Aeronautical and Aviation Engineering Department, Royal Thai Air Force Academy for their support and encouragement.

Finally, I would like to express my extreme gratitude to my parents, sister and wife for their sacrifice, encouragement and patience. I also thank several of my friends, who have helped me throughout my career.

July 7, 2010

ABSTRACT

SOLUTION TO INCOMPRESSIBLE NAVIER STOKES EQUATIONS
BY USING FINITE ELEMENT METHOD

WANCHAI JIAJAN, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Brian Dennis

The finite element method has become a popular method for the solution of the incompressible Navier-Stokes equations. High Reynolds number cases require fine meshes so computational efficiency becomes an important factor in algorithm and code development. In this work, a Galerkin finite element method is proposed to solve the two dimensional incompressible Navier-Stokes equations. This approach typically leads to a sparse and indefinite matrix that is difficult to solve efficiently. The formation of an indefinite matrix is avoided in the present work by introducing an artificial compressibility term in the continuity equation. The concept of this method is to transform the elliptic incompressible equation to the hyperbolic type compressible system which can be solved by standard implicit or explicit time-marching methods.

The primitive variables are used for flow properties. The method features unequal order interpolation for the velocity variables and pressure. The Taylor Galerkin formulation is introduced as a stabilization procedure to eliminate the numerical oscillations that occur in convection dominated flows. The Newton-Raphson method is used to resolve the non-linearity at each time step.

In order to test the method, a finite element code was developed for triangular meshes. The code was applied to the standard lid driven cavity problem. The numerical results were compared with benchmark results from the literature.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Most of every real situation in fluid flows is characterized by the Navier Stokes equations that are the model of nonlinear partial differential equation. The nonlinearity is due to convective acceleration, which is an acceleration associated with the change in velocity over position. These nonlinearities make most problems difficult or impossible to solve. However, it is very useful for engineering to describe and analyze not only the physics of fluid flow problems but also more complex materials if the Navier Stokes equations can be solved. There are many cases in fluid flows that can be represented by the Navier Stokes equation such as ocean currents, water flow in a pipe, air flow around a wing, and blood flow.

At the present time, there are many approaches that can be employed to solve these equations such as the finite difference, finite volume, and finite element method. These methods are widely used in the numerical solution of Navier-Stokes equations. For example, the finite different scheme based on artificial compressibility method to solve unsteady incompressible Navier-Stokes equations was discussed by Chorin [1], and the finite different scheme based on SIMPLE, SIMPLER and Vorticity-Stream function approaches have been studied by Matyka [2]. For these schemes, both of primitive variable and vorticity-stream function approaches were employed as the main variables to solve incompressible Navier-Stokes equations for both of unsteady and steady case.

However, as presented in numerous literature of numerical method, the finite element method has emerged as a valuable tool for the solution of the Navier-Stokes equations, especially where complex geometries or boundary conditions render analytical or other numerical solutions difficult or impossible. To indicate this numerical is more popular than the others, we can summarize the comparing of advantage and disadvantage for three different schemes in the table 1.1.

Table 1.1. Comparing the advantage and disadvantage of Finite element, Finite difference, and finite volume scheme

FEM
+high accuracy
+ easy to treatment of complex geometries
- mesh and order(hp)-adaptation difficult
FDM
+easy implementation
-problems along curved boundaries
- difficult stability and convergence analysis
- mesh adaptation difficult
FVM
+based on physical conservation properties
- problems on unstructured meshes
- difficult stability and convergence analysis.
- only heuristic mesh adaptation

From table 1.1, although the finite element method is better than the the others, there are many studies of the solution to Navier-Stokes equations by using finite difference method that have been succeed and widely used such as finite difference scheme based on artificial compressibility method (AC method). Because of the AC method can be used to deal with the difficulty of incompressibility condition by using

the concept of the transformation of continuity equation, this method has been widely used for both of finite different and finite element method.

The AC method was originally introduced by Chorin (1967) with the objective of solving the steady state incompressible Navier-Stokes equations [3]. Moreover, it was also used to solve for unsteady case. For example, [4] and [5] were some of the first to extend the AC method to the solution of the unsteady incompressible Navier-Stokes equations. The concept of this method is to transform the elliptic incompressible continuity equation to hyperbolic compressible system by adding the artificial compressibility term in continuity equation. As a result, the new transformed equations can be solved directly by standard time-dependent approaches that is not complicated to apply in the solution. We find numerous example of the AC method used for solving both of steady and unsteady case such as [4],[6], [7], [8], [9] and [10].

Following the success of the finite difference scheme based on AC method, several finite element solution based on AC methods have been presented for example the Characteristic-Based-Split algorithm (CBS method) based on AC method which features equal order of all variables was employed to solve steady and unsteady incompressible Navier-Stokes equations by [11]. They used both of implicit and explicit scheme as the time integration approach to deal with the time dependent problem. The advantages of the proposed CBS scheme based on AC method include easy parallelization and implementation procedure. However, this method make the large number of iterations to reach the steady state when using both of explicit and implicit scheme.

In this paper, the Galerkin finite element formulation based on AC method of two dimensional unsteady incompressible Navier Stokes equations will be discussed. The unequal order primitive variable of velocity components and pressure will be employed as the main approach. The Taylor Galerkin technique will be employed

as the stabilization procedures to deal with the numerical oscillations due to the convection dominated at the high Reynolds number. For the numerical solution, the iterative method of Newton-Raphson will be used to solve the set of non-linear equations and the backward different scheme will be employed as the time-integration approach to deal with the time dependent term.

The numerical method is verified by the benchmark of lid-cavity driven that have been become standard problem to test incompressible Navier-Stokes flow. The efficiency, the accuracy, and the steady-state convergence of the Galerkin finite element formulation based on AC method are verified by comparing with numerical solutions available in the literature of Ghia et al. [12].

1.2 Thesis Overview

This paper is organized as follows. chapter 2 discusses the finite element procedure of incompressible Navier-Stokes equation based on AC method. Additionally, the selection of interpolation functions for finite element model and the coefficient matrices construction will be discussed. chapter 3 details the numerical solution and express the finite element programming. chapter 4 presents the results of numerical solution. chapter 5 gives the conclusions and recommendations for future work.

CHAPTER 2

INCOMPRESSIBLE NAVIER STOKES EQUATION

2.1 Introduction

As mentioned in previous chapter, the Galerkin finite element formulation based on AC method will be employed to solve the unsteady incompressible Navier-Stokes equations. Before solving, in this chapter, we will present the physical of mathematic model of these equations and dicretize these equations into the weak forms by using weak formulation throughout transform these weak forms to the finite element matrix that is suitable for programming construction.

In the section 2.2, the derivation of the governing equations of the two dimensional unsteady incompressible Navier Stokes equations will be expressed. Under isothermal condition, the energy equation is uncoupled from the momentum equations (energy equation can be neglected). Thus, only Incompressible Navier Stokes equation (momentum) and continuity equation need to be solved [13].

For the section 2.3, to deal with the difficulty of convection dominated due to the high Reynolds number fluid flows, the stabilized method will be presented. The incompressible Navier-Stokes equations based on stabilized tecniques will be discretized by the finite element method. The finite element models are developed base on the weak formulation of the viscous, incompressible fluid under isothermal condition [13].

In order to deal with the difficulty of incompressible continuity, the AC method will be presented in section 2.4. This scheme will be employed to transform the

incompressible elliptic to compressible hyperbolic continuity equation. As a result, the new transformed equations can be solved by the standard time integration approaches.

The section 2.5 will be devoted to the selection of interpolation functions to finite element model. Because of the interpolation function have more effect with the accuracy result, this section will explain how to choose the shape function that satisfy LBB compatibility condition and suitable for the solutions.

For the last sections, the construction of finite element matrix formulas will be derived. In order to establish the coefficient matrices easily, the transformation of shape function will be presented.

2.2 Governing Equations

Let us start with the governing equations of unsteady incompressible Navier-Stokes equations. These equations are the conservative form of continuity equation, momentum equation, and energy equation. The momentum equations consist of the set of nonlinear partial differential equations in the terms of velocities components. As mentioned before, under isothermal condition, the energy equation is uncoupled from the momentum equations. Thus, only Incompressible Navier Stokes equations and continuity equation need to be solved [13].

The equations are written here is the 2-D unsteady Incompressible Navier-Stokes equations which are expressed into the form of Cartesian derivative components.

Continuity equation:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.1}$$

Where u and v represents the velocity components. For the finite element method based on artificial compressibility, it should be noted that the artificial compressibility

term will be added to the continuity equation in order to transform the incompressible elliptic partial differential equation to compressible hyperbolic mathematic form. This method will be discussed in section 2.4.

Momentum equation:

x- direction:

$$\rho\left[\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]=\frac{\partial(\sigma_x-p)}{\partial x}+\frac{\partial \tau_{xy}}{\partial y} \tag{2.2}$$

y- direction:

$$\rho\left[\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]=\frac{\partial \tau_{xy}}{\partial x}+\frac{\partial(\sigma_y-p)}{\partial y} \tag{2.3}$$

Constitutive equations:

$$\sigma_x=2\mu\frac{\partial u}{\partial x} \tag{2.4}$$

$$\sigma_y=2\mu\frac{\partial v}{\partial y} \tag{2.5}$$

$$\tau_{xy}=\mu\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right] \tag{2.6}$$

Where u and v represents the velocity components, p is the pressure, ρ is the density, and μ is the viscosity. It should be note that the body forces does not appear in (2.2), and (2.3) because they may be grouped with the pressure terms when the body force can be expressed as the gradient of a potential function. Substitute (2.4)-(2.6) into (2.2), and (2.3). Thus, the final momentum equation can be expressed as follow

x- direction:

$$-\rho\left[\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x}-p\right]+\mu\frac{\partial}{\partial y}\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]=0 \tag{2.7}$$

y- direction:

$$-\rho\left[\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]+\frac{\partial}{\partial y}\left[2\mu\frac{\partial v}{\partial y}-p\right]+\mu\frac{\partial}{\partial x}\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]=0 \tag{2.8}$$

From the equations (2.7) and (2.8), it is obvious that the first term of these equations consist of nonlinear term that is due to convective inertia term. As we have

mentioned before, the nonlinearity will make the problem difficulty to solve. Thus, we have to apply some formulations that can be employ to deal with the non-linear term. In this paper, the iterative method of Newton-Raphson will be chosen to solve this problem. This method will be discussed in chapter 4.

The second and the third term are due to the force from viscosity. If the ratio of convective and viscosity term close to zero, these equations will be changed to Stokes equation that is the linear partial differential equation.

2.3 Finite Element Method

Before introducing finite element techniques for the numerical of the unsteady Navier-Stokes equations, we will present the two main difficulties that encounter in the solution.

A first difficulty is due to the presence of nonlinear and non-sysmetric convective terms in the momentum equation (2.7) and (2.8). The nonlinearity in these equations make the problem difficult or complicate to solve. To deal with this difficulty, the iterative method such as Newton-Raphson and Picard method must be employed to reduce a set of non-linear algebraic equations to a linear algebraic system. In this paper, the iterative method of Newton-Raphson will be selected as the main algorithm in the solutions. This method will be discussed in the next chapter. In addition, this difficulty will increase when the large value of the Reynolds number apply in the solutions. When the Reynolds number is increased, the flows are convection dominated. As a result, the unpredicted oscillations encounter in the solutions. Thus, in order to deal with this problem, stabilization techniques such as SUPG, GLS, SGS or LS, and Taylor Galerkin method must be used to provide the meaningful finite element solutions at high Reynolds number [14]. In this paper, for time dependent problem, the Taylor Galerkin stabilized method will be selected to

stabilize the numerical oscillations in the solutions of the unsteady incompressible Navier-Stokes equations.

Another difficulty is the incompressibility condition. From the continuity equation and momentum equations as expressed in (2.1),(2.7), and (2.8), it can be seen that the continuity equation consists of a constraint on the velocity field which must be divergence free and the momentum equations consist of both of velocity field and pressure. From these equations, the pressure term has to be considered as a variable not related to any constitutive equation. However, the pressure term represents an addition degree of freedom that needed to satisfy the compressibility constraint. Thus, in order to satisfy the divergence-free velocity, the pressure must adjust itself instantaneously that means the pressure is acting as a Lagrangian multiplier of the incompressibility constraint and thus there is a coupling between the velocity and the pressure unknowns [14].

There are many formulations that have been proposed in the literature to deal with the difficulty of incompressibility condition such as penalty formulation and mixed finite element method. In this paper, the mixed finite element method based on artificial compressibility method will be chosen to solve this problem.

2.4 Taylor Galerkin Stabilized Method

In order to deal with the difficulty of convection dominated, the stabilized techniques need to be required for the solutions. As we have mentioned, in this study, the Taylor Galerkin Stabilized formulation will be selected to deal with the spatial oscillations due to the discretization of convection transport term at high Reynolds number. The concept of Taylor Galerkin method is to transform the time domain of momentum equations (2.7) and (2.8) to the second order accuracy of Taylors series expansion [14]. After derivation as shown in reference [14], the equations of (2.7) and

(2.8) are replaced as follow

x-direction:

$$-\rho\left[\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x}-p\right]+\mu\frac{\partial}{\partial y}\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]$$
$$+u\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]+v\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]=0 \tag{2.9}$$

y-direction:

$$-\rho\left[\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]+\frac{\partial}{\partial y}\left[2\mu\frac{\partial v}{\partial y}-p\right]+\mu\frac{\partial}{\partial x}\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]$$
$$+u\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]+v\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]=0 \tag{2.10}$$

From the equation of (2.9) and (2.10), it is obvious that not only the convective terms but also the Taylor Galerkin Stabilized that express in the last two terms are non-linear equations. As we mentioned in previous section, although the nonlinearity make the problem difficult to solve, this difficulty can be solve by the iterative method that will be discussed in the next chapter.

2.5 Primitive variable formulation (velocity and pressure variable)

As mentioned in previous section, the difficulty of incompressibility condition can be solved by the primitive variable formulation. It was also called mixed finite element method. In order to present how to deal with this difficulty, let us recall the pressure acts as a Lagrangian multiplier of the incompressibility constraint. Then, the algebraic system for the nodal values of velocity and pressure in a Galerkin formulation will be governed by a partitioned matrix with a null submatrix on the diagonal. Solvability of the algebraic system depends on a proper choice of finite element spaces for velocity and pressure. These choices must satisfy a LBB compatibility condition [14].

In order to make more understanding how to deal with the difficulty of incompressibility condition, the procedure of finite element method based on primitive variable formulation will be shown as step by step. We will start with the dicretization of incompressible Navier-Stokes equation into weak form by using weak formulation. Then, the selection of finite element spaces for velocity and pressure will be discussed.

Step1: Discretize to weak form, let us start with the discretization of the equation of (2.1), (2.9), and (2.10). The finite element models of these equations will be developed into their weak forms. The weighted-integral statements are taken into these equations over a typical element in domain. Thus, the weak form of equation (2.1), (2.9), and (2.10) are expressed as follow.

$$\int_{\Omega^e} Q\left[\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right] d\Omega = 0 \tag{2.11}$$

$$\begin{aligned}&\int_{\Omega^e} W\left[-\rho\left[\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]+\frac{\partial}{\partial x}\left[2\mu\frac{\partial u}{\partial x}-p\right]+\mu\frac{\partial}{\partial y}\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]\right] d\Omega\\&+\int_{\Omega^e} W\left[u\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]+v\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]\right] d\Omega = 0\end{aligned} \tag{2.12}$$

$$\begin{aligned}&\int_{\Omega^e} W\left[-\rho\left[\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]+\frac{\partial}{\partial y}\left[2\mu\frac{\partial v}{\partial y}-p\right]+\mu\frac{\partial}{\partial x}\left[\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right]\right] d\Omega\\&+\int_{\Omega^e} W\left[u\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]+v\frac{\Delta t}{2}\frac{\partial}{\partial x}\left[u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]\right] d\Omega = 0\end{aligned} \tag{2.13}$$

Where Q and W are weight functions that will be equated in the Rayleigh-Ritz-Galerkin finite element models. The first weight function represents pressure and the second represents velocities components.

Step2: Selection of the approximated function, since the Incompressible Navier Stokes equations have already discretized to weak form, the selection of the approximated functions to these weak forms will be discussed in this section. As we have

mentioned, solvability of the algebraic system depends on a proper choice of finite element spaces for velocity and pressure. Moreover, these choices must be satisfied by LBB compatibility condition. From these conditions, the choice of the weight functions must be restricted to the spaces of approximation functions that used for the pressure and velocity fields. Thus, it is clear that the dependent variables of u, v, and p are approximated by expansions form.

$$u(x,y,t) = \sum_{i=1}^{n} \psi(x,y)u_i(t) \tag{2.14}$$

$$v(x,y,t) = \sum_{i=1}^{n} \psi(x,y)v_i(t) \tag{2.15}$$

$$p(x,y,t) = \sum_{i=1}^{n} \phi(x,y)p_i(t) \tag{2.16}$$

Where ψ and ϕ are vectors of Interpolation functions,u and v are the vectors of nodal values of velocity components, and p are the vectors of nodal values of pressure. For the condition of LBB compatibility, we will discussed in next section,

From the expansion of approximate function (2.14)-(2.16), as the weight functions (Q, W) and (ϕ, ψ) are identical, we can substitute the interpolation functions (2.14)-(2.16) into the weak form of (2.11)-(2.13) and integrate them by using by part integration. Thus, the result of incompressible Navier Stokes equation in the finite element form can be expressed as follow.

Continuity equation:

$$\int_{\Omega^e} \left[\phi \frac{\partial \psi^T}{\partial x} u + \phi \frac{\partial \psi^T}{\partial y} v \right] d\Omega = 0 \tag{2.17}$$

Momentum equations:

x-direction:

$$\begin{aligned}
&\int_{\Omega^e}\left[\left(\rho\psi\psi^T\frac{\partial u}{\partial t}\right)+\rho\left(\psi(\psi^T u)\frac{\partial\psi^T}{\partial x}+\psi(\psi^T v)\frac{\partial\psi^T}{\partial y}\right)\right]ud\Omega\\
&+\int_{\Omega^e}\left[\left(2\mu\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial x}+\mu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial y}\right)\right]ud\Omega\\
&+\frac{\Delta t}{2}\int_{\Omega^e}\left[uu\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial x}+uv\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial y}+vu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial x}+vv\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial y}\right]ud\Omega\\
&+\int_{\Omega^e}\left[\left(\mu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial x}\right)v-\left(\frac{\partial\psi}{\partial x}\phi^T\right)p\right]d\Omega=\int_{\Omega^e}\rho\psi f_i d\Omega
\end{aligned}\tag{2.18}$$

y-direction:

$$\begin{aligned}
&\int_{\Omega^e}\left[\left(\rho\psi\psi^T\frac{\partial v}{\partial t}\right)+\rho\left(\psi(\psi^T u)\frac{\partial\psi^T}{\partial x}+\psi(\psi^T v)\frac{\partial\psi^T}{\partial y}\right)\right]vd\Omega\\
&+\int_{\Omega^e}\left[\left(2\mu\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial x}+\mu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial y}\right)\right]vd\Omega\\
&+\frac{\Delta t}{2}\int_{\Omega^e}\left[uu\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial x}+uv\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial y}+vu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial x}+vv\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial y}\right]vd\Omega\\
&+\int_{\Omega^e}\left[\left(\mu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial x}\right)u-\left(\frac{\partial\psi}{\partial y}\phi^T\right)p\right]d\Omega=\int_{\Omega^e}\rho\psi f_i d\Omega
\end{aligned}\tag{2.19}$$

Thus, the equations (2.17)-(2.19) can be written in the form of matrix

$$\begin{aligned}
&\begin{bmatrix}M&0&0\\0&M&0\\0&0&0\end{bmatrix}\begin{bmatrix}\dot u\\\dot v\\\dot p\end{bmatrix}+\begin{bmatrix}C(u,v)&0&0\\0&C(u,v)&0\\0&0&0\end{bmatrix}\begin{bmatrix}u\\v\\p\end{bmatrix}\\
&+\begin{bmatrix}2K_{11}+K_{11}&K_{21}&-Q_1\\K_{21}&K_{11}+2K_{11}&-Q_1\\-Q_1^T&-Q_1^T&0\end{bmatrix}\begin{bmatrix}u\\v\\p\end{bmatrix}\\
&+\begin{bmatrix}Kse(u,v)&0&0\\0&Kse(u,v)&0\\0&0&0\end{bmatrix}\begin{bmatrix}u\\v\\p\end{bmatrix}=\begin{bmatrix}F1\\F2\\0\end{bmatrix}
\end{aligned}\tag{2.20}$$

Table 2.1. Coefficient matrix formulas of incompressible Navier-Stokes equations based on artificial compressibility method

Abbriviation	Formulas	Represent
M	$\int_{\Omega^e} \rho\psi\psi^T d\Omega$	Mass matrix
C(u,v)	$\int_{\Omega^e} \rho \left(\psi(\psi^T u)\frac{\partial\psi^T}{\partial x} + \psi(\psi^T v)\frac{\partial\psi^T}{\partial y}\right)$	Convective matrix
K_{ij}	$\int_{\Omega^e} \left(\mu\frac{\partial\psi}{\partial x_i}\frac{\partial\psi^T}{\partial x_j}\right) d\Omega$	Diffusion matrix
Q_i	$\int_{\Omega^e} \left(\frac{\partial\psi}{\partial x_i}\phi^T\right) d\Omega$	Gradient matrix
$Kse(u,v)$	$\frac{\Delta t}{2}\int_{\Omega^e} \left(u_i u_j \frac{\partial\psi}{\partial x_i}\frac{\partial\psi^T}{\partial x_j}\right) d\Omega$	Stabilized matrix
F_i	$\int_{\Omega^e} \rho\psi f_i d\Omega$	Force vector

or

$$[M]\dot{\mathbf{U}} + [K(u)]\mathbf{U} = \{\mathbf{F}\} \tag{2.21}$$

Where the coefficient matrices shown in equations (2.20) are defined in table 2.1

From the matrix form of (2.20), one more difficulty that encounter in the numerical solution is the presence of zeros on the matrix diagonals corresponding to the time derivative of pressure. Because of the lack of time derivative for the pressure in continuity equation, the element mass matrix is singular. Thus, the explicit algorithm cannot be applied to this equation. However, such method can be used with the artificial compressibility method that will be discussed in the next section. The concept of this method have been well known as continuity transformation by adding artificial time derivative term of pressure. As the result, the difficulty of singular mass matrix cannot encounter in the solutions.

2.6 Artificial Compressibility Method (AC method)

One of the early techniques proposed for solving the incompressible N-S equation in primitive variable form was the artificial compressibility method of Chorin (1967). The concept of this method is to transform the elliptic partial incompressible

equation to hyperbolic compressible partial differential form by adding the artificial term into continuity equation. The addition of artificial compressibility term will be vanished when the steady state solution is reached [15]. With the addition of this term to continuity equation, the N-S equation will be changed to a mixed type of hyperbolic-parabolic equations which can be solved by standard time- dependent approach . In order to describe this method, let us apply the artificial term into continuity equation. Thus, the equation (2.1) is replaced by

$$\frac{\partial \rho}{\partial t} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.22}$$

where ρ is the artificial density. The artificial density is related to the pressure by the artificial equation of state.

$$p = \rho\beta \tag{2.23}$$

Substitute (2.23) into (2.22), we got the new form of artificial compressibility continuity equation as

$$\frac{1}{\beta}\frac{\partial p}{\partial t} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.24}$$

In order to get the new matrix formula of Incompressible Navier Stokes equations, let us repeat the procedure step of discretization as mentioned in previous section. From the discretization of (2.24) based on artificial compressibility method, the weak form of (2.22) can be expressed as

$$\int_{\Omega} \left(\frac{Q}{\beta}\frac{\partial p}{\partial t} + Q\frac{\partial u}{\partial x} + Q\frac{\partial v}{\partial y} \right) d\Omega = 0 \tag{2.25}$$

or

$$\int_{\Omega} \left(\frac{\phi\phi^T}{\beta}\frac{\partial p}{\partial t} + \phi\frac{\partial \psi^T}{\partial x}u + \phi\frac{\partial \psi^T}{\partial y}v \right) d\Omega = 0 \tag{2.26}$$

Take equation (2.26) into the matrix form (2.20) by using the same procedure as mentioned in previous section. As a result, we got the final matrix form of 2-D

unsteady incompressible Navier Stokes equation based on artificial compressibility method as expressed in (2.30)

$$\begin{bmatrix} [M] & 0 & 0 \\ 0 & [M] & 0 \\ 0 & 0 & -[M_p] \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} C(u,v) & 0 & 0 \\ 0 & C(u,v) & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} + \begin{bmatrix} 2K_{11}+K_{11} & K_{21} & -Q_1 \\ K_{21} & K_{11}+2K_{11} & -Q_1 \\ -Q_1^T & -Q_1^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} + \begin{bmatrix} Kse(u,v) & 0 & 0 \\ 0 & Kse(u,v) & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} F1 \\ F2 \\ 0 \end{bmatrix} \tag{2.27}$$

or

$$[M]\dot{\mathbf{U}} + [K(u)]\mathbf{U} = \{\mathbf{F}\} \tag{2.28}$$

Where $M_p = \frac{1}{\beta}\int_{\Omega^e} \phi\phi^T d\Omega$ represents mass matrix (pressure term). From the equation (2.27), it is obvious that the mass matrix is not singular anymore. Thus, the standard time-dependent approach can be employed to solve this equation.

In addition, the coefficient matrices in table 2.1, it can be seen that the integration form of coefficient matrix is depended on the type of element area or type of interpolation function. As we mentioned before in section 2.4, the accuracy result is depended on the selection of interpolation functions. Thus, it is necessary to select the interpolation functions that proper for incompressible Navier Stokes equation. For the choice of interpolation function, we will discuss more detail in next section.

2.7 The Choice of Interpolation Functions

As mentioned in previous section, the difficulty of incompressibility have encountered in the incompressible Navier Stokes equations, the selection of interpolation functions must be concerned in order to get the accuracy result. Thus, the condition to select the interpolation functions for these equations should be written here for guiding how to choose.

2.7.1 The condition of interpolation functions

The choice of interpolation functions used for the pressure variable in the mixed finite element model is further constrained by the role of pressure. As we have mentioned, the pressure can be interpreted as a Lagrange multiplier that serve to enforce the incompressibility constraint on the velocity field. For this reason, in order to prevent an overconstrained system of discreted equations:

1. The interpolation used for pressure must be at least one order lower than that used for the velocities (unequal order interpolation)[13].

2. Pressure need not be made continuous across element because the pressure variable does not constitute a primary variable of the weak form.

From these conditions, it is clear that 2-D linear interpolation functions must be the lowest order that is represented by the form of pressure. In this thesis, 2-D quadratic interpolation functions will be selected to substitute into the table of coefficient matrix formulas. In order to compute the matrix form easily, we will choose the quadratic triangular element (6-nodes) as the interpolation functions.

In addition, in order to get the convergent solutions, the mixed finite element method must satisfy the LBB compatibility condition which state that the existence of stable finite element approximate solution $(U, p)$to Navier-Stokes equations de-

pends on choosing a pair of weight function (Q, W). However, the discussion of LBB condition is beyond the scope of the present study and will not be discussed here.

2.7.2 Quadratic triangular elements (6-nodes)

In this paper, the quadratic triangular element (6-nodes) will be selected as the interpolation functions in order to directly compute the finite element matrix form by using the exact integral formula. To satisfy the conditions of shape functions which state that the order for pressure must be at least one order lower than that used for the velocities, the pressure variable should be located at corner nodes (3-nodes triangular) and velocity components are located at all of six nodes as shown in figure 2.1.

From the figure 2.1 that expresses the location of velocity components and pressure, it is obvious that this element consist of 6 unknowns for velocities and 3 unknowns for pressure. Thus, the quadratic interpolation functions represent velocity components while linear interpolation function represents the form of pressure. As a result, the unknown variables for velocities and pressure will be represented as 15 unknowns per each element. Thus, the expansions of dependent variable u, v,and p are expressed as

$$u(x, y, t) = \sum_{i=1}^{6} \psi(x, y) u_i(t) \tag{2.29}$$

$$v(x, y, t) = \sum_{i=1}^{6} \psi(x, y) v_i(t) \tag{2.30}$$

$$p(x, y, t) = \sum_{i=1}^{3} \phi(x, y) p_i(t) \tag{2.31}$$

From expansion of approximated function of (2.29)-(2.31), it can be seen that the vectors of interpolation functions in these equations are the main functions that must be employed to substitute in the weak form (2.27). In this paper, in order to construct

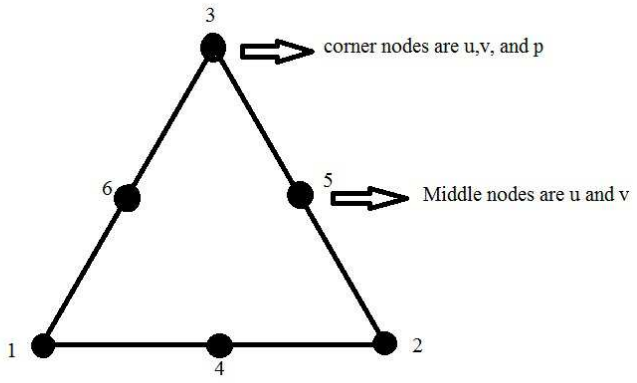

Figure 2.1. Quadratic triangular elements.

the finite element matrix easily by using the exact integral formula, the interpolation functions from the theory of area coordinate will be employed as the quadratic and linear triangular interpolation functions for velocities and pressure, respectively. These interpolation functions are expressed in (2.32) and (2.33).

$$\begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \end{bmatrix} = \begin{bmatrix} L_1(2L_1-1) \\ L_2(2L_2-1) \\ L_3(2L_3-1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{bmatrix} \tag{2.32}$$

And

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{2.33}$$

WhereL_1, L_2 andL_3 are the functions of Cartesian coordinate(x, y) and elements area as expressed in (2.34)

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2A}(a_1 + b_1x + c_1y) \\ \frac{1}{2A}(a_2 + b_2x + c_2y) \\ \frac{1}{2A}(a_3 + b_3x + c_3y) \end{bmatrix} \tag{2.34}$$

Where A is an element area and a_i, b_i and c_i are depended on the coordinate by point to point.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} (x_2y_3 - x_3y_2) \\ (x_3y_1 - x_1y_3) \\ (x_1y_2 - x_2y_1) \\ y_2 - y_3 \\ y_3 - y_1 \\ y_1 - y_2 \\ x_3 - x_2 \\ x_1 - x_3 \\ x_2 - x_1 \end{bmatrix} \tag{2.35}$$

At this time, the unsteady incompressible Navier Stokes equations have been computed to the finite element matrix form based on artificial compressibility method and the area coordinate forms of quadratic triangular interpolation functions are completed. In order to compute the coefficient matrices, the equation of (2.32) and (2.33) will be substituted in the equation (2.28). For the construction of finite element matrix formulas, we will discuss in the next section.

2.8 The Construction of Finite Element Matrix Formulas

Before construction the finite element matrix, let us recall the coefficient matrices formulas in table 2.1

As shown in table 2.1, the coefficient matrices will be derived by the substitution of the interpolation function (2.32) and (2.33). In order to derive the coefficient matrices formulas easily, we will start with the transformation of interpolation functions.

2.8.1 The transformation of interpolation functions.

Form the equation (2.32), the vector of interpolation functions can be transformed to the new matrix that consist of the coefficient matrix [A] and the vector form of matrix [R] as expressed in equation (2.36).

$$\psi = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1 L_2 \\ L_2 L_3 \\ L_3 L_1 \end{bmatrix} \tag{2.36}$$

Where

$$[A]_{6x6} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \tag{2.37}$$

And

$$[R]_{6x1} = \begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1L_2 \\ L_2L_3 \\ L_3L_1 \end{bmatrix} \tag{2.38}$$

As we got the new transformation of interpolation function in equation (2.36), we will substitute this matrix into the coefficient matrices formulas in table 2.1.

2.8.2 The coefficient matrices construction

Once the matrix transformation for the interpolation functions are completed, the coefficient matrices formulas in table 2.1 will be derived. Let us start with the coefficient matrix of time derivative term of velocity components and pressure(mass matrix).

Mass matrix: $M = \rho \int_{\Omega^e} \psi\psi^T d\Omega$

Let us take the new matrix transformation into mass matrix and integrate this form by using the exact integral formula of $\int_{A^e} L_1^a L_2^b L_3^c dA = \frac{a!b!c!}{(a+b+c+2)!} 2A$, we got

$$M = [A][A]^T \frac{2A}{360} \begin{bmatrix} 12 & 2 & 2 & 3 & 1 & 3 \\ 2 & 12 & 2 & 3 & 3 & 1 \\ 2 & 2 & 12 & 1 & 3 & 3 \\ 3 & 3 & 1 & 2 & 1 & 1 \\ 1 & 3 & 3 & 1 & 2 & 1 \\ 3 & 1 & 3 & 1 & 1 & 2 \end{bmatrix} \tag{2.39}$$

Where A is an element area. The derivation of mass matrix is available in APPENDIX B.

Mass matrix: $M_p = \frac{1}{\beta}\int_{\Omega^e} \phi\phi^T d\Omega$

$$\phi\phi^T = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix} = \begin{bmatrix} L_1^2 & L_1L_2 & L_1L_3 \\ L_1L_2 & L_2^2 & L_2L_3 \\ L_1L_3 & L_2L_3 & L_3^2 \end{bmatrix} \tag{2.40}$$

Let us call the integration of equation (2.40) as $[G]$ matrix, we got

$$G = \int_{A^e} \begin{bmatrix} L_1^2 & L_1L_2 & L_1L_3 \\ L_1L_2 & L_2^2 & L_2L_3 \\ L_1L_3 & L_2L_3 & L_3^2 \end{bmatrix} dA = \frac{A}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{2.41}$$

Thus, the final form of mass matrix M_p (Artificial compressibility term) is

$$M_p = \frac{1}{\beta}\int_{A^e} \begin{bmatrix} L_1^2 & L_1L_2 & L_1L_3 \\ L_1L_2 & L_2^2 & L_2L_3 \\ L_1L_3 & L_2L_3 & L_3^2 \end{bmatrix} dA = \frac{1}{\beta}[G] \tag{2.42}$$

Diffusion matrix: $K_{11} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial x} d\Omega$ and $K_{22} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial y} d\Omega$

The derivative form of interpolation function can be expressed as

$$\frac{\partial\psi}{\partial x} = [A]\frac{\partial}{\partial x}\begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1L_2 \\ L_2L_3 \\ L_3L_1 \end{bmatrix} = [A]\frac{1}{2A}\begin{bmatrix} 2b_1 & 0 & 0 \\ 0 & 2b_2 & 0 \\ 0 & 0 & 2b_3 \\ b_2 & b_1 & 0 \\ 0 & b_3 & b_2 \\ b_3 & 0 & b_1 \end{bmatrix}\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{2.43}$$

And

$$\frac{\partial \psi}{\partial y} = [A]\frac{\partial}{\partial x}\begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1L_2 \\ L_2L_3 \\ L_3L_1 \end{bmatrix} = [A]\frac{1}{2A}\begin{bmatrix} 2c_1 & 0 & 0 \\ 0 & 2c_2 & 0 \\ 0 & 0 & 2c_3 \\ c_2 & c_1 & 0 \\ 0 & c_3 & c_2 \\ c_3 & 0 & c_1 \end{bmatrix}\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \tag{2.44}$$

Or, we can abbreviate these forms as

$$\frac{\partial \psi}{\partial x} = [A][B][H] \tag{2.45}$$

$$\frac{\partial \psi}{\partial y} = [A][C][H] \tag{2.46}$$

Then, substitute (2.45) and (2.46) into diffusion matrix, we got

$$K_{11} = \int_{A^e} \mu \frac{\partial \psi}{\partial x}\frac{\partial \psi^T}{\partial x} dA = \mu[A][B][A]^T[B]^T \int_{A^e} [H][H]^T dA \tag{2.47}$$

$$K_{22} = \int_{A^e} \mu \frac{\partial \psi}{\partial y}\frac{\partial \psi^T}{\partial y} dA = \mu[A][C][A]^T[C]^T \int_{A^e} [H][H]^T dA \tag{2.48}$$

Let us recall [G] matrix to substitute into (2.47) and (2.48), we got the final diffusion matrix formula for x and y direction, respectively as

$$K_{11} = \mu[A][B][G][B]^T[A]^T \tag{2.49}$$

$$K_{22} = \mu[A][C][G][C]^T[A]^T \tag{2.50}$$

Similarly procedure for

Diffusion matrix: $K_{12} = \int_{\Omega^e} \mu \frac{\partial \psi}{\partial x}\frac{\partial \psi^T}{\partial y} d\Omega$ and $K_{21} = \int_{\Omega^e} \mu \frac{\partial \psi}{\partial y}\frac{\partial \psi^T}{\partial x} d\Omega$

We got

$$K_{12} = \int_{\Omega^e} \mu \frac{\partial \psi}{\partial x}\frac{\partial \psi^T}{\partial y} d\Omega = \mu[A][B][G][A]^T[C]^T \tag{2.51}$$

$$K_{21} = \int_{\Omega^e} \mu \frac{\partial \psi}{\partial y}\frac{\partial \psi^T}{\partial x} d\Omega = \mu[A][C][G][B]^T[A]^T \tag{2.52}$$

Stabilized matrix:

$$Kse(u,v) = \frac{\Delta t}{2}\int_{\Omega^e}\left[uu\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial x} + uv\frac{\partial\psi}{\partial x}\frac{\partial\psi^T}{\partial y} + vu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial x} + uu\frac{\partial\psi}{\partial y}\frac{\partial\psi^T}{\partial y}\right]d\Omega \quad (2.53)$$

$$Kse(u,v) = \frac{\Delta t}{2}\left[uu[A][B][G][B][A] + uv[A][B][G][C][A] + vu[A][C][G][B][A]\right] + [vv[A][C][G][C][A]] \quad (2.54)$$

Stiffness Gradient matrix: $Q_1 = \int_{\Omega^e}\frac{\partial\psi}{\partial x}\phi^T d\Omega$ and $Q_2 = \int_{\Omega^e}\frac{\partial\psi}{\partial y}\phi^T d\Omega$

Substitute the equation (2.43)and (2.44) into the both mixed term of stiffness matrix. Thus, we got

$$Q_1 = [A][B]\int_{\Omega^e}[H][H]^T d\Omega = [A][B][G] \quad (2.55)$$

$$Q_2 = [A][C]\int_{\Omega^e}[H][H]^T d\Omega = [A][C][G] \quad (2.56)$$

Convective matrix: $C(u,v) = \int_{\Omega^e}\rho\left(\psi(\psi^T u)\frac{\partial\psi^T}{\partial x} + \psi(\psi^T v)\frac{\partial\psi^T}{\partial y}\right)d\Omega$

Let us start with substitution the equation (2.43) into the derivative form of x direction

The first term:

$$C_x(u) = \int_{\Omega^e}\rho\psi(\psi^T u)\frac{\partial\psi^T}{\partial x}d\Omega = [A][A]^T[A][B]\int_{\Omega^e}[R][R]^T[u][H]d\Omega \quad (2.57)$$

The second term:

$$C_y(v) = \int_{\Omega^e}\rho\psi(\psi^T v)\frac{\partial\psi^T}{\partial y}d\Omega = [A][A]^T[A][C]\int_{\Omega^e}[R][R]^T[v][H]d\Omega \quad (2.58)$$

Let us apply [F] matrix represent as $F = \int_{\Omega^e}[R][R]^T[u][H]d\Omega$ and $F = \int_{\Omega^e}[R][R]^T[v][H]d\Omega$. The integration of $[F]$ matrix will be derived in APPENDIX B.

From the integration of $[F]$ matrix, we got the final form of convective matrix for derivative form in x-direction as

$$C_x(u) = \frac{2A}{5040}[A][A]^T[A][B][F] \tag{2.59}$$

In similar way, we can get the convective matrix for derivative form in y-direction as

$$C_y(v) = \frac{2A}{5040}[A][A]^T[A][C][F] \tag{2.60}$$

At this time, finite element matrix formulas have already been proved to the final general forms. In order to make more convenient for MATLAB codes construction, we will summarize the general forms of finite element matrix in Appendix B.

CHAPTER 3

FINITE ELEMENT SOLUTIONS AND PROGRAMMING

3.1 Introduction

As we have mentioned in chapter two, the unsteady incompressible Navier-Stokes equation based on artificial compressibility scheme consist of nonlinear term and time dependent problem. Thus, in order to deal with these problems, the solution of iterative methods for nonlinear equation and time-approximation schemes for time dependent problem will be discussed in this chapter.

The first section explain the iterative method of Newton-Raphson and time approximate scheme that can be used to solve the non linear equation and time dependent problem. The derivation of these schemes will be presented in this section. Moreover, this section demonstrates how to select the best time approximate scheme that suitable for the set of equations. For the last two sections, the algorithm of time dependent approach and iterative method will be tested by the benchmark of lid-driven cavity. This problem will be constructed by the finite element programming codes by using MATLAB program. The procedure of finite element programming will be demonstrated and constructed in the last section.

3.2 Solutions to incompressible Navier-Stokes equations

Once the local element matrices are assembled to the form of the system equations and boundary conditions are imposed, the set of equation (2.27) are ready to solve. As we know, the solution of nonlinear incompressible Navier Stokes equations either for steady and unsteady flow is a significant computational challenge.

Thus, before solving these equations, the choice of effective solution algorithms must be considered. From the equation (2.27), these equations can be divided into two parts. The first part is time derivative term and the second part is nonlinear matrix. Thus, we can solve these equations by combining a transient time integration algorithm with the iterative method at each time step [16]. For an iteration scheme, Newton-Raphson method is the most popular and widely used because of its superior convergence properties. Thus, Newton- Raphson method will be selected to solved the part of nonlinear matrix. For the transient time integration schemes, in order to reduce the number of time step, backward Euler scheme will be chosen to deal with the time derivative term.

3.3 Newton- Raphson method

As mentioned before, in order to improve the rate of convergence, Newton Raphson method must be discussed. The concept of this method is to balance the stiffness matrix and load vector. And then, modify the balance set of nonlinear stiff matrix by using derivation form respect to the velocity component and pressure. In order to derive this equation, let us start with the second term of equation (2.27) that is non-linear equations as

$$\begin{bmatrix} C + 2K_{11} + K_{11} + Kse & K_{12} & -Q_1 \\ K_{21} & C + K_{11} + 2K_{11} + Kse & -Q_1 \\ -Q_1^T & -Q_1^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ p \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ 0 \end{bmatrix} \tag{3.1}$$

or

$$[K(u)]\mathbf{U} = \{\mathbf{F}\} \tag{3.2}$$

Let us transform the equation (3.2) into Newton-Raphson form as

$$R(U) = [K(u)]\mathbf{U} - \{\mathbf{F}\} = 0 \tag{3.3}$$

Newton-Raphson method is based on a truncated Taylors series expansion of R(U) about the known solution U^n

$$0 = R(U^n) + \frac{\partial R}{\partial U} \triangle U + 0(\triangle U^2) \tag{3.4}$$

Where $\triangle U = U^{n+1} - U^n$. Omitting the terms of order two and higher, we obtain

$$R(U^n) = -\frac{\partial R}{\partial U}(U^{n+1} - U^n) \cong -J(U^n)(U^{n+1} - U^n) \tag{3.5}$$

Where J is the Jacobean matrix

$$J = -\frac{\partial R}{\partial U} \tag{3.6}$$

By solving equation (3.4), we got

$$U^{n+1} = U^n - J^{-1}(U^n)R(U^n) \tag{3.7}$$

Let the equation (3.1) apply to (3.3), we got

$$\begin{aligned}
R1 &= C_1(u)u + C_2(v)u + (2K_{11} + K_{22})u + K_{12}v - Q_1p - F_1 \\
&\quad + \frac{\Delta t}{2}\left[Ks_{11}(uu)u + Ks_{12}(uv)u + Ks_{21}(vu)u + Ks_{22}(vv)u\right] \\
R2 &= C_1(u)u + C_2(v)u + (2K_{11} + K_{22})u + K_{12}v - Q_1p - F_2 \\
&\quad + \frac{\Delta t}{2}\left[Ks_{11}(uu)v + Ks_{12}(uv)v + Ks_{21}(vu)v + Ks_{22}(vv)v\right] \\
R3 &= -Q_1^T u - Q_2^T v
\end{aligned} \tag{3.8}$$

To evaluate the Jacobian matrix, substitute (3.8) into (3.6),We got the derivative of stiffness matrix in the Newton-Raphson form

$$J = -\frac{\partial R}{\partial U} = -\begin{bmatrix} \frac{\partial R1}{\partial u} & \frac{\partial R1}{\partial v} & \frac{\partial R1}{\partial p} \\ \frac{\partial R2}{\partial u} & \frac{\partial R2}{\partial v} & \frac{\partial R2}{\partial p} \\ \frac{\partial R3}{\partial u} & \frac{\partial R3}{\partial v} & \frac{\partial R3}{\partial p} \end{bmatrix} \tag{3.9}$$

Where

$$
\begin{aligned}
\frac{\partial R1}{\partial u} &= C_1(u) + C_2(1)u + C_2(v) + 2K_{11} + K_{22} \\
&\quad + \frac{\Delta t}{2}\left[Ks_{11}(uu) + Ks_{11}(2u)u + Ks_{12}(uv)\right] \\
&\quad + \frac{\Delta t}{2}\left[Ks_{12}(v)u + Ks_{21}(vu) + Ks_{21}(v)u + Ks_{22}(vv)\right] \\
\frac{\partial R1}{\partial v} &= C_2(1)u + K_{12} + \frac{\Delta t}{2}\left[Ks_{12}(u)u + Ks_{21}(u)u + Ks_{22}(2v)u\right] \\
\frac{\partial R1}{\partial p} &= -Q_1 \\
\frac{\partial R2}{\partial u} &= C_1(1)v + K_{21} + \frac{\Delta t}{2}\left[Ks_{12}(v)v + Ks_{21}(v)v + Ks_{22}(2u)v\right] \\
\frac{\partial R2}{\partial v} &= C_1(u) + C_2(v) + C_2(1) + K_{11} + 2K_{22} \\
&\quad + \frac{\Delta t}{2}\left[Ks_{11}(uu) + Ks_{12}(uv) + Ks_{12}(u)v\right] \\
&\quad + \frac{\Delta t}{2}\left[Ks_{21}(vu) + Ks_{21}(u)v + Ks_{22}(vv) + Ks_{22}(2v)v\right] \\
\frac{\partial R2}{\partial p} &= -Q_2 \\
\frac{\partial R3}{\partial u} &= -Q_1^T \\
\frac{\partial R3}{\partial v} &= -Q_2^T \\
\frac{\partial R3}{\partial p} &= 0
\end{aligned}
\tag{3.10}
$$

Substitute the Jacobian matrix (3.10) into (3.5), we got

$$
R(U^n) = -\begin{bmatrix} \frac{\partial R1}{\partial u} & \frac{\partial R1}{\partial v} & \frac{\partial R1}{\partial p} \\ \frac{\partial R2}{\partial u} & \frac{\partial R2}{\partial v} & \frac{\partial R2}{\partial p} \\ \frac{\partial R3}{\partial u} & \frac{\partial R3}{\partial v} & \frac{\partial R3}{\partial p} \end{bmatrix} (U^{n+1} - U^n) \tag{3.11}
$$

Thus, the equation (3.1) can be replaced by

$$
\begin{bmatrix} \frac{\partial R1}{\partial u} & \frac{\partial R1}{\partial v} & \frac{\partial R1}{\partial p} \\ \frac{\partial R2}{\partial u} & \frac{\partial R2}{\partial v} & \frac{\partial R2}{\partial p} \\ \frac{\partial R3}{\partial u} & \frac{\partial R3}{\partial v} & \frac{\partial R3}{\partial p} \end{bmatrix} \begin{bmatrix} \triangle u^{n+1} \\ \triangle v^{n+1} \\ \triangle p^{n+1} \end{bmatrix} = -\begin{bmatrix} R1 \\ R2 \\ R3 \end{bmatrix} \tag{3.12}
$$

where

$$\begin{bmatrix} \triangle u^{n+1} \\ \triangle v^{n+1} \\ \triangle p^{n+1} \end{bmatrix} = \begin{bmatrix} u^{n+1} - u^n \\ v^{n+1} - v^n \\ p^{n+1} - p^n \end{bmatrix} \tag{3.13}$$

Once we got the set of Newton-Raphson equation (3.12), this equation will be substituted into the original equation (2.27). As the result, the equation (2.27) will be replaced by equation (3.14) as

$$\begin{bmatrix} [M] & 0 & 0 \\ 0 & [M] & 0 \\ 0 & 0 & -[M_p] \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{p} \end{bmatrix} + \begin{bmatrix} \frac{\partial R1}{\partial u} & \frac{\partial R1}{\partial v} & \frac{\partial R1}{\partial p} \\ \frac{\partial R2}{\partial u} & \frac{\partial R2}{\partial v} & \frac{\partial R2}{\partial p} \\ \frac{\partial R3}{\partial u} & \frac{\partial R3}{\partial v} & \frac{\partial R3}{\partial p} \end{bmatrix} \begin{bmatrix} \triangle u^{n+1} \\ \triangle v^{n+1} \\ \triangle p^{n+1} \end{bmatrix} = - \begin{bmatrix} R1 \\ R2 \\ R3 \end{bmatrix} \tag{3.14}$$

or

$$[M]\dot{\mathbf{U}} + [K(\mathbf{U})]\triangle\mathbf{U} = \{\mathbf{R}\} \tag{3.15}$$

Let us called this equation as AC Newton Raphson Equation

3.4 Backward Euler Time-Approximation scheme

From the AC Newton Raphson Equation (3.15), it can be seen that this equation represents a discrete space and continuous time approximation to the original system of partial differential equations. In order to solve the time dependent problem, the choice of time approximation schemes must be considered. There are three types of time dependent approach: Explicit, Implicit, and Semi- implicit integration methods. These methods have many different advantages and disadvantages. As the reference [15], from the comparing with these methods, Implicit Euler Scheme (Backward Euler finite difference method) is the best choice that suitable for the solution of AC Newton Raphson Equation because this method is unconditional stability and save more CPU time to iterate. However, this scheme makes more computationally expensive than the others.

In addition, although the implicit Euler scheme make more computationally expensive than the others, this scheme are desirable due to their increased stability and the consistent treatment of the pressure. Moreover, number of the time steps can be reduced by this scheme. However, although this scheme is unconditional stability, the time step size of this method must be concerned because of the nonlinearity of AC Newton Raphson equation can make the unpredicted oscillation value.

To derive the solution of time approximation method by using Implicit method, let us start with the time derivative term of AC Newton Raphson Equation. This term can be transformed to the new form by using Taylor series expansion as expressed in (3.16).

$$\dot{\mathbf{U}} = \frac{\partial U}{\partial t} = \frac{U^{n+1} - U^n}{\triangle t} \tag{3.16}$$

Substitute (3.16) into (3.15), we got

$$[M]U^{n+1} = \triangle t[R(U^{n+1}) - K(U^{n+1})(U^{n+1})] + [M]U^n \tag{3.17}$$

Or in a form more suitable for computation

$$[M + \triangle tK(U^{n+1})]U^{n+1} = \triangle t[R(U^{n+1}) - [M]U^n \tag{3.18}$$

As we have derived previously, it can be seen that the Newton Raphson method and Backward Euler scheme are employed to modify the equation(2.25) until we got the new equation as expressed by (3.18). This equation represents the final form of two dimensional unsteady incompressible Navier Stokes equation based on AC method by using the solution of Newton Raphson method for nonlinear term and Backward Euler scheme for time dependent matrix. Thus, let us call the equation of (3.18) as AC Backward Newton Raphson Equation.

Once we got the final form of AC Backward Newton Raphson Equation, this equation will be verified by the benchmark of lid-driven cavity problem that is the

standard benchmark test for incompressible flows. The application problem of cavity flow will be discussed in next section.

3.5 The Benchmark of Lid-driven cavity

3.5.1 Boundary conditions

As we mentioned previously, the flow in a lid-driven cavity is one of the most widely use benchmark problems to test the incompressible Navier Stokes codes. This example become a standard benchmark test for incompresiible flows. We will show the procedure how to set the boundary condition for the physical of incompressible Navier Stokes problem. The boundary conditions are indicated in figure 3.1. There are various ways to assume the boundary conditions but the most common use is one in which the velocity along the top surface increase from the left corner node to the driven value in the length of element (ramp condition)[14]. The fixed velocity is located at along the bottom wall and along the both of vertical wall. As the figure 3.1, it is obvious that Dirichlet boundary conditions are imposed on every boundary in this example. Thus, a singulary in the pressure should be located at the lower left corner of the cavity, the reference value p=1 is prescribed [14].

3.5.2 Meshing

To demonstrate the influence of mesh density on the solution procedure, four different meshes are selected for this problem. These meshes consist of 25x25 nodes structure mesh, 25x25 nodes new refine mesh, 41x41 nodes structure mesh, and 41x41 nodes new refined mesh. Both of the new refined mesh 25x25 and 41x41nodes are generated by refining the mesh at the corner and coarsening the mesh at the center of cavity. These meshes are shown in figure 3.2.

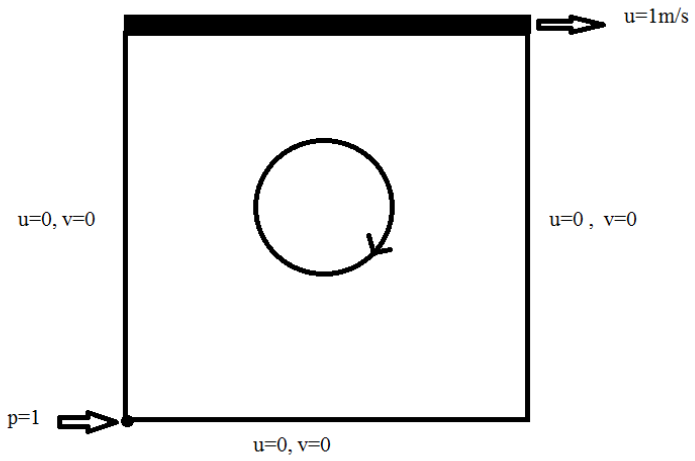

Figure 3.1. Boundary conditions of the benchmark of lid-driven cavity.

Once the boundary conditions have already set, we will solve the equation (3.18) by using the standard Galerkin formulation. The numerical result of steady state solution will be obtained when the calculation is continued until the maximum difference of the variables u, v, and p between two consecutive time steps became less than 10^{-6}. For the formula of steady state checking, we will discuss in the next section.

To verify the numerical of steady state solutions, the standard value from this problem need to be employed for comparing. As we know, this problem have been studied by many researchers who quoted many solutions and data result for different number of Reynolds number. The data set of those Ghia et al.[12] is one of the standard value that we want to select for comparison. In this thesis, the Reynolds numbers of 100, 400, 1,000, and 3,200 are selected as the initial input data. The Reynolds numbers can be expressed by the form

$$Re = \frac{\rho U L}{\mu} \tag{3.19}$$

Where ρ, U, μ are density, velocity, and viscosity, respectively. L is the unit length of cavity. For the numerical results of steady state solution at different Reynolds number, the velocity distributions along the centre-line from these Reynolds numbers will be compared to those of Ghia value. Moreover, streamline plots and pressure contours will be compared by using different meshes. The part of result will be discussed in the next chapter.

3.6 Finite Element Programming

When the derivation of AC Backward Newton Raphson equations are completed and the boundary condition have already set, the finite element programming codes will be constructed to solve these equations. These programming codes are written by MATLAB program. In order to construct the programming codes easily, the procedure of finite element programming will be explained as step by step.

3.6.1 The procedures of finite element programming

The finite element program consist of one main program and three sub programs (three sub functions). Let us call this program as Complete Incompressible Navier Stoke. The detail of this program is available in Appendix C. Howver, in order to make more understanding with finite element programming, we can summarize the procedure into 7 steps as following.

Step 1. For the first time step, the program start with the input data of the lid-cavity driven problem such as initial guess value, number of nodes, elements, iterations, time steps, coordinates, and boundary conditions. These properties will be set as the numbering of nodes.

Step 2. The input data will be constructed as the local element matrices that have the numbering at each elements. Once the construction of local element matrices

(mass matrix, convective matrix, stiffness matrix, and load vector) are completed, the subfunction programming will be called to assemble these matrices to Global matrix that is the system of equations. This function program is called TRI1.

Step 3. Once the assembling matrices are completed, the boundary conditions must be applied to the global system. In this step, the boundary subfunction programming will be called to apply the boundary value at some nodes in the problem. The boundary programming is called APPLYBC1.

Step 4. In this step, the global system will be solved when the boundary conditions have already applied to the equation. For solving part, Gauss eliminate sub function programming will be called to solve the set of equations system. As the result, velocity components and pressure variables will be obtained for the first iteration at this step. Step 5. Once we got the numerical results for the first iteration from step 4, these results will be sent to test the iteration convergence by the convergence test. The equation of convergence test can be expressed as

$$\varepsilon = \sum_{i=1}^{n} \left| \frac{U^{n+1} - U^{n}}{\Delta t} \right| \tag{3.20}$$

If the value of equation (3.20) is greater than 0.01, the result from this step will be sent to compute again in step 2 for the next iteration until the result of last iteration is less than 0.01. Once the result is satisfy the convergence condition, these values will be set as the initial value for the next time step. Thus, the procedure of first step will be restarted again. As we mentioned in chapter 2, the artificial compressibility will be vanished when the steady state is reached. Thus, the artificial compressibility term must be checked by the pressure convergence test as

$$\varepsilon = \frac{1}{\beta} \sum_{i=1}^{n} \left| \frac{p^{n+1} - p^{n}}{\Delta t} \right| \tag{3.21}$$

We will assume the value of pressure convergence test as 10^{-6} . If the result for the last iteration is satisfied pressure convergence test (3.21), the process for the next

time step will be stopped. As a result, the final value of velocity components and pressure are obtained. In order to make more clear in the process of the finite element programming procedure, these steps will be shown in the chart 3.1.

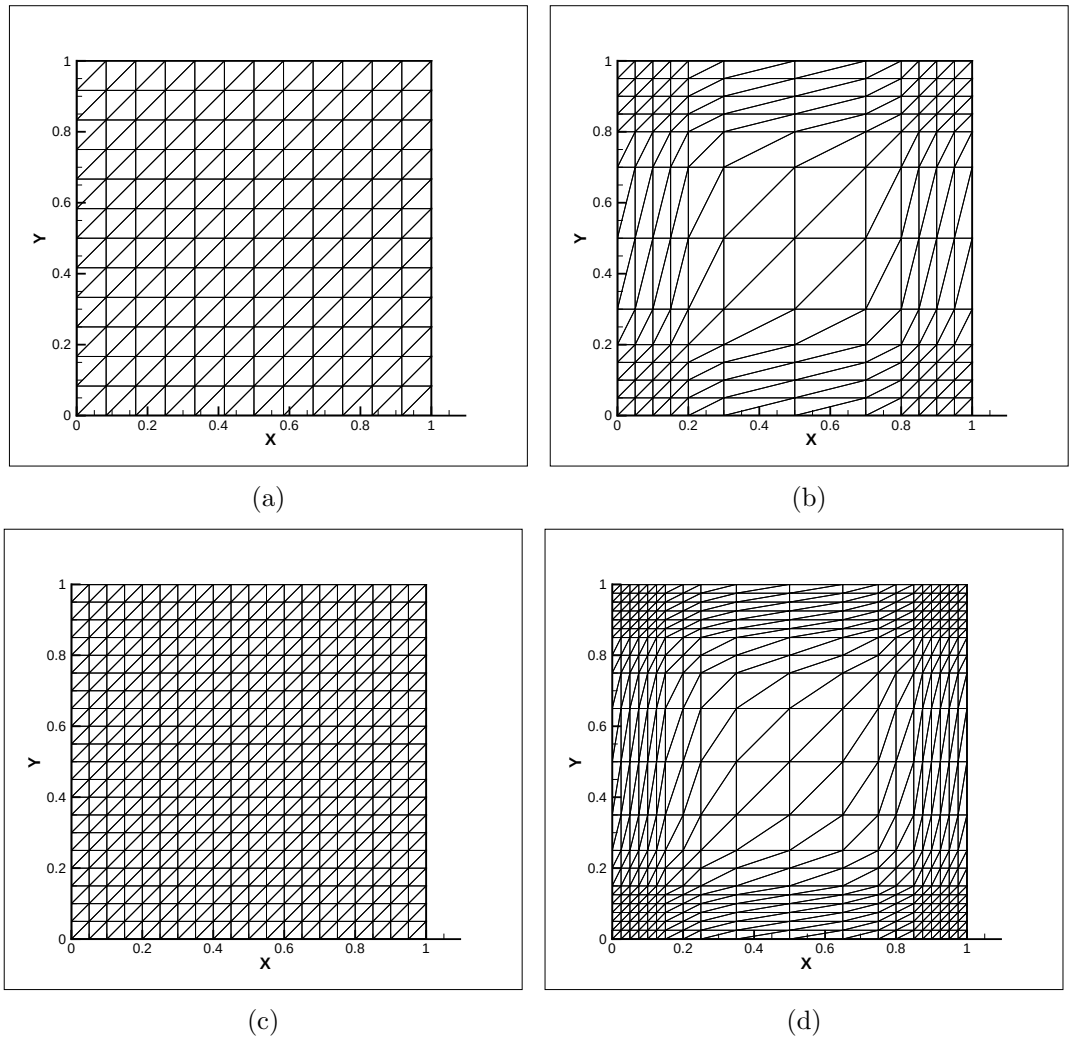

Figure 3.2. Four different quadratic triangular mesh of (a) Mesh1: 25x25 nodes, (b) Mesh2: New refined 25x25nodes (c) Mesh3: 41x41 nodes and (d)Mesh4: New refined 41x41nodes.

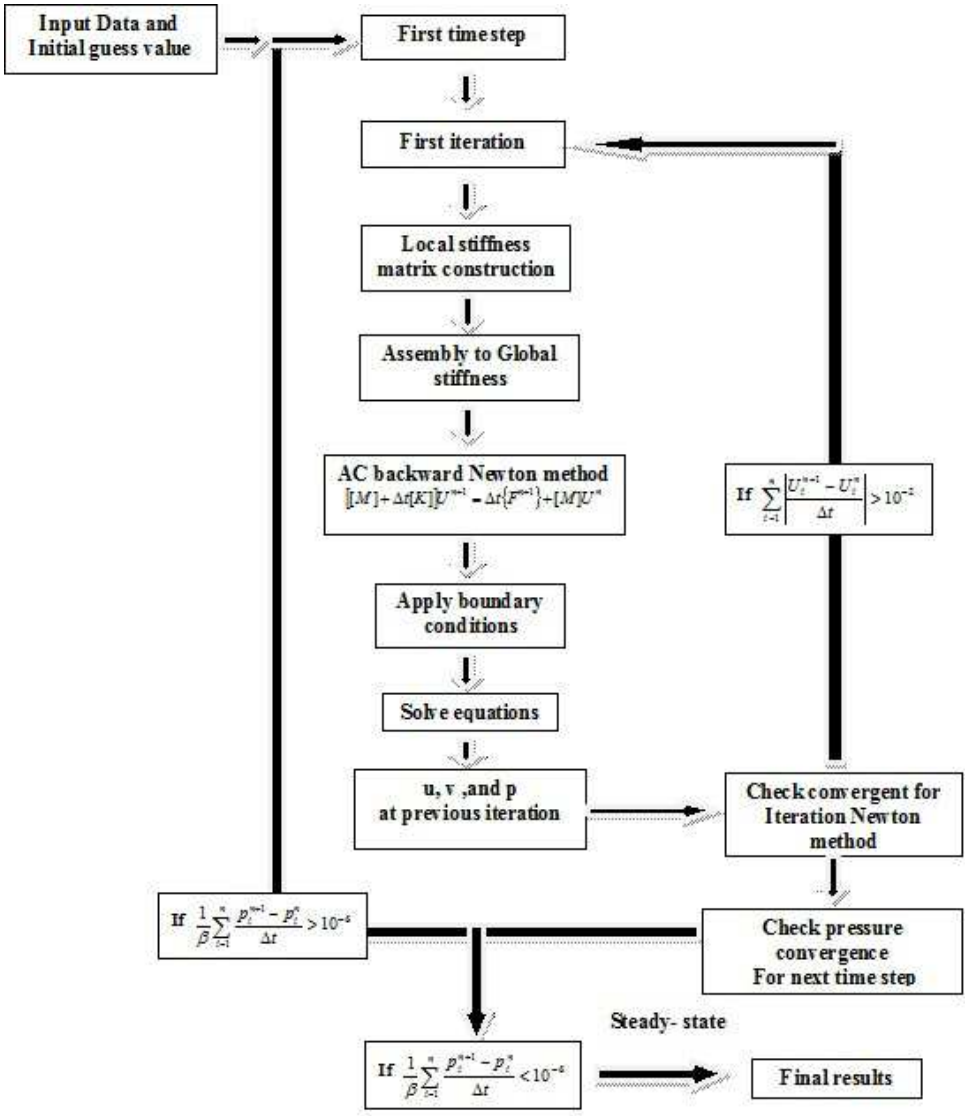

Figure 3.3. The procedure of finite element programming.

CHAPTER 4

NUMERICAL RESULTS AND DISCUSSIONS

4.1 Introduction

Once the finite element programming codes in the previous chapter have already constructed and tested by the benchmark of lid-driven cavity, the numerical results will be presented and analyzed in this chapter. As mentioned before in previous chapter, in order to verify the finite element programming codes, the numerical results from finite element programming will be compared with the numerical results obtained from finite element analysis done by Ghia et al.[12]. The velocity stream line and contour plots at the different Reynolds numbers and different meshes will be compared together in order to demonstrate the influence of mesh density on the solution procedures. However, these results must be satisfied by the condition of steady-state convergence checking in the finite element programming.

4.2 Comparing the velocity streamlines plot and pressure contours

As mentioned in section 3.4, before using the numerical results to present, these results must be satisfied by the convergence checking as expressed in the equation 3.21. To demonstrate the numerical results that are satisfied by this test, in this section, the steady-state convergence history of different meshes (refined mesh 2 and mesh 4) at different Reynolds number of 100, 400, 1000, and 3200 are presented as shown in figure 4.1-4.2

From the figure 4.1-4.2, it can be seen that the convergence history depend on the time step size, different meshes, and different Reynolds number.

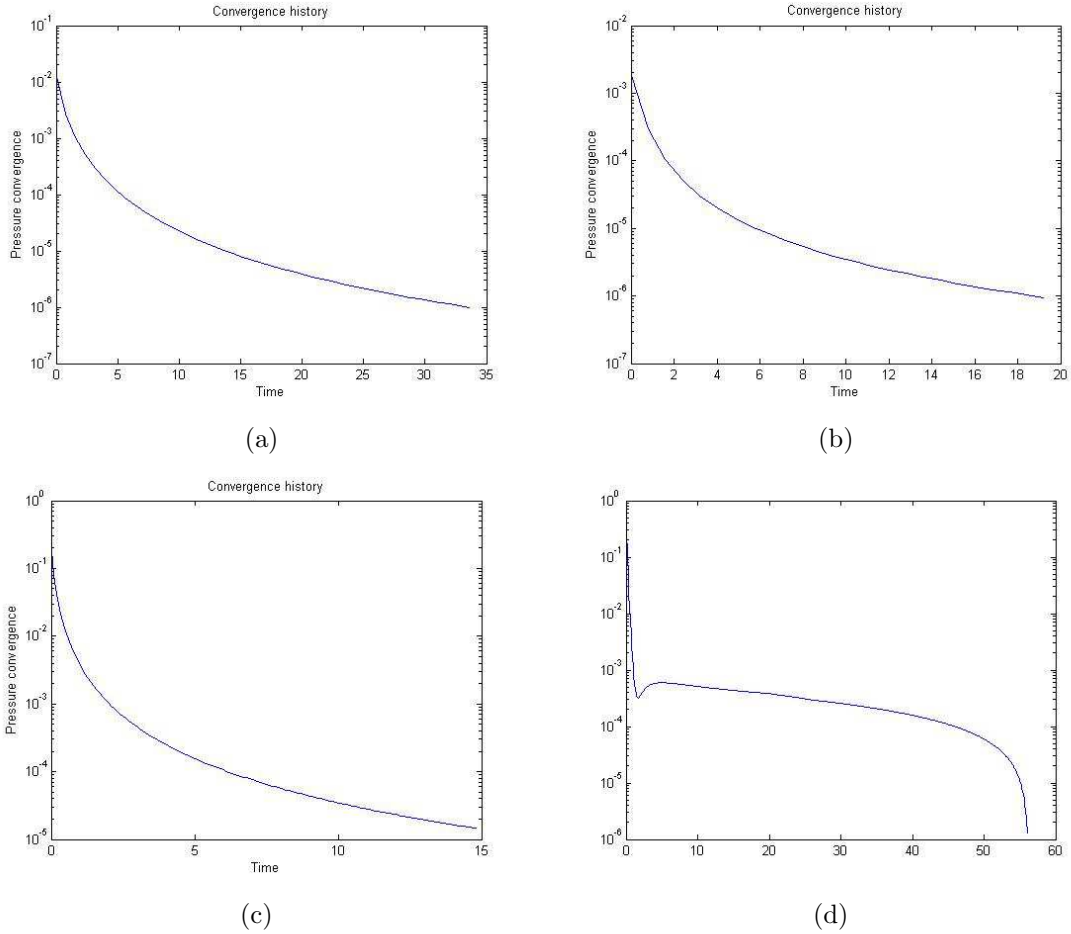

Figure 4.1. The convergence history graph at different Reynolds number (a)Re 100: Δt=0.8, 580 iterations, 35 time steps, (b)Re 400: Δt=0.8, 890 iterations, 20 time steps,(c)Re 1000: Δt=0.1, 1050 iterations, 15 time steps and (d)Re 3200: Δt=0.1, 3000 iterations, 56 time steps by using New mesh 25x25 nodes.

Once the steady state in the solutions is reached, the numerical results will be obtained. As mentioned in previous chapter, four different meshes: 25x25 nodes structure mesh, 25x25 nodes new refined mesh, 41x41 nodes structure mesh, and 41x41 nodes new refined mesh are employed to apply in the finite element solution in order to compare the behavior of velocity components and pressure that occurred in the cavity. For the general cases, the velocity streamlines plot and pressure contours can be the best graph that clearly demonstrate the influence of these different meshes

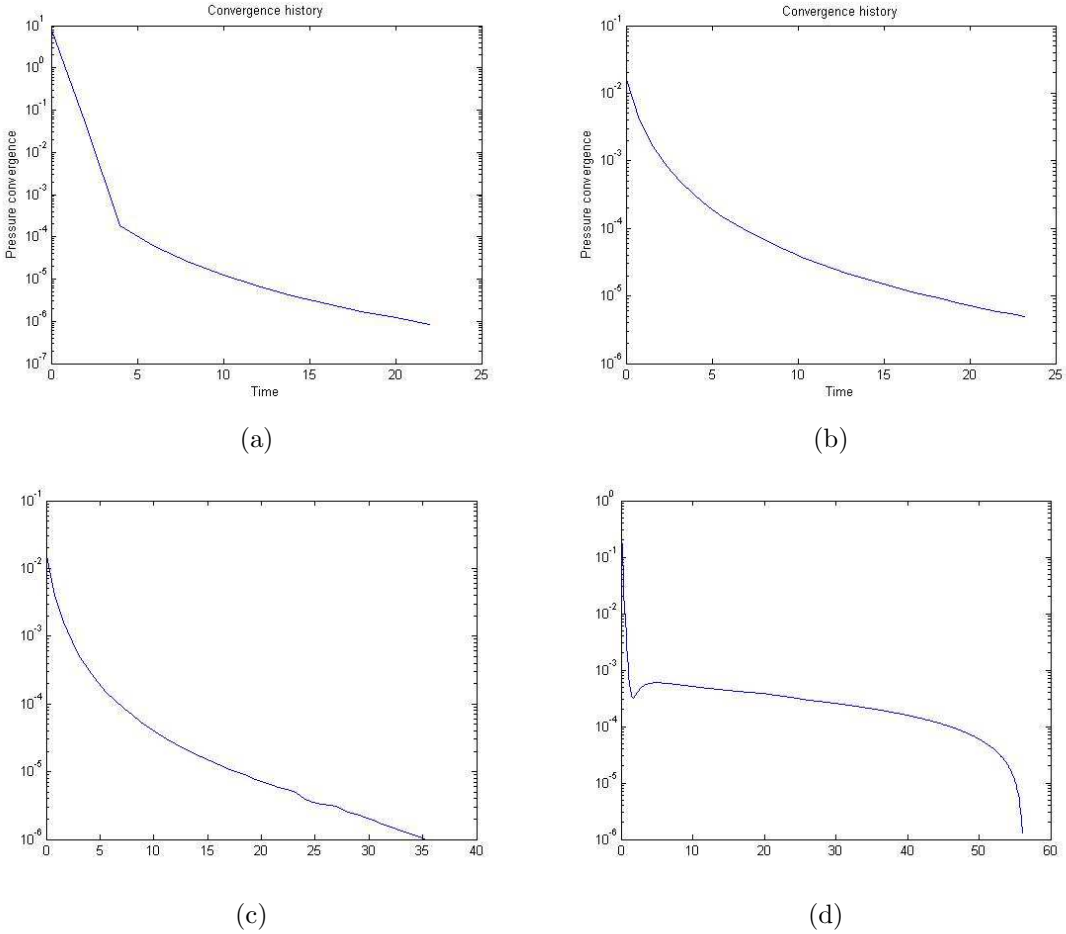

Figure 4.2. The convergence history graph at different Reynolds number (a)Re 100: Δt=0.8, 1050 iterations, 25 time steps, (b)Re 400: Δt=0.8, 2000 iterations, 25 time steps, Re 1000: Δt=0.05, 2705 iterations, 40 time stepsand (d)Re 3200: Δt=0.05, 3500 iterations, 40 time steps by using New mesh 41x41 nodes.

on the numerical solutions. Thus, the numerical results will be presented here in the term of velocity streamlines plot and pressure contour at different Reynolds numbers of 100, 400, 1,000, and 3,200 as shown in figure 4.3 - 4.10.

From figure 4.3-4.10, it can be seen that the numerical results in terms of the streamlines plot and pressure contour graph by using the last two meshes(41x41 and 41x41 nodes new mesh) are smoother contour than the first two meshes(25x25 and 25x25 new mesh). However,from comparing the streamline plot by using the same

number of nodes, it is obvious that the new refined meshes can generate the second and third smooth vortex at the bottom right-hand corner and left-hand corner. And these vortexes will become larger when the Reynolds number is increased to Re 400, 1,000, and 3,200 respectively. From the velocity streamlines plot at high Reynolds number of 3200, it can be seen that mesh 4 can generate the forth vortex appears at the top left corner of the cavity while the others are failed to generate this occurrence because of insufficient mesh resolution. In the general for this problem, if we can generate the smooth vortex at the left and right bottom corner, some oscillations in the process can be reduced. Thus, the very fine meshes near the wall are required if we want to achieve the good results. These ideas can be demonstrated by the comparison between the present results and the numerical results from those of Ghia et al. [12] in the term of vertical and horizontal velocity distributions at the center-line of cavity by using different meshes at the same Reynolds number. The comparing of these numerical results will be presented in next section.

From the comparing pressure contour at Re 100, it can be seen that some oscillations appear at the top left and right corner of cavity. And these oscillations disappear become larger when the Reynolds number is increased to Re 400, 1000, and 3200 respectively. However, these oscillations will be reduced by using mesh 4. Thus, from the numerical results in term of velocity streamlines plot and pressure contour, we can predict the very fined near the wall or mesh 4 can give numerical results better accuracy than that used for mesh 1, mesh 2, and mesh 3.

4.3 Comparing the velocity distributions at central line

In order to verify the finite element codes, the comparison between present results and other numerical data obtained by Ghia et al. will be presented in the term of velocity distributions at vertical and horizontal central line by using differ-

ent meshes. Both of horizontal and vertical velocity distributions are presented at Reynolds number of 100, 400, 1,000, and 3,200 as shown in figure 4.11 - 4.18.

From the comparisons in figure 4.11-4.18, it can be seen that the numerical results by using different four meshes at the low Reynolds number of Re 100 are very close to the numerical results of Ghia who use very fine mesh 121x121 to the solutions. Thus, these present numerical results show excellent agreement when compared with Ghia. When the Reynolds number is increased to Re 400 ,and 1,000 respectively, numerical oscillations due to the convection dominated encounter the solutions. As a result, some error occur in the solutions. However, although the inaccurate result will be occur when the Reynolds number is increased, both of the new refined mesh 25x25 nodes (mesh 2) and 41x41 nodes (mesh 4) can make the result more accuracy than both of structure meshes when compared with those of Ghia value. Therefore, in order to compute the accurate result, the very fine mesh near the wall of cavity must be required.

As we have mentioned in chapter 2, the main difficulty of high Reynolds number of Re 3200 which is due to the presence of nonlinear and non-sysmetric convective terms in the momentum equation encountered in the numerical solutions. When the Reynolds number is increased (more than Re 1000), the velocity gradients will be developed near the cavity walls. This behavior will generates the non-physical oscillations in the Galerkin finite element solution for the velocity. For this reason, the mesh 4 or the refined mesh near the wall is not enough to deal with this difficulty. In order to deal with this difficulty, the very fine meshes near the wall need to be required. However, although the difficulty of high Reynolds number can be dealt with by using these meshes, the number of equations based on Galerkin finite element method becomes larger. As a result, the finite element programming makes more CPU time to iterate the system of equations. Thus, as mentioned in chapter 2, to avoid

using these meshes, the Taylor Galerkin Stabilized technique is the main technique to deal with this problem. From the numerical results at Re 3200 by using different meshes as shown in figure 4.6, it can be seen that the Taylor Galerkin stabilized formulation can be used to solve this difficulty. It is also found that this method can save more CPU time to iterate the system of equations as can be seen in the convergent history graph.

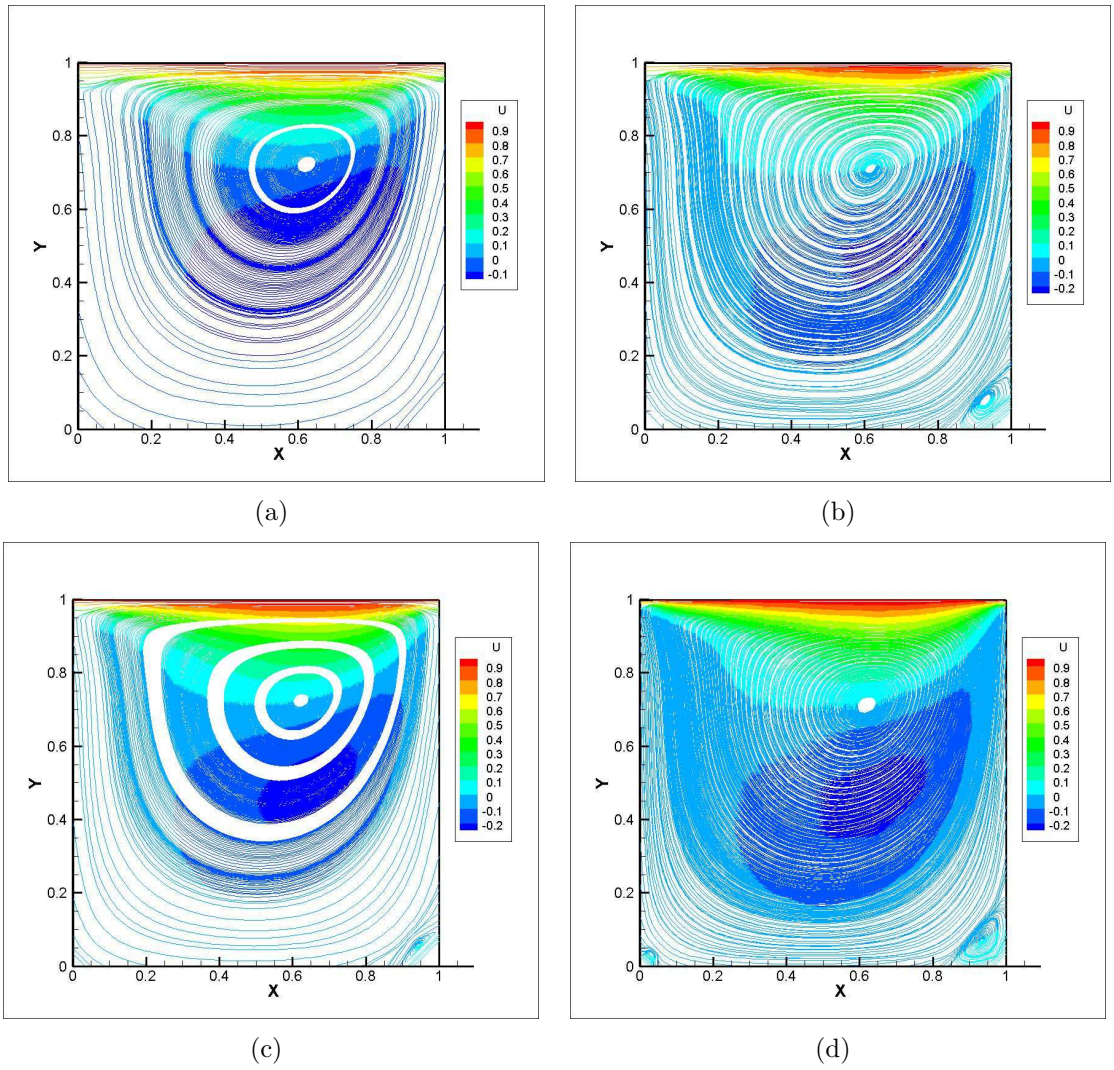

Figure 4.3. The velocity streamlines plot at Re 100 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

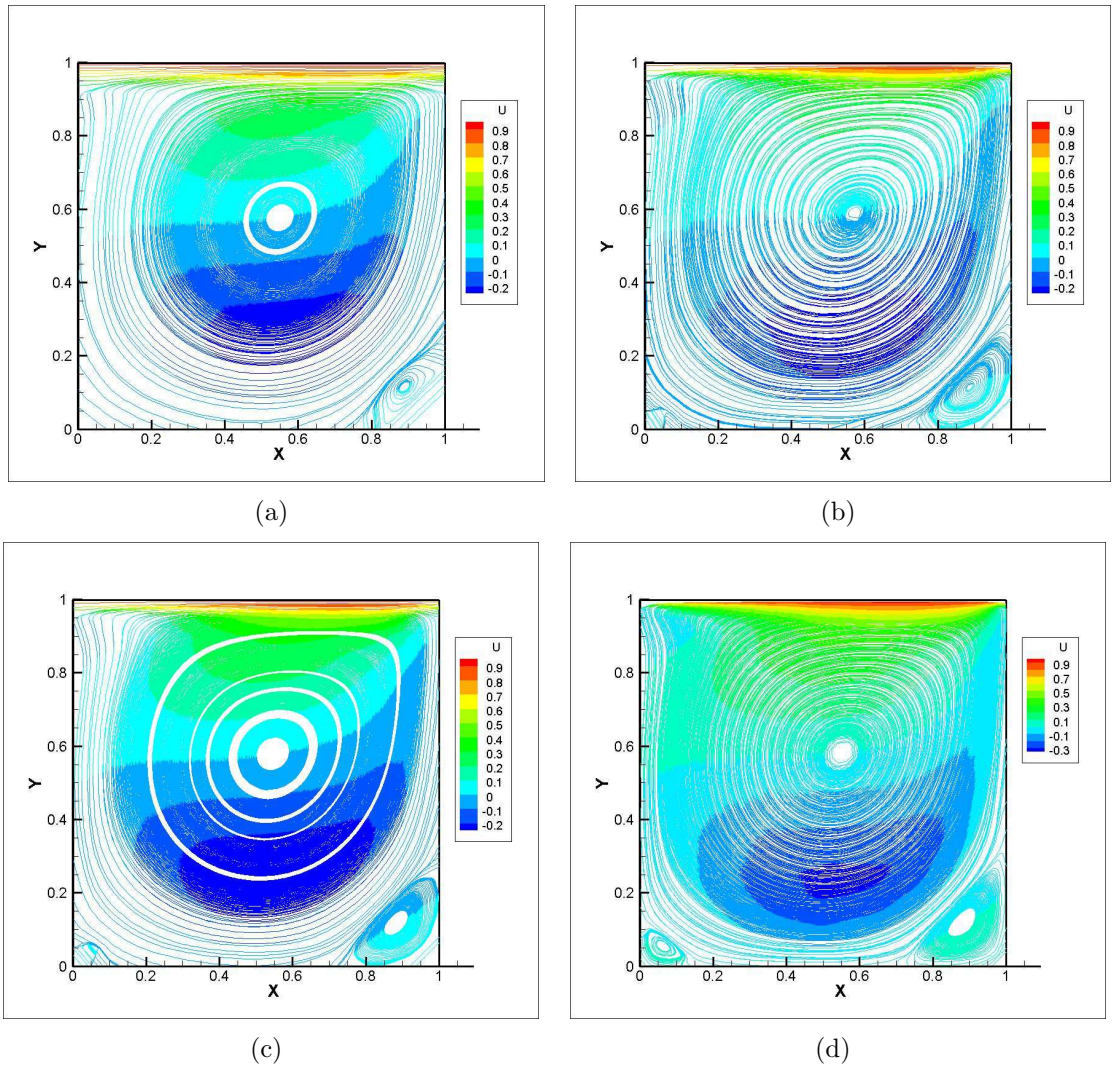

Figure 4.4. The velocity streamlines plot at Re 400 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

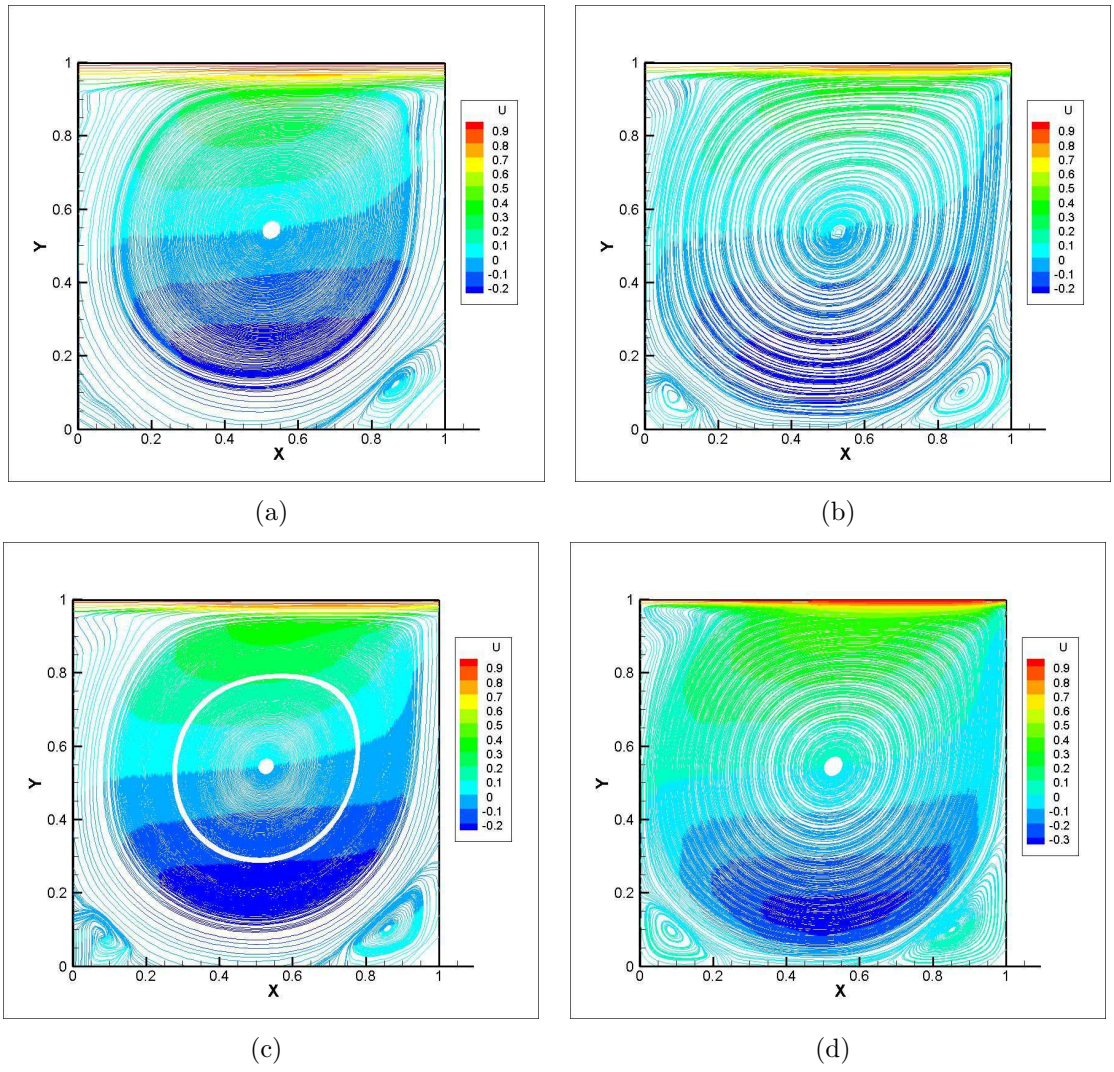

Figure 4.5. The velocity streamlines plot at Re 1000 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

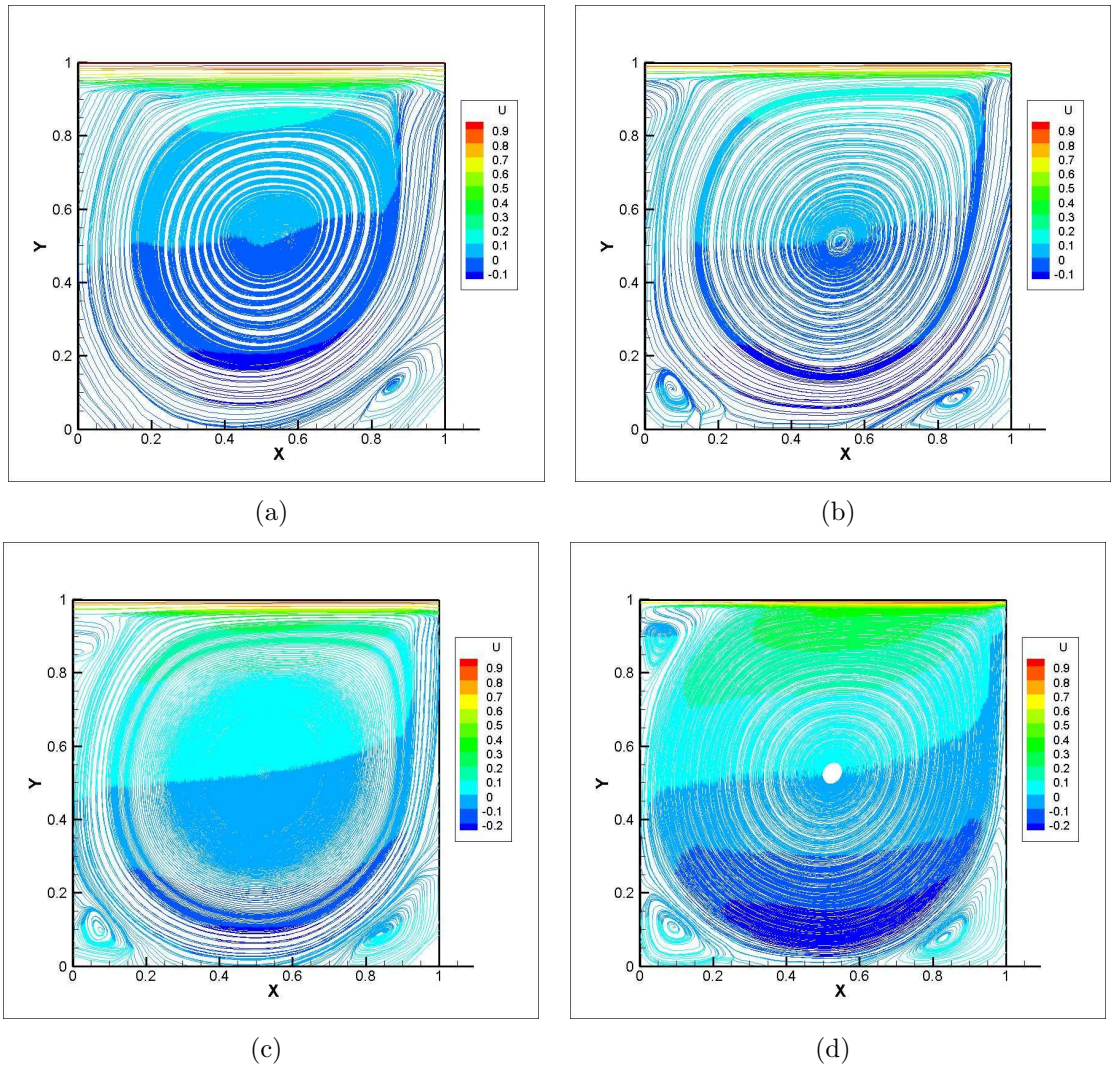

Figure 4.6. The velocity streamlines plot at Re 3200 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

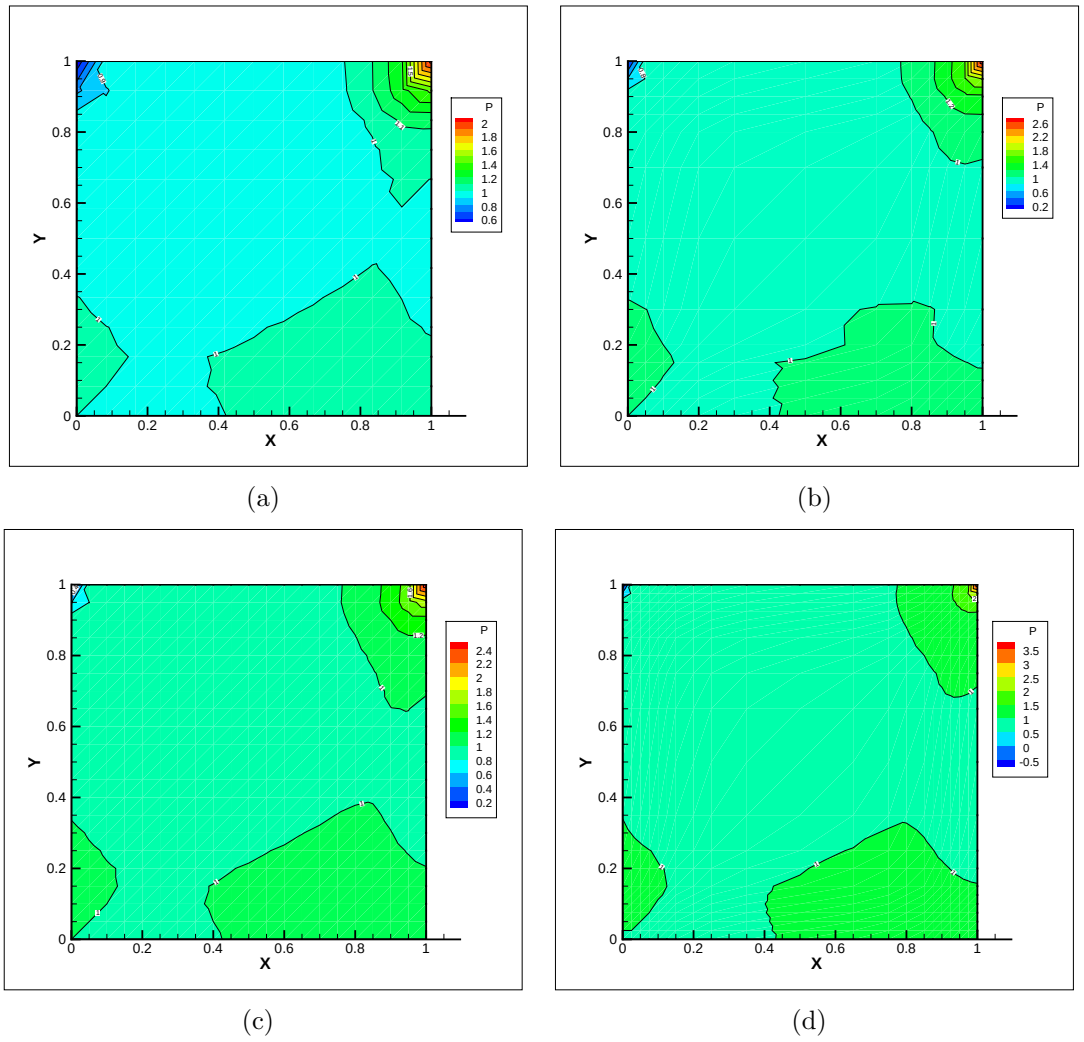

Figure 4.7. The pressure contours at Re 100 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

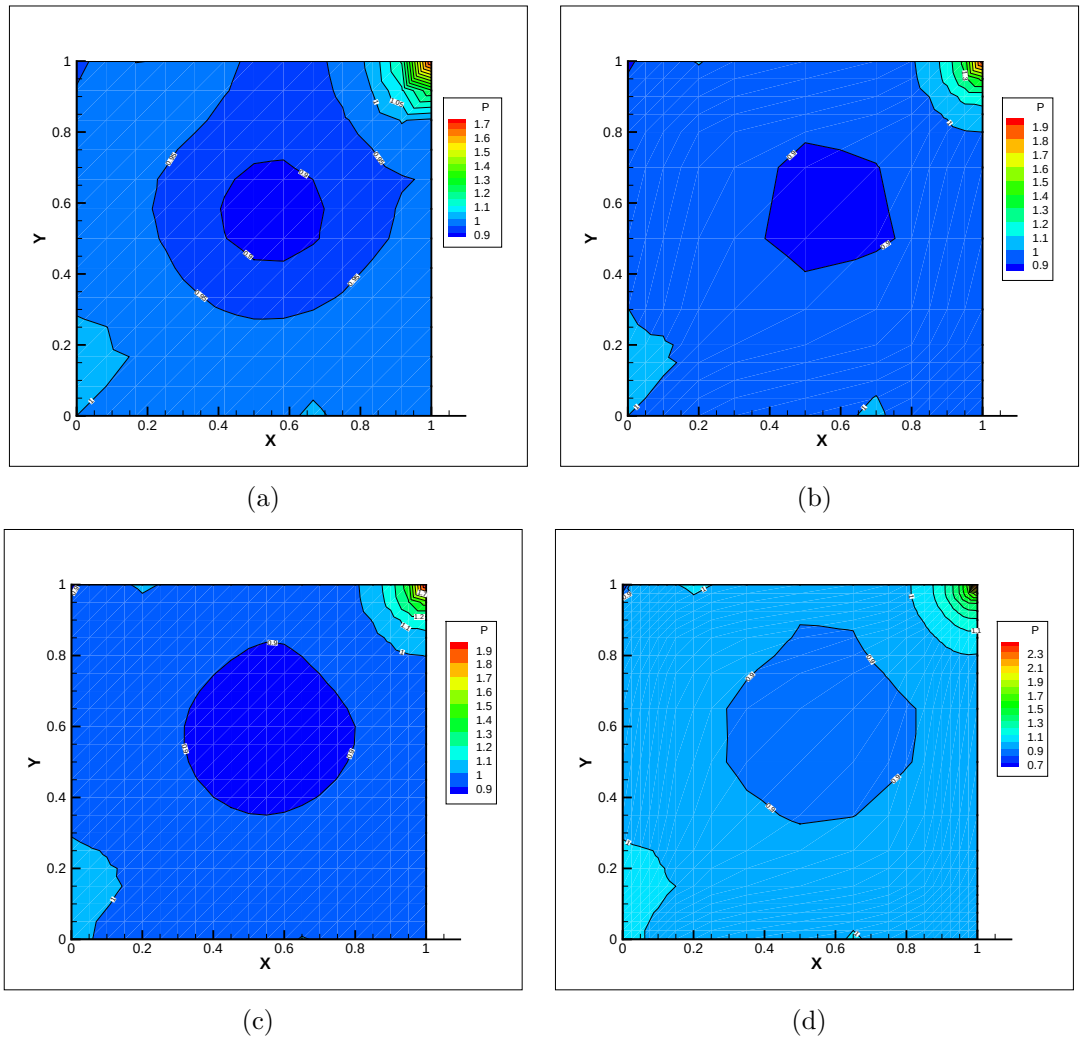

Figure 4.8. The pressure contours at Re 400 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

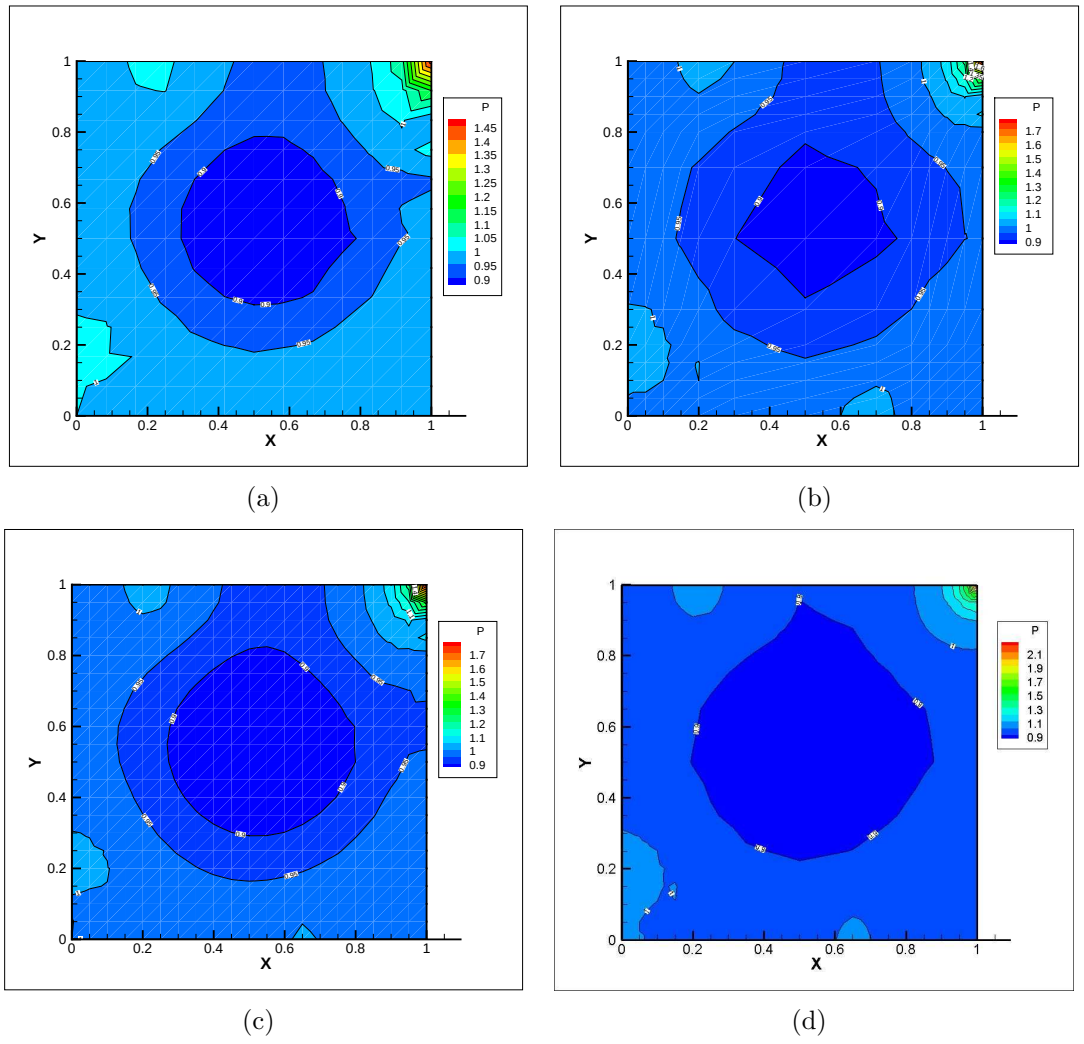

Figure 4.9. The pressure contours at Re 1000 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

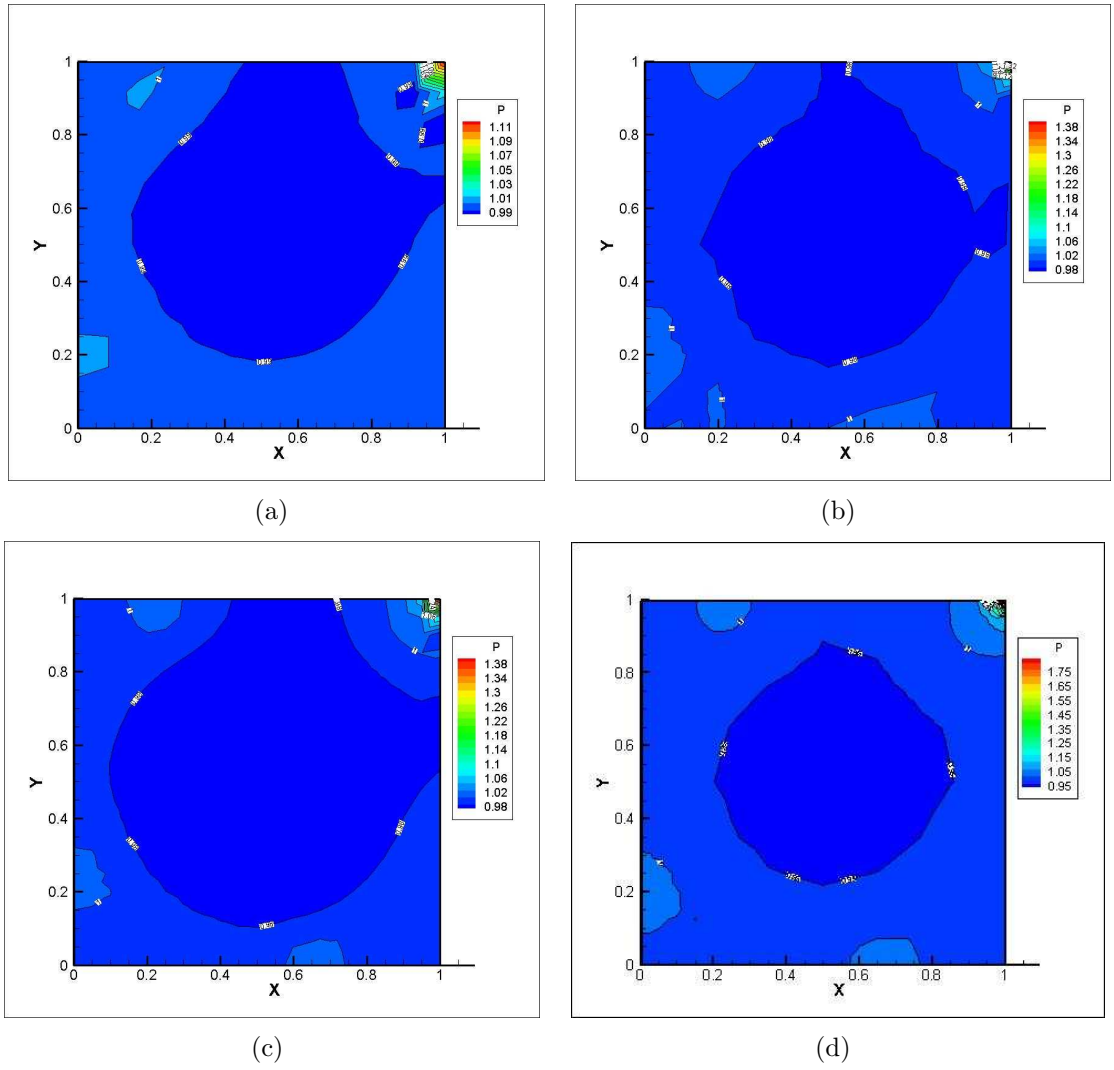

Figure 4.10. The pressure contours at Re 3200 by using different meshes of (a)Mesh1, (b)Mesh2, (c)Mesh3, and (d)Mesh4.

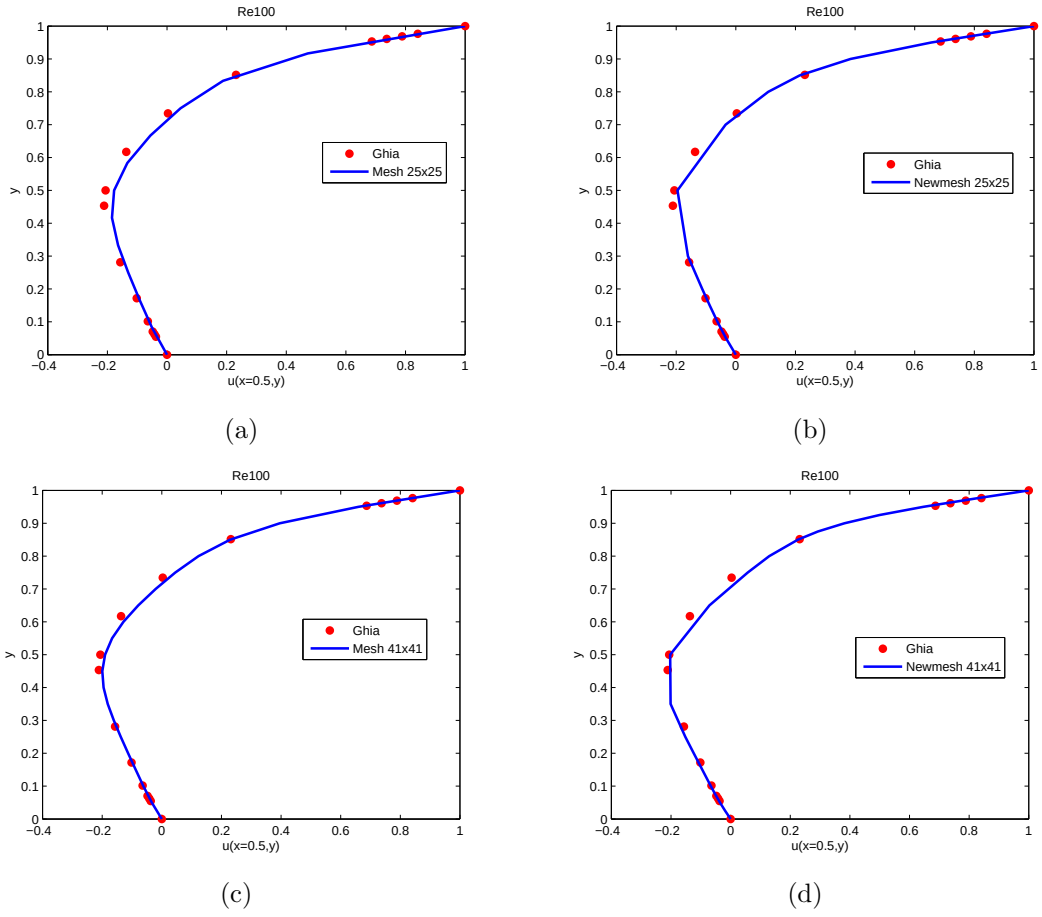

Figure 4.11. The comparison of horizontal velocity distributions at Re 100 with Ghia et al. by using different meshes.

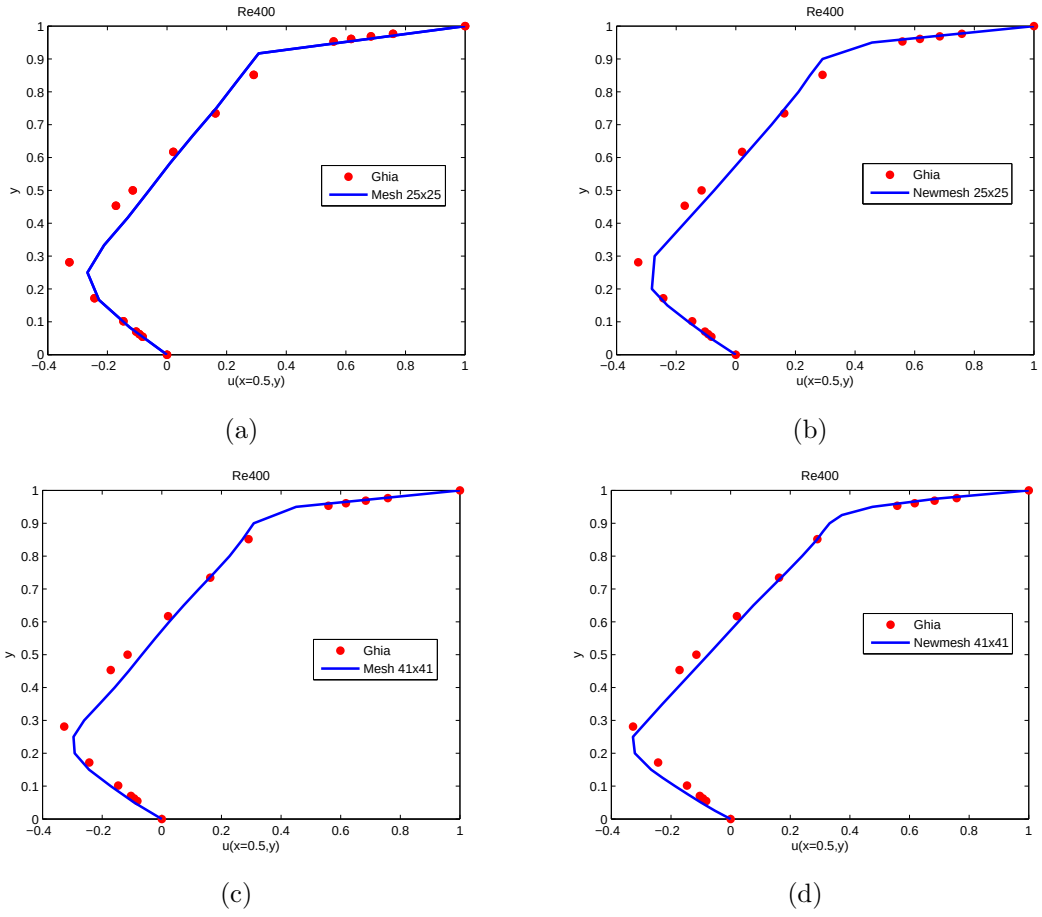

Figure 4.12. The comparison of horizontal velocity distributions at Re 400 with Ghia et al. by using different meshes.

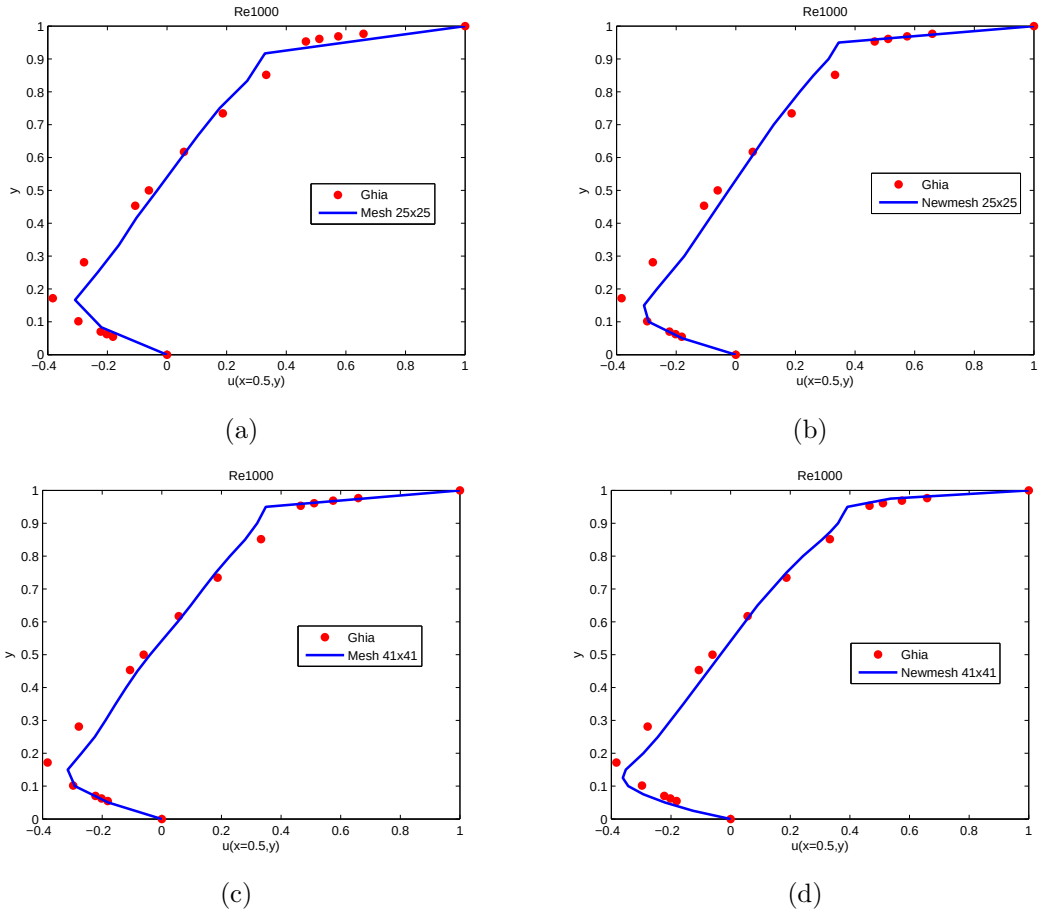

Figure 4.13. The comparison of horizontal velocity distributions at Re 1000 with Ghia et al. by using different meshes.

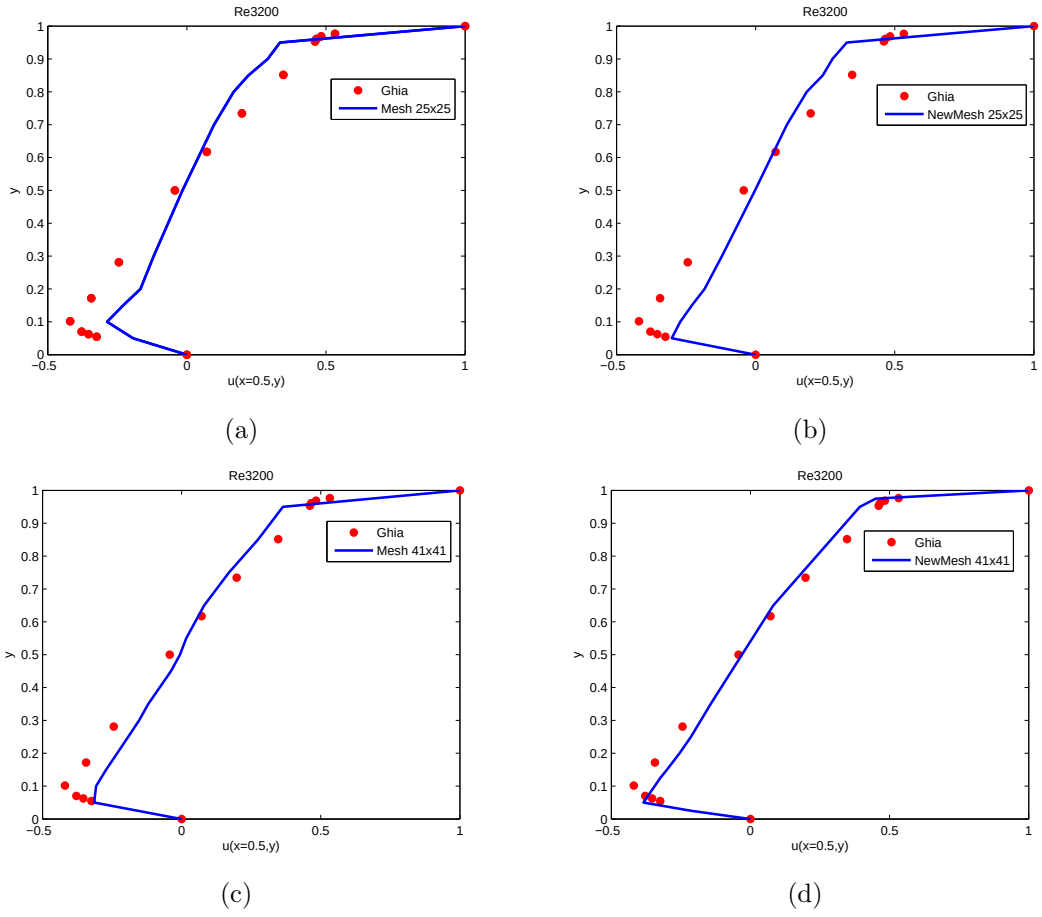

Figure 4.14. The comparison of horizontal velocity distributions at Re 3200 with Ghia et al. by using different meshes.

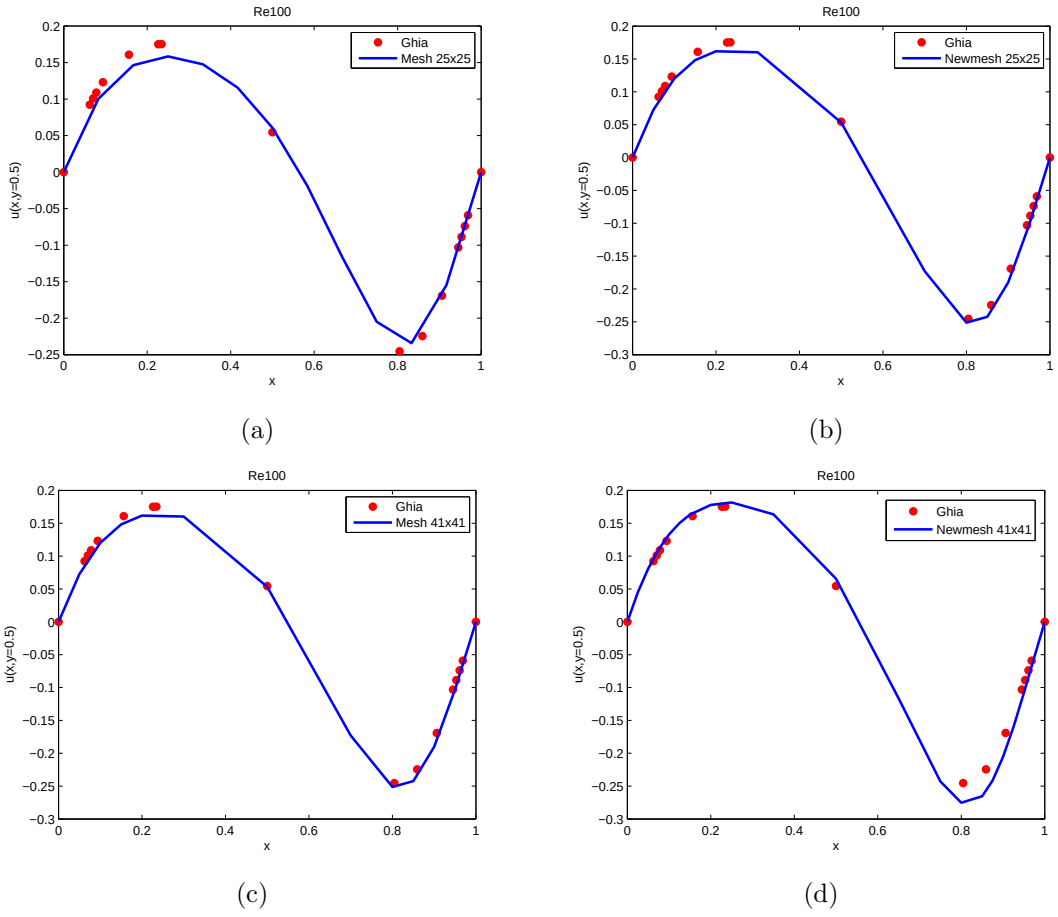

Figure 4.15. The comparison of vertical velocity distributions at Re 100 with Ghia et al. by using different meshes.

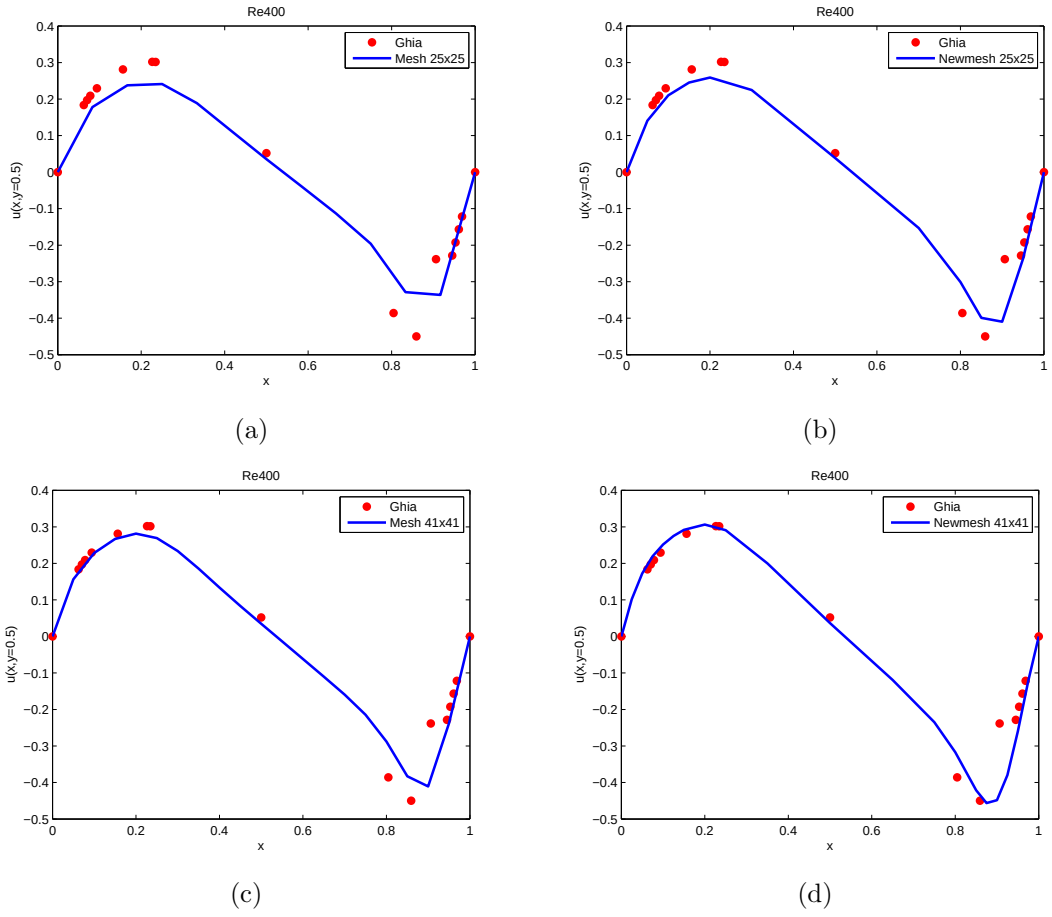

Figure 4.16. The comparison of vertical velocity distributions at Re 400 with Ghia et al. by using different meshes.

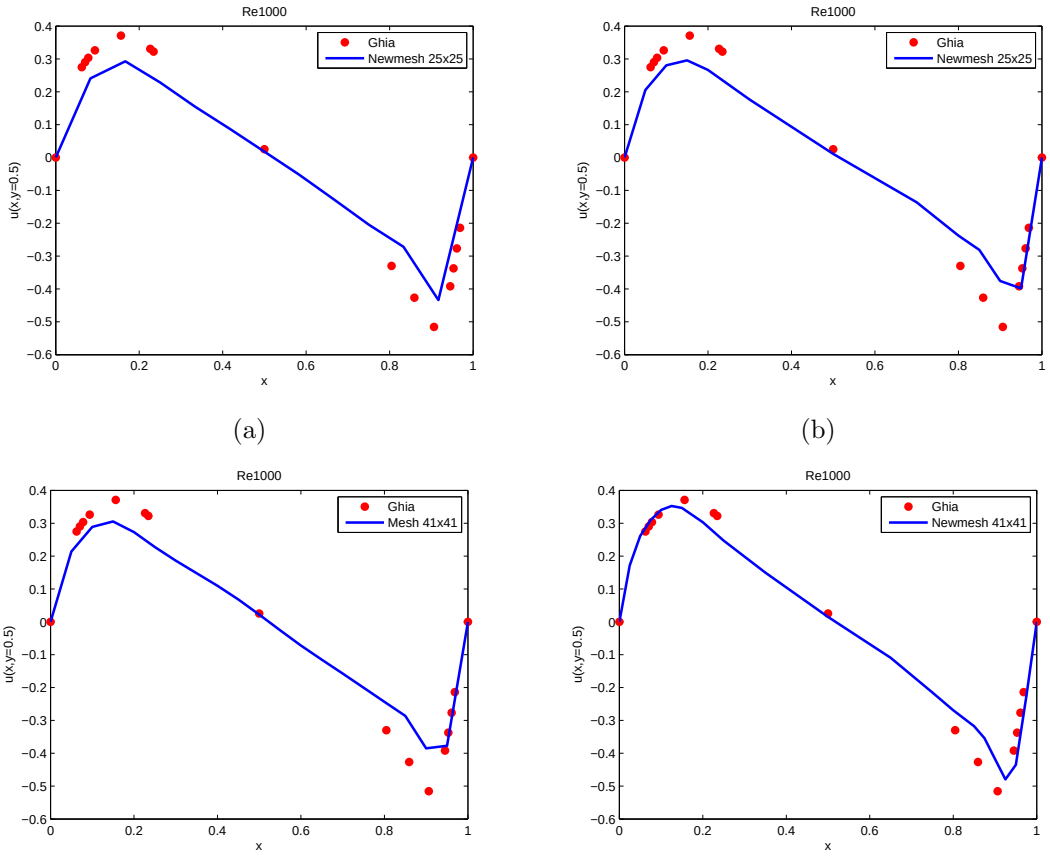

Figure 4.17. The comparison of vertical velocity distributions at Re 1000 with Ghia et al. by using different meshes.

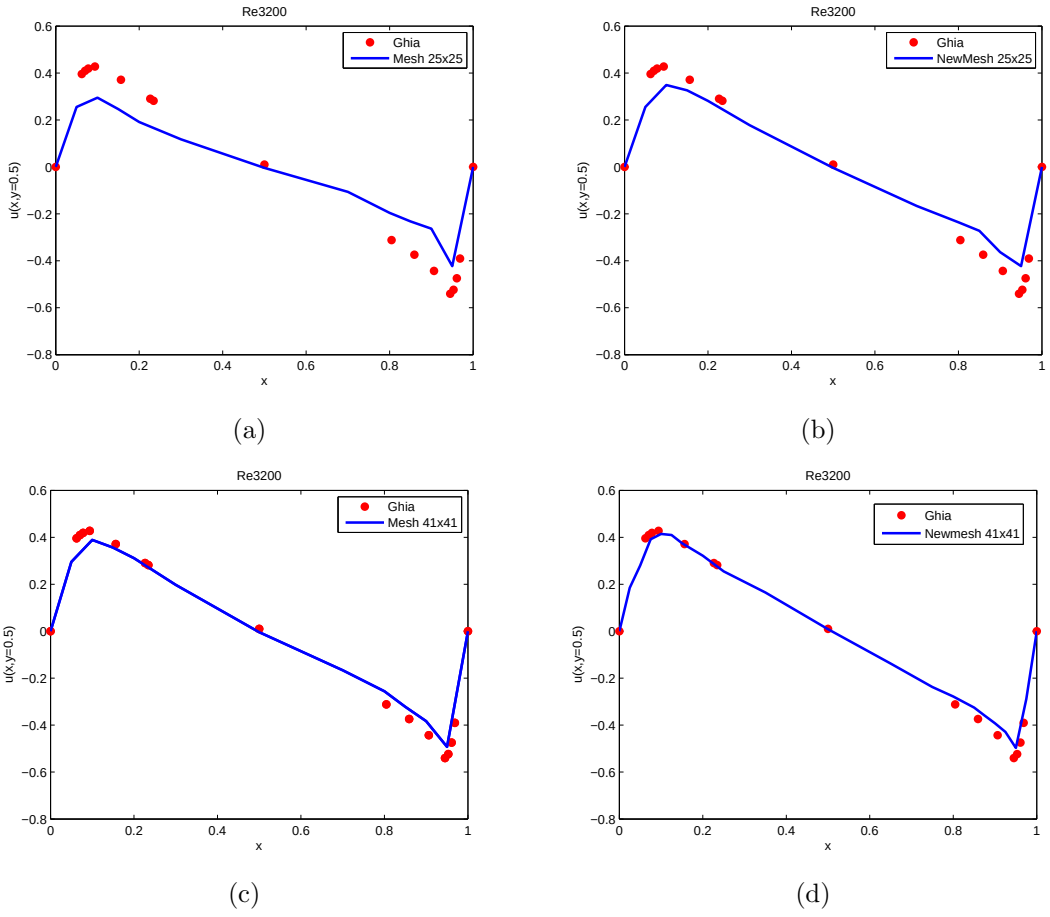

Figure 4.18. The comparison of vertical velocity distributions at Re 3200 with Ghia et al. by using different meshes.

CHAPTER 5

CONCLUSION AND RECOMMENDATION FOR FUTURE WORK

In this paper, a standard Galerkin finite element method which features unequal order interpolation for velocity variables and pressure for the conservative formulation of the unsteady incompressible Navier Stokes equations was discussed. The artificial compressibility method was employed to transform the elliptic incompressible continuity equation to hyperbolic compressible system by adding artificial compressibility term to continuity equation in order to deal with the difficulty of incompressibility constraint conditions and singular matrix. The Taylor Galerkin formulation was employed as the stabilization procedures to deal with the difficulty of numerical oscillations due to the convection dominated at high Reynolds number flows. The coefficient matrices associated with the continuity equation and momentum equations are directly constructed by employing the transformation of interpolation functions of quadratic triangular element and the time derivative of mass terms are linearized by using the first order of Taylor expansion.

In the solution, unsteady incompressible Navier Stokes equations were solved by the combining a transient time integration algorithm with an iterative method at each time plane. The Newton-Raphson iterative method was used to solve the set of non-linear equations and backward difference scheme was employed to dealt with the problem of time derivative term. The mixed set of these algorithms are called the AC Backward Newton-Raphson method. The finite element programming codes of these methods were set by MATLAB program. The final numerical result were tested by the convergence checking associated with velocity components and pressure.

The numerical method is verified by solving the flow in the benchmark of lid-driven cavity. In order to demonstrate the influence of very fine mesh with numerical results, four different meshes of 25x25 nodes structure grid, 41x41 nodes structure grid, 25x25 nodes new refined mesh, and 41x41 nodes new refined mesh were employed in the solutions. As mentioned in previous chapter, the iteration method of Newton-Raphson can make the solution very fast convergence. However, although the number of iterations can be reduced by this method, the CPU time is still large for a conventional PC because of the Galerkin formulation make the matrix that involve in the solution became large. Thus, the large CPU time for solving the set of equations still encountered in the solution. For this reason, the very big meshes (very fined mesh) cannot be used to reduce some errors in the process. Therefore, the small vortices at the bottom right and left corner can not be generated. For solving this problem, the refined mesh near the wall of cavity need to be required. In this paper, to avoid using very fine meshes that can make more CPU times in the solutions, the structure meshes (25x25 and 41x41 nodes) would be refined to new refined mesh near the wall 25x25 and 41x41 nodes. However, these meshes were not enough to deal with the numerical oscillations due to the convection dominated at high Reynolds number of Re 3200. Thus, to deal with this difficulty, the Taylor Galerkin stabilized method need to be required. From using these meshes to the solutions and employing the Taylor Galerkin method to stabilize the numerical results, it was found that the final numerical results were obtained when CPU time involved is less than one hour in a conventional PC, depending on Reynolds number.

The numerical results were compared with those of Ghia et al. [1] for Reynolds number of 100, 400, and 1000 by using different meshes. From the comparing, it was found that the method seems to achieve good result for low Reynolds number (Re 100) as can be seen in figure 4.7-4.13. When the Reynolds number is increased

to Reynolds number of 400, and 1000, respectively, some errors encountered in the solution. However, the very fine mesh near the wall of cavity can make the results accuracy than the structure meshes when compared by using the same number of nodes. Moreover, it was also found the smooth vortices can be generated at the bottom right and left corner by using these meshes. For the high Reynolds number at Re 3200, as show in figure 4.14, it was found that the difficulty of the numerical oscillations due to convection dominated can be solved by the stabilized method of Taylor Galerkin. Moreover, it was also found that the accuracy numerical results were obtained when the very fine mesh near the wall was selected in the solutions.

From these numerical results, the advantage of the solution to incompressible Navier-Stokes equations by using numerical finite element method based on artificial compressibility method and Taylor Galerkin stabilized technique can be concluded as follow

1. The program seems to achieve good results when compared with numerical results of Ghia et al.(121x121mesh) at Re 100, 400, 1000, and 3200) with even less refined meshes to the solutions.

2. The numerical solutions of Newton and Raphson method can give the numerical results are very fast convergence to reach steady state.

3. The finite element programming can save more CPU time to iterate the system of equations because the stabilized technique can be employed to deal with the difficulty of numerical oscillations due to convection dominated instead of using very fine meshes to the solutions that can make more CPU times.

As the future work, there are two extensions of this work,

Firstly, in order to improve the accuracy numerical results at Re 100, 400, 1000, and 3200 the very big fine mesh near the wall of cavity need to be refined for example 64x64 nodes new refined mesh is enough to get the accuracy results.

in additionally, in order to reduce the CPU time, the language of C++ and FORTRAN program can be the best choice to solve the system of equations instead of MATLAB program. Moreover, the Characteristic-Based-Split algorithm based on artificial compressibility method can be employed to solve incompressible Navier-Stoke equations by using both of implicit and explicit scheme[17]. This method is widely used at the present time because not only the CPU time can be reduced but also the finite element programming codes by using this method can be constructed easily.

APPENDIX A

THE CONSTRUCTION OF FINITE ELEMENT MATRIX FORMULAS

In this appendix, the coefficient matrices formulas in table 2.1 will be derived to the general form of matrix. Moreover, the summary of matrix form will be expressed in order to make more convenient for finite element programming codes construction.

A.1 The construction of coefficient matrix formulas

Let us start with the transformation of interpolation functions

$$\psi = \begin{bmatrix} L_1(2L_1-1) \\ L_2(2L_2-1) \\ L_3(2L_3-1) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{bmatrix} = \begin{bmatrix} L_1^2 - L_1(L_2+L_3) \\ L_2^2 - L_2(L_3+L_1) \\ L_3^2 - L_3(L_1+L_2) \\ 4L_1L_2 \\ 4L_2L_3 \\ 4L_3L_1 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1L_2 \\ L_2L_3 \\ L_3L_1 \end{bmatrix} \tag{A.1}$$

Where

$$[A]_{6x6} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \tag{A.2}$$

and

$$[R]_{6x1} = \begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1L_2 \\ L_2L_3 \\ L_3L_1 \end{bmatrix} \tag{A.3}$$

Mass matrix:$M = \rho \int_{\Omega^e} \psi\psi^T d\Omega$

$$\psi\psi^T = [A][A]^T \begin{bmatrix} L_1^2 \\ L_2^2 \\ L_3^2 \\ L_1L_2 \\ L_2L_3 \\ L_3L_1 \end{bmatrix} \begin{bmatrix} L_1^2 & L_2^2 & L_3^2 & L_1L_2 & L_2L_3 & L_3L_1 \end{bmatrix} \tag{A.4}$$

Substitute [A.4] into mass matrix $[M]$ and apply this formula$\int_{A^e} L_1^a L_2^b L_3^c dA = \frac{a!b!c!}{(a+b+c+2)!} 2A$ to integrate. The results of integrations are shown below.

$$\int_A L_1^4 dA = \frac{4!0!0!}{(4+0+0+2)!} 2A = \frac{2A}{30}$$

$$\int_A L_1^2 L_1^2 dA = \frac{2!2!0!}{(2+2+0+2)!} 2A = \frac{2A}{180}$$

$$\int_A L_1 L_2^2 L_3 dA = \frac{2!1!1!}{(2+1+1+2)!} 2A = \frac{2A}{360}$$

$$\int_A L_2^3 L_3 dA = \frac{3!1!0!}{(3+1+0+2)!} 2A = \frac{2A}{120}$$

$$M = [A][A]^T \frac{2A}{360} \begin{bmatrix} 12 & 2 & 2 & 3 & 1 & 3 \\ 2 & 12 & 2 & 3 & 3 & 1 \\ 2 & 2 & 12 & 1 & 3 & 3 \\ 3 & 3 & 1 & 2 & 1 & 1 \\ 1 & 3 & 3 & 1 & 2 & 1 \\ 3 & 1 & 3 & 1 & 1 & 2 \end{bmatrix} \tag{A.5}$$

Thus, we got the final form of mass matrix as expressed in (A.5). Where A is an element area.

A.2 Integration of $[F]$ matrix

$$F = \int_{\Omega^e} \begin{bmatrix} L_1^4 & L_1^2L_2^2 & L_1^2L_3^2 & L_1^3L_2 & L_1^2L_2L_3 & L_1^3L_3 \\ L_1^2L_2^2 & L_2^4 & L_2^2L_3^2 & L_1L_2^3 & L_2^3L_3 & L_2^2L_3L_1 \\ L_1^2L_3^2 & L_2^2L_3^2 & L_3^4 & L_3^2L_2L_1 & L_3^3L_2 & L_3^3L_1 \\ L_1^3L_2 & L_1L_2^3 & L_3^2L_2L_1 & L_1^2L_2^2 & L_2^2L_3L_1 & L_1^2L_3L_2 \\ L_1^2L_2L_3 & L_2^3L_3 & L_3^3L_2 & L_2^2L_3L_1 & L_2^2L_3^2 & L_3^2L_2L_1 \\ L_1^3L_3 & L_2^2L_3L_1 & L_3^3L_1 & L_1^2L_3L_2 & L_3^2L_2L_1 & L_3^2L_1^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} [H]d\Omega \tag{A.6}$$

Integrate (A.6) by using $\int_{A^e} L_1^a L_2^b L_3^c dA = \frac{a!b!c!}{(a+b+c+2)!}2A$. The result of integration are shown below

$\int_{A^e} L_1^5 dA = \frac{5!0!0!}{(5+0+0+2)!}2A = \frac{2A}{42}$

$\int_{A^e} L_1^4 L_2 dA = \frac{4!1!0!}{(4+1+0+2)!}2A = \frac{2A}{210}$

$\int_{A^e} L_1^2 L_2^2 L_3 dA = \frac{2!2!1!}{(2+2+1+2)!}2A = \frac{2A}{1260}$

$\int_{A^e} L_2^3 L_2^2 dA = \frac{3!2!0!}{(3+2+0+2)!}2A = \frac{2A}{420}$

$\int_{A^e} L_1^3 L_2 L_2 dA = \frac{3!1!1!}{(3+1+1+2)!}2A = \frac{2A}{840}$

$\int_{A^e} L_2^3 L_2 dA = \frac{3!1!0!}{(3+1+0+2)!}2A = \frac{2A}{210}$

Thus, we got [F] matrix as

$$F = \frac{2A}{5040}\begin{bmatrix} F(1,1) & F(1,2) & F(1,3) & F(1,4) & F(1,5) & F(1,6) \\ F(2,1) & F(2,2) & F(2,3) & F(2,4) & F(2,5) & F(2,6) \\ F(3,1) & F(3,2) & F(3,3) & F(3,4) & F(3,5) & F(3,6) \end{bmatrix} \tag{A.7}$$

where

$$F(1,1) = \begin{bmatrix} 120 & 12 & 12 & 24 & 6 & 24 \end{bmatrix}[u]$$

$$F(1,2) = \begin{bmatrix} 12 & 24 & 14 & 12 & 6 & 4 \end{bmatrix}[u]$$

$$F(1,3) = \begin{bmatrix} 12 & 4 & 24 & 4 & 6 & 24 \end{bmatrix}[u]$$

$$F(1,4) = \begin{bmatrix} 24 & 12 & 4 & 12 & 4 & 6 \end{bmatrix}[u]$$

$$F(1,5) = \begin{bmatrix} 6 & 6 & 6 & 4 & 4 & 4 \end{bmatrix}[u]$$

$$F(1,6) = \begin{bmatrix} 120 & 12 & 12 & 24 & 6 & 24 \end{bmatrix}[u]$$

$$F(2,1) = \begin{bmatrix} 24 & 12 & 4 & 12 & 4 & 6 \end{bmatrix}[u]$$

$$F(2,2) = \begin{bmatrix} 12 & 120 & 12 & 24 & 24 & 6 \end{bmatrix}[u]$$

$$F(2,3) = \begin{bmatrix} 4 & 12 & 24 & 4 & 12 & 6 \end{bmatrix}[u]$$

$$F(2,4) = \begin{bmatrix} 12 & 12 & 4 & 12 & 6 & 4 \end{bmatrix}[u]$$

$$F(2,5) = \begin{bmatrix} 4 & 24 & 12 & 6 & 12 & 4 \end{bmatrix}[u]$$

$$F(2,6) = \begin{bmatrix} 6 & 6 & 6 & 4 & 4 & 4 \end{bmatrix}[u]$$

$$F(3,1) = \begin{bmatrix} 24 & 12 & 24 & 6 & 4 & 12 \end{bmatrix}[u]$$

$$F(3,1) = \begin{bmatrix} 4 & 24 & 12 & 6 & 12 & 4 \end{bmatrix}[u]$$

$$F(3,1) = \begin{bmatrix} 12 & 12 & 120 & 6 & 24 & 24 \end{bmatrix}[u]$$

$$F(3,1) = \begin{bmatrix} 6 & 6 & 6 & 4 & 4 & 4 \end{bmatrix}[u]$$

$$F(3,1) = \begin{bmatrix} 4 & 12 & 24 & 4 & 12 & 6 \end{bmatrix}[u]$$

$$F(3,1) = \begin{bmatrix} 12 & 4 & 24 & 4 & 6 & 12 \end{bmatrix} [u]$$

A.3 The summary of coefficient matrix formulas

These are the coefficient matrix formulas of incompressible Navier-Stokes equations. Let start with mass matrix: $M = \rho \int_{\Omega^e} \psi\psi^T d\Omega$

$$M = [A][A]^T \frac{2A}{360} \begin{bmatrix} 12 & 2 & 2 & 3 & 1 & 3 \\ 2 & 12 & 2 & 3 & 3 & 1 \\ 2 & 2 & 12 & 1 & 3 & 3 \\ 3 & 3 & 1 & 2 & 1 & 1 \\ 1 & 3 & 3 & 1 & 2 & 1 \\ 3 & 1 & 3 & 1 & 1 & 2 \end{bmatrix} \tag{A.8}$$

Mass matrix M_p (Artificial compressibility term):$M_p = \frac{1}{\beta} \int_{\Omega^e} \phi\phi^T d\Omega$

$$M_p = \frac{1}{\beta} \int_{A^e} \begin{bmatrix} L_1^2 & L_1L_2 & L_1L_3 \\ L_1L_2 & L_2^2 & L_2L_3 \\ L_1L_3 & L_2L_3 & L_3^2 \end{bmatrix} dA = \frac{1}{\beta}[G] \tag{A.9}$$

Diffusion matrix: $K_{11} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial x} \frac{\partial\psi^T}{\partial x} d\Omega$ and $K_{22} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial y} \frac{\partial\psi^T}{\partial y} d\Omega$

$$K_{11} = \mu[A][B][G][B]^T[A]^T \tag{A.10}$$

$$K_{22} = \mu[A][C][G][C]^T[A]^T \tag{A.11}$$

Diffusion matrix: $K_{12} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial x} \frac{\partial\psi^T}{\partial y} d\Omega$ and $K_{21} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial y} \frac{\partial\psi^T}{\partial x} d\Omega$

$$K_{12} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial x} \frac{\partial\psi^T}{\partial y} d\Omega = \mu[A][B][G][A]^T[C]^T \tag{A.12}$$

$$K_{21} = \int_{\Omega^e} \mu \frac{\partial\psi}{\partial y} \frac{\partial\psi^T}{\partial x} d\Omega = \mu[A][C][G][B]^T[A]^T \tag{A.13}$$

Stiffness mixed matrix: $Q_1 = \int_{\Omega^e} \frac{\partial \psi}{\partial x} \phi^T d\Omega$ and $Q_2 = \int_{\Omega^e} \frac{\partial \psi}{\partial y} \phi^T d\Omega$

$$Q_1 = [A][B] \int_{\Omega^e} [H][H]^T d\Omega = [A][B][G] \tag{A.14}$$

$$Q_2 = [A][C] \int_{\Omega^e} [H][H]^T d\Omega = [A][C][G] \tag{A.15}$$

Stabilized matrix:

$$\begin{aligned} Kse(u,v) = \frac{\Delta t}{2} &[uu[A][B][G][B][A] + uv[A][B][G][C][A] + vu[A][C][G][B][A]] \\ &+ [vv[A][C][G][C][A]] \end{aligned} \tag{A.16}$$

Convective matrix: $C(u,v) = \int_{\Omega^e} \rho \left(\psi(\psi^T u) \frac{\partial \psi^T}{\partial x} + \psi(\psi^T v) \frac{\partial \psi^T}{\partial y} \right) d\Omega$

$$C_x(u) = \frac{2A}{5040} [A][A]^T [A][B][F] \tag{A.17}$$

$$C_y(v) = \frac{2A}{5040} [A][A]^T [A][C][F] \tag{A.18}$$

Where

$$[A]_{6x6} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix} \tag{A.19}$$

$$B = \frac{1}{2A} \begin{bmatrix} 2b_1 & 0 & 0 \\ 0 & 2b_2 & 0 \\ 0 & 0 & 2b_3 \\ b_2 & b_1 & 0 \\ 0 & b_3 & b_2 \\ b_3 & 0 & b_1 \end{bmatrix} \tag{A.20}$$

$$C = \frac{1}{2A}\begin{bmatrix} 2c_1 & 0 & 0 \\ 0 & 2c_2 & 0 \\ 0 & 0 & 2c_3 \\ c_2 & c_1 & 0 \\ 0 & c_3 & c_2 \\ c_3 & 0 & c_1 \end{bmatrix} \tag{A.21}$$

$$G = \frac{A}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{A.22}$$

where element area:

$$A = \frac{1}{2}\left[x_2(y_3 - y_1) + x_1(y_2 - y_3) + x_3(y_1 - y_2)\right] \tag{A.23}$$

Element coefficient coordinates:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} (x_2y_3 - x_3y_2) \\ (x_3y_1 - x_1y_3) \\ (x_1y_2 - x_2y_1) \\ y_2 - y_3 \\ y_3 - y_1 \\ y_1 - y_2 \\ x_3 - x_2 \\ x_1 - x_3 \\ x_2 - x_1 \end{bmatrix} \tag{A.24}$$

Matrix $[F]$ formula for convective term:

$$F = \frac{2A}{5040}\begin{bmatrix} F(1,1) & F(1,2) & F(1,3) & F(1,4) & F(1,5) & F(1,6) \\ F(2,1) & F(2,2) & F(2,3) & F(2,4) & F(2,5) & F(2,6) \\ F(3,1) & F(3,2) & F(3,3) & F(3,4) & F(3,5) & F(3,6) \end{bmatrix} \tag{A.25}$$

where

$$F(1,1) = \begin{bmatrix} 120 & 12 & 12 & 24 & 6 & 24 \end{bmatrix} [u]$$

$$F(1,2) = \begin{bmatrix} 12 & 24 & 14 & 12 & 6 & 4 \end{bmatrix} [u]$$

$$F(1,3) = \begin{bmatrix} 12 & 4 & 24 & 4 & 6 & 24 \end{bmatrix} [u]$$

$$F(1,4) = \begin{bmatrix} 24 & 12 & 4 & 12 & 4 & 6 \end{bmatrix} [u]$$

$$F(1,5) = \begin{bmatrix} 6 & 6 & 6 & 4 & 4 & 4 \end{bmatrix} [u]$$

$$F(1,6) = \begin{bmatrix} 120 & 12 & 12 & 24 & 6 & 24 \end{bmatrix} [u]$$

$$F(2,1) = \begin{bmatrix} 24 & 12 & 4 & 12 & 4 & 6 \end{bmatrix} [u]$$

$$F(2,2) = \begin{bmatrix} 12 & 120 & 12 & 24 & 24 & 6 \end{bmatrix} [u]$$

$$F(2,3) = \begin{bmatrix} 4 & 12 & 24 & 4 & 12 & 6 \end{bmatrix} [u]$$

$$F(2,4) = \begin{bmatrix} 12 & 12 & 4 & 12 & 6 & 4 \end{bmatrix} [u]$$

$$F(2,5) = \begin{bmatrix} 4 & 24 & 12 & 6 & 12 & 4 \end{bmatrix} [u]$$

$$F(2,6) = \begin{bmatrix} 6 & 6 & 6 & 4 & 4 & 4 \end{bmatrix} [u]$$

$$F(3,1) = \begin{bmatrix} 24 & 12 & 24 & 6 & 4 & 12 \end{bmatrix} [u]$$

$$F(3,1) = \begin{bmatrix} 4 & 24 & 12 & 6 & 12 & 4 \end{bmatrix} [u]$$

$$F(3,1) = \begin{bmatrix} 12 & 12 & 120 & 6 & 24 & 24 \end{bmatrix} [u]$$

$$F(3,1) = \begin{bmatrix} 6 & 6 & 6 & 4 & 4 & 4 \end{bmatrix} [u]$$

$$F(3,1) = \begin{bmatrix} 4 & 12 & 24 & 4 & 12 & 6 \end{bmatrix} [u]$$

$$F(3,1) = \begin{bmatrix} 12 & 4 & 24 & 4 & 6 & 12 \end{bmatrix} [u]$$

APPENDIX B

THE FINITE ELEMENT PROGRAMMING CODES

In this appendix, the standard Galerkin finite element programming codes for the unsteady incompressible Navier-Stokes equations based on artificial compressibility method will be written here. The finite element programming codes are constructed by MATLAB program. These codes consist of one main program and three sub functions as mentioned in chapter 3.

B.1 The Main programming codes

B.1.1 Input data

```
Incompressible Navier-Stokes equations program
NAME: Wanchai Jiajan, Aerospace Engineering
clc
clear all
Parameter input
Mxpoiv=625;
Mxpoip=169;
Mxele=288;
Mxfree=1;
Mxneq=2*Mxpoiv+Mxpoip; Number of equation in system
Npoiv=625; Number nodes of velocity components
Npoip=169; Number nodes of pressure
Nelem=288; Number of elements
Reynolds number (density*velocity*lenght/viscosity)
den=1;
vis=0.01;
Artificial parameter
```

```
betha=8;
Matrix dimensions
Uvel=zeros(Mxpoiv,1);
Vvel=zeros(Mxpoiv,1);
Pres=zeros(Mxpoiv,1);
SysM=zeros(Mxneq,Mxneq); size of Mass matrix
SysK=zeros(Mxneq,Mxneq); size of stiffness matrix System
SysR=zeros(Mxneq,1);
SysN=zeros(Mxneq,1); size of force matrix system
Sol=zeros(Mxneq,1); size of result solution
Dsol=zeros(Mxneq,1); size of result solution
nodes=zeros(Mxele,6);
nodesf=zeros(Mxfree,4);
Ibcu=zeros(Mxpoiv,1); number of boundary for U
Ibcv=zeros(Mxpoiv,1); number of boundary for V
Ibcp=zeros(Mxpoiv,1); number of boundary for P
deltt=0.1; time step size for transient analysis
stime=0.0; initial time
ftime=0.0005; termination time
ntime=fix((ftime-stime)/deltt); number of time increment
Read input file (element connectivity)
fid = fopen('Mesh288elements.txt', 'r');
data1 = textscan (fid, '
nodes=[data1{1} data1{2} data1{3} data1{4} data1{5} data1{6} ];
Read input file ( Node co-ordinates)
fid = fopen('Meshnodes625.txt', 'r');
```

```
data2 = textscan(fid, '
X1=data2{1} ;
Y1=data2{2} ;
gcoord=[X1 Y1];
Plot mesh
figure(1)
clf
plot(X1,Y1,'.r','MarkerSize',10)
hold on
tri=nodes;
triplot(tri,X1,Y1);
Boundary condition
fid = fopen('625BC.txt', 'r');
data1 = textscan(fid, '
Ibcu=data1{1} ;
Ibcv=data1{2} ;
Ibcp=data1{3} ;
Uvel=data1{4} ;
Vvel=data1{5} ;
Pres=data1{6} ;
```

B.1.2 Loop for number of equations

```
Neq=2*Npoiv+Npoip;
Loop for initilize force vector
for i=1:Neq
SysR(i)=0;
```

```
end
Loop for initilize stiffness matrix
for i=1:Neq
for j=1:Neq
SysK(i,j)=0;
end
end
Loop for initilize mass matrix
for i=1:Neq
for j=1:Neq
SysM(i,j)=0;
end
end
```

B.1.3 Mass matrix construction

```
[SysM]=TRInew1(Npoiv,Nelem,den,vis,gcoord,nodes,SysM);
```

B.1.4 Initial guess value input

```
for i=1:Npoiv loop for velocity components
Sol(i)=0.0356;
Sol(i+Npoiv)=0.00032272;
end
for i=1:Npoip loop for pressure
Sol(i+Npoiv+Npoiv)=0.998;
end
```

B.1.5 Numerical programming codes "AC Backward Newton-Raphson Method"

```
SysK=SysM+deltt*SysK;
sum=zeros(ntime,1);
check=zeros(ntime,1);
for it=1:ntime iteration in each time plane
for iter=1:100 number of iterations
Newton-Raphson Non-linear matrix construction
[SysK,SysR]=TRI(Npoiv,Npoip,Nelem,Nfree,Neq,den,vis,
gcoord,nodes,SysK,SysR,Sol);
Backward different scheme
SysN=deltt*SysR+SysM*Sol;
apply boundary condition call "ApplyBC55"
[SysK, SysN]=ApplyBC55(Npoiv,Npoip,Neq,Ibcu,Ibcv,Ibcp,
SysK,SysN,Uvel,Vvel,Pres);
Solve equations
Sol=inv(SysK)*SysN
iteration checking
for i=1:Npoip
Psol(i,it)=Sol(2*Npoiv+i);
Psol(i,it+1)=Sol(2*Npoiv+i);
end
Iteration=iter
end
Ntime=it
end
Convergence Checking
```

```
for it=1:ntime
for i=1:Npoip
sum(it)=sum(it)+(Psol(i,it+1)-Psol(i,it))/deltt;
check(it)=abs(1/betha*sum(it));
end
Check(it,1)=check(it);
if(abs(check(it))¡1e-6)
fprintf('Pressure converged in iterations');
break;
end
Ntime=it;
end
```

B.1.6 Output

```
plot the history of convergence
figure(2)
time=0:deltt:Ntime*deltt;
semilogy(time,(Check(:,1)),'-');
xlabel('Time')
ylabel('Pressure convergence')
title('Convergence history')
Print output
for i=1:Npoiv
Uvelocity(i)=Sol(i);
Vvelocity(i)=Sol(Npoiv+i);
end
```

```
for i=1:Npoip
Pressure(i)=Sol(i+2*Npoiv);
end
Velocity=[Uvelocity' Vvelocity']
Pressure=Pressure'
plot velocity vector
x=X1;
y=Y1;
scale=2;
u=Uvelocity';
v=Vvelocity';
figure(3)
quiver ( x, y, u, v, scale, 'b' );
axis equal
hold on
k = convhull ( x, y );
plot ( x(k), y(k), 'r' );
hold on
xmin = min ( x );
xmax = max ( x );
ymin = min ( y );
ymax = max ( y );
delta = 0.05 * max ( xmax - xmin, ymax - ymin );
plot ( [ xmin - delta,xmax + delta,xmax + delta,xmin - delta, xmin - delta ],
[ ymin - delta, ymin - delta,ymax + delta,ymax + delta,ymin - delta ],
'w' );
```

```
hold off
```

B.2 Sub programming "Functions"

B.2.1 Mass matrix Coefficient Function "TRInew1"

```
function [SysM]=TRInew1(Npoiv,Nelem,den,vis,gcoord,nodes,SysM)
A=zeros(6,6);B=zeros(6,3);C=zeros(6,3);G=zeros(3,3);
M11=zeros(6,6);M22=zeros(6,6);M33=zeros(3,3);Mele=zeros(15,15);
Rele=zeros(15,1);
Set up matrix [A]
for i=1:6
for j=1:6
A(i,j)=0;
end
end
A(1,1)=1;A(2,2)=1;A(3,3)=1;A(4,4)=4;A(5,5)=4;
A(6,6)=4;A(1,5)=-1;A(1,6)=-1;A(2,4)=-1;A(2,6)=-1;
A(3,4)=-1;A(3,5)=-1;
betha=8;
Anew=vis/den; kinematic viscosity
for iel=1:Nelem loop for the total number of elements
II=nodes(iel,1);
JJ=nodes(iel,2);
KK=nodes(iel,3);
LL=nodes(iel,4);
MM=nodes(iel,5);
```

```
NN=nodes(iel,6);
x1=gcoord(II,1); y1=gcoord(II,2);
x2=gcoord(JJ,1); y2=gcoord(JJ,2);
x3=gcoord(KK,1); y3=gcoord(KK,2);
Area=0.5*((x2*y3)+(x1*y2)+(x3*y1)-(x2*y1)-(x1*y3)-(x3*y2));
Area2=2*Area;
b1=(y2-y3)/Area2;
b2=(y3-y1)/Area2;
b3=(y1-y2)/Area2;
c1=(x3-x2)/Area2;
c2=(x1-x3)/Area2;
c3=(x2-x1)/Area2;
Compute matrix [B] [C] and [G]
for i=1:6
for j=1:3
B(i,j)=0;
C(i,j)=0;
end end
B(1,1)=2*b1;B(5,1)=b3;B(6,1)=b2;B(2,2)=2*b2;
B(4,2)=b3; B(6,2)=b1; B(3,3)=2*b3; B(4,3)=b2; B(5,3)=b1;
C(1,1)=2*c1; C(2,2)=2*c2; C(3,3)=2*c3; C(5,1)=c3;
C(5,3)=c1; C(6,1)=c2; C(6,2)=c1; C(4,2)=c3; C(4,3)=c2;
Fac=Area/12;
Fac2=2*Fac;
G(1,1)=Fac2; G(2,2)=Fac2; G(3,3)=Fac2; G(1,2)=Fac; G(1,3)=Fac;
G(2,1)=Fac; G(2,3)=Fac; G(3,1)=Fac; G(3,2)=Fac;
```

```
Set up mass matrix
D(1,1)=12; D(2,2)=12; D(3,3)=12; D(4,4)=2; D(5,5)=2; D(6,6)=2;
D(1,2)=2; D(1,3)=2; D(1,4)=1; D(1,5)=3; D(1,6)=3; D(2,3)=2; D(2,4)=3;
D(2,5)=1; D(2,6)=3; D(3,4)=3; D(3,5)=3; D(3,6)=1; D(4,5)=1;
D(4,6)=1; D(5,6)=1;
for i=1:6
for j=1:6
D(j,i)=D(i,j);
end
end
M11=2*Area/360*den*A*D*A';
M22=2*Area/360*den*A*D*A';
M33=G/betha;
Then the matrix (15x15) on LHS is
for i=1:15
for j=1:15
Mele(i,j)=0;
end
end
Final local mass matrix
for i=1:6
for j=1:6
Mele(i,j)=M11(i,j);
Mele(i+6,j+6)=M22(i,j);
end
end
```

```
for i=1:3
for j=1:3
Mele(i+12,j+12)=M33(i,j);
end
end
Assembly Global mass matrix
[SysM]=ASSEMBLEnew1(iel,nodes,Mele,SysM,Npoiv);
end
```

B.2.2 Assembly to Global mass matrix function "ASSEMBLEnew1"

```
function [SysM]=ASSEMBLEnew1(iel,nodes,Mele,SysM,Npoiv)
Contribution of coeficients assocoated with u,v velocities
for i=1:6
for j=1:6
II=nodes(iel,i);
JJ=nodes(iel,j);
k=i+6;
l=j+6;
KK=Npoiv+II;
LL=Npoiv+JJ;
SysM(II,JJ)=SysM(II,JJ)+Mele(i,j);
SysM(II,LL)=SysM(II,LL)+Mele(i,l);
SysM(KK,JJ)=SysM(KK,JJ)+Mele(k,j);
SysM(KK,LL)=SysM(KK,LL)+Mele(k,l);
end
end
```

```
Contribution of coeficients assocoated with P pressure
for i=1:3
for j=1:3
II=nodes(iel,i);
JJ=nodes(iel,j);
k=i+12;
l=j+12;
KK=2*Npoiv+II;
LL=2*Npoiv+JJ;
SysM(II,LL)=SysM(II,LL)+Mele(i,l);
SysM(KK,LL)=SysM(KK,LL)+Mele(k,l);
end
end
```

B.2.3 Local stiffness coefficient matrix function "TRI"

```
function [SysK,SysR]=TRI(Npoiv,Npoip,Nelem,Neq,den,vis,
gcoord,SysK,SysR,Sol);
A=zeros(6,6);B=zeros(6,3);C=zeros(6,3);G=zeros(3,3);F=zeros(6,6,3);
Uele=zeros(6,1);Vele=zeros(6,1);Pele=zeros(3,1);
Sxx=zeros(6,6);Sxy=zeros(6,6);Syx=zeros(6,6);Syy=zeros(6,6);
Qx=zeros(3,6);Qy=zeros(3,6);Qxt=zeros(6,3);Qyt=zeros(6,3);
ABGXUG=zeros(6,6);AGBXUG=zeros(6,6);AGBYVG=zeros(6,6);
ABGYVG=zeros(6,6);ABGXVG=zeros(6,6);ABGYUG=zeros(6,6);
Gxx=zeros(6,6);Gyy=zeros(6,6);Alx=zeros(6,6);Aly=zeros(6,6);
Akele=zeros(15,15);Rele=zeros(15,1);FX=zeros(6,1);FY=zeros(6,1);
FI=zeros(3,1);
```

```
Set up matrix [A]
for i=1:6
for j=1:6
A(i,j)=0;
end
end
A(1,1)=1; A(2,2)=1; A(3,3)=1; A(4,4)=4; A(5,5)=4;
A(6,6)=4; A(1,5)=-1; A(1,6)=-1; A(2,4)=-1; A(2,6)=-1;
A(3,4)=-1; A(3,5)=-1;
Anew=vis/den;
for iel=1:Nelem loop for the total number of elements
II=nodes(iel,1);
JJ=nodes(iel,2);
KK=nodes(iel,3);
LL=nodes(iel,4);
MM=nodes(iel,5);
NN=nodes(iel,6);
x1=gcoord(II,1); y1=gcoord(II,2);
x2=gcoord(JJ,1); y2=gcoord(JJ,2);
x3=gcoord(KK,1); y3=gcoord(KK,2);
Area=0.5*((x2*y3)+(x1*y2)+(x3*y1)-(x2*y1)-(x1*y3)-(x3*y2));
Area2=2*Area;
b1=(y2-y3)/Area2;
b2=(y3-y1)/Area2;
b3=(y1-y2)/Area2;
c1=(x3-x2)/Area2;
```

```
c2=(x1-x3)/Area2;
c3=(x2-x1)/Area2;
Compute matrix [B] [C] and [G]
for i=1:6
for j=1:3
B(i,j)=0;
C(i,j)=0;
end
end
B(1,1)=2*b1; B(5,1)=b3; B(6,1)=b2; B(2,2)=2*b2;
B(4,2)=b3; B(6,2)=b1; B(3,3)=2*b3; B(4,3)=b2;
B(5,3)=b1; C(1,1)=2*c1; C(2,2)=2*c2; C(3,3)=2*c3;
C(5,1)=c3; C(5,3)=c1; C(6,1)=c2; C(6,2)=c1;
C(4,2)=c3; C(4,3)=c2;
Set up [G] matrix
Fac=Area/12;
Fac2=2*Fac;
G(1,1)=Fac2; G(2,2)=Fac2; G(3,3)=Fac2; G(1,2)=Fac; G(1,3)=Fac;
G(2,1)=Fac; G(2,3)=Fac; G(3,1)=Fac; G(3,2)=Fac;
Set up matrix [F]
Factor=2*Area/5040;
F4=Factor*4;
F6=Factor*6; F12=Factor*12; F24=Factor*24; F120=Factor*120;
F(1,1,1)=F120; F(1,2,1)=F12; F(1,3,1)=F12; F(1,4,1)=F6;
F(1,5,1)=F24; F(1,6,1)=F24; F(2,2,1)=F24; F(2,3,1)=F4;
F(2,4,1)=F6; F(2,5,1)=F4; F(2,6,1)=F12; F(3,3,1)=F24;
```

```
F(3,4,1)=F6; F(3,5,1)=F12; F(3,6,1)=F4; F(4,4,1)=F4;
F(4,5,1)=F4; F(4,6,1)=F4; F(5,5,1)=F12; F(5,6,1)=F6; F(6,6,1)=F12;
for i=1:6
for j=1:6
F(j,i,1)=F(i,j,1);
end
end
F(1,1,2)=F24; F(1,2,2)=F12; F(1,3,2)=F4; F(1,4,2)=F4;
F(1,5,2)=F6; F(1,6,2)=F12; F(2,2,2)=F120; F(2,3,2)=F12;
F(2,4,2)=F24; F(2,5,2)=F6; F(2,6,2)=F24; F(3,3,2)=F24;
F(3,4,2)=F12; F(3,5,2)=F6; F(3,6,2)=F4; F(4,4,2)=F12;
F(4,5,2)=F4; F(4,6,2)=F6; F(5,5,2)=F4; F(5,6,2)=F4; F(6,6,2)=F12;
for i=1:6
for j=1:6
F(j,i,2)=F(i,j,2);
end
end
F(1,1,3)=F24; F(1,2,3)=F4; F(1,3,3)=F12; F(1,4,3)=F4;
F(1,5,3)=F12; F(1,6,3)=F6; F(2,2,3)=F24; F(2,3,3)=F12;
F(2,4,3)=F12; F(2,5,3)=F4; F(2,6,3)=F6; F(3,3,3)=F120;
F(3,4,3)=F24; F(3,5,3)=F24; F(3,6,3)=F6; F(4,4,3)=F12;
F(4,5,3)=F6; F(4,6,3)=F4; F(5,5,3)=F12; F(5,6,3)=F4; F(6,6,3)=F4;
for i=1:6
for j=1:6
F(j,i,3)=F(i,j,3);
end
```

```
end
Extract element nodal u v and p
Uele(1)=Sol(II);
Uele(2)=Sol(JJ);
Uele(3)=Sol(KK);
Uele(4)=Sol(LL);
Uele(5)=Sol(MM);
Uele(6)=Sol(NN);
Vele(1)=Sol(II+Npoiv);
Vele(2)=Sol(JJ+Npoiv);
Vele(3)=Sol(KK+Npoiv);
Vele(4)=Sol(LL+Npoiv);
Vele(5)=Sol(MM+Npoiv);
Vele(6)=Sol(NN+Npoiv);
Pele(1)=Sol(II+Npoiv+Npoiv);
Pele(2)=Sol(JJ+Npoiv+Npoiv);
Pele(3)=Sol(KK+Npoiv+Npoiv);
Compute [sxx] [sxy] [syx] [syy] matrices
for IA=1:6
for IB=1:6
K11=0;
K22=0;
K12=0;
K21=0;
for i=1:6
for j=1:3
```

```
for k=1:3
for l=1:6
K11=K11+A(IA,i)*B(i,j)*A(IB,l)*B(l,k)*G(j,k);
K22=K22+A(IA,i)*C(i,j)*A(IB,l)*C(l,k)*G(j,k);
K12=K12+A(IA,i)*C(i,j)*A(IB,l)*B(l,k)*G(j,k);
K21=K21+A(IA,i)*B(i,j)*A(IB,l)*C(l,k)*G(j,k);
end
end
end
end
Sxx(IA,IB)=2*Anew*K11+Anew*K22;
Sxy(IA,IB)=Anew*K12;
Syx(IA,IB)=Anew*K21;
Syy(IA,IB)=Anew*K11+2*Anew*K22;
end
end
Compute stabilize term matrices
for IA=1:6
for IB=1:6
Ks11uu=0;
Ks112uu=0;
Ks22vv=0;
Ks222vv=0;
Ks222vu=0;
Ks222uv=0;
Ks12uv=0;
```

```
Ks12uu=0;
Ks12vu=0;
Ks12vv=0;
Ks21vu=0;
Ks21uv=0;
Ks21uu=0;
Ks21vv=0;
for i=1:6
for j=1:3
for k=1:3
for l=1:6
Ks11uu=Ks11uu+A(IA,i)*B(i,j)*A(IB,l)*B(l,k)*G(j,k)*Uele(l)*Uele(i);
Ks112uu=Ks112uu+A(IA,i)*B(l,k)*A(IB,l)*B(i,j)*G(j,k)*2*Uele(l)*Uele(i);
Ks22vv=Ks22vv+A(IA,i)*C(i,j)*A(IB,l)*C(l,k)*G(j,k)*Vele(l)*Vele(i);
Ks222vv=Ks222vv+A(IA,i)*C(l,k)*A(IB,l)*C(i,j)*G(j,k)*2*Vele(l)*Vele(i);
Ks12uv=Ks12uv+A(IA,i)*C(i,j)*A(IB,l)*B(l,k)*G(j,k)*Uele(l)*Vele(i);
Ks12vu=Ks12vu+A(IA,i)*C(l,k)*A(IB,l)*B(i,j)*G(j,k)*Vele(l)*Uele(i);
Ks21vu=Ks21vu+A(IA,i)*B(i,j)*A(IB,l)*C(l,k)*G(j,k)*Vele(l)*Uele(i);
Ks222vu=Ks222vu+A(IA,i)*C(i,j)*A(IB,l)*C(l,k)*G(j,k)*2*Vele(l)*Uele(i);
Ks222uv=Ks222uv+A(IA,i)*C(i,j)*A(IB,l)*C(l,k)*G(j,k)*2*Uele(l)*Vele(i);
Ks12uu=Ks12uu+A(IA,i)*C(i,j)*A(IB,l)*B(l,k)*G(j,k)*Uele(l)*Uele(i);
Ks12vv=Ks12vv+A(IA,i)*C(i,j)*A(IB,l)*B(l,k)*G(j,k)*Vele(l)*Vele(i);
Ks21uv=Ks21uv+A(IA,i)*B(i,j)*A(IB,l)*C(l,k)*G(j,k)*Uele(l)*Vele(i);
Ks21uu=Ks21uu+A(IA,i)*B(i,j)*A(IB,l)*C(l,k)*G(j,k)*Uele(l)*Uele(i);
Ks21vv=Ks21vv+A(IA,i)*B(i,j)*A(IB,l)*C(l,k)*G(j,k)*Uele(l)*Uele(i);
end
```

```
end
end
end
KS11uu(IA,IB)=Ks11uu;
KS112uu(IA,IB)=Ks112uu;
KS22vv(IA,IB)=Ks22vv;
KS222vv(IA,IB)=Ks222vv;
KS222vu(IA,IB)=Ks222vu;
KS222uv(IA,IB)=Ks222uv;
KS12uv(IA,IB)=Ks12uv;
KS12uu(IA,IB)=Ks12uu;
KS12vu(IA,IB)=Ks12vu;
KS12vv(IA,IB)=Ks12vv;
KS21vu(IA,IB)=Ks21vu;
KS21uv(IA,IB)=Ks21uv;
KS21uu(IA,IB)=Ks21uu;
KS21vv(IA,IB)=Ks21vv;
end
end
for i=1:6
for j=1:6
Kse1(i,j)=deltt/2*(KS11uu(i,j)+KS112uu(i,j)+KS22vv(i,j)+
KS12uv(i,j)+KS12vu(i,j) +KS21vu(i,j)+KS21vu(i,j));
Kse2(i,j)=deltt/2*(KS11uu(i,j)+KS22vv(i,j)+
KS12uv(i,j)+KS12uv(i,j)+KS21vu(i,j)+KS21uv(i,j)+KS222vv(i,j));
Kse3(i,j)=deltt/2*(KS12uu(i,j)+KS21uu(i,j)+KS222vu(i,j));
```

```
Kse4(i,j)=deltt/2*(KS12vv(i,j)+KS21vv(i,j)+KS222uv(i,j));
end
end
Compute [Q1] and [Q2]
for IA=1:3
for IB=1:6
Cx=0;
Cy=0;
for i=1:6
for j=1:3
Cx=Cx+A(IB,i)*B(i,j)*G(j,IA);
Cy=Cy+A(IB,i)*C(i,j)*G(j,IA);
end
end
Qx(IA,IB)=Cx/den;
Qy(IA,IB)=Cy/den;
end end
Then the corresponding two matrices on the upper right
for IA=1:3
for IB=1:6
Qxt(IB,IA)= -Qx(IA,IB);
Qyt(IB,IA)= -Qy(IA,IB);
end
end
Compute all matrices associated with the inertia term
for IA=1:6
```

```
for IB=1:6
Cabgxug=0;
Cagbxug=0;
Cagbyvg=0;
Cabgyvg=0;
Cabgxvg=0;
Cabgyug=0;
for i=1:6
for j=1:6
for k=1:6
for l=1:6
for m=1:3
Cabgxug=Cabgxug+A(IA,i)*A(IB,j)*A(k,l)*B(l,m)*F(i,j,m)*Uele(k);
Cagbxug=Cagbxug+A(IA,i)*A(k,j)*A(IB,l)*B(l,m)*F(i,j,m)*Uele(k);
Cagbyvg=Cagbyvg+A(IA,i)*A(k,j)*A(IB,l)*C(l,m)*F(i,j,m)*Vele(k);
Cabgyvg=Cabgyvg+A(IA,i)*A(IB,j)*A(k,l)*C(l,m)*F(i,j,m)*Vele(k);
Cabgxvg=Cabgxvg+A(IA,i)*A(IB,j)*A(k,l)*B(l,m)*F(i,j,m)*Vele(k);
Cabgyug=Cabgyug+A(IA,i)*A(IB,j)*A(k,l)*C(l,m)*F(i,j,m)*Uele(k);
end
end
end
end
end
ABGXUG(IA,IB)=Cabgxug;
AGBXUG(IA,IB)=Cagbxug;
AGBYVG(IA,IB)=Cagbyvg;
```

```
ABGYVG(IA,IB)=Cabgyvg;
ABGXVG(IA,IB)=Cabgxvg;
ABGYUG(IA,IB)=Cabgyug;
end
end
Take in system equation
for i=1:6
for j=1:6
Gxx(i,j)=ABGXUG(i,j)+AGBXUG(i,j)+AGBYVG(i,j)+Sxx(i,j)+Kse1(i,j);
Gyy(i,j)=ABGYVG(i,j)+AGBYVG(i,j)+AGBXUG(i,j)+Syy(i,j)+Kse2(i,j);
Alx(i,j)=ABGXVG(i,j)+Sxy(i,j)+Kse3(i,j);
Aly(i,j)=ABGYUG(i,j)+Syx(i,j)+Kse4(i,j);
end
end
Then the matrix (15x15) on LHS is
for i=1:15
for j=1:15
Akele(i,j)=0;
end
end
for i=1:6
for j=1:6
Akele(i,j)=Gxx(i,j);
Akele(i+6,j+6)=Gyy(i,j);
Akele(i,j+6)=Aly(i,j);
Akele(i+6,j)=Alx(i,j);
```

```
end
for j=1:3
Akele(i,j+12)=Qxt(i,j);
Akele(i+6,j+12)=Qyt(i,j);
end
end
for i=1:3
for j=1:6
Akele(i+12,j)=Qx(i,j);
Akele(i+12,j+6)=Qy(i,j);
end
end
Begin computing the residuals on RHS of element equation
for i=1:6
Term1=0;
Term2=0;
Term3=0;
Term4=0;
Term5=0;
Term6=0;
Term7=0;
Term8=0;
Term9=0;
for j=1:6
Term1=Term1+ABGXUG(i,j)*Uele(j);
Term2=Term2+ABGYUG(i,j)*Vele(j);
```

```
Term4=Term4+Sxx(i,j)*Uele(j);
Term5=Term5+Sxy(i,j)*Vele(j);
Term6=Term6+KS11uu(i,j)*Uele(j);
Term7=Term7+KS12uv(i,j)*Uele(j);
Term8=Term8+KS21vu(i,j)*Uele(j);
Term9=Term9+KS22vv(i,j)*Uele(j);
end
for j=1:3
Term3=Term3+Qxt(i,j)*Pele(j);
end
FX(i)=Term1+Term2+Term3+Term4+Term5+
deltt/2*(Term6+Term7+Term8+Term9);
end
for i=1:6
Term1=0;
Term2=0;
Term3=0;
Term4=0;
Term5=0;
Term6=0;
Term7=0;
Term8=0;
Term9=0;
for j=1:6
Term1=Term1+ABGXVG(i,j)*Uele(j);
Term2=Term2+ABGYVG(i,j)*Vele(j);
```

```
Term4=Term4+Syx(i,j)*Uele(j);
Term5=Term5+Syy(i,j)*Vele(j);
Term6=Term6+KS11uu(i,j)*Vele(j);
Term7=Term7+KS12uv(i,j)*Vele(j);
Term8=Term8+KS21vu(i,j)*Vele(j);
Term9=Term9+KS22vv(i,j)*Vele(j);
end
for j=1:3
Term3=Term3+Qyt(i,j)*Pele(j);
end
FY(i)=Term1+Term2+Term3+Term4+Term5+
deltt/2*(Term6+Term7+Term8+Term9);
end
for i=1:3
Term1=0;
Term2=0;
for j=1:6
Term1=Term1+Qx(i,j)*Uele(j);
Term2=Term2+Qy(i,j)*Vele(j);
end
FI(i)=Term1+Term2;
end
```

Thus the residual vector on RHS of element equation is

```
for i=1:6
Rele(i)=-FX(i);
Rele(i+6)=-FY(i);
```

```
end
for i=1:3
Rele(i+12)=-FI(i);
end
[SysK SysR]=ASSEMBLE(iel,nodes,Akele,Rele,SysK,SysR,
Npoiv,Neq,Nelem);
end
```

B.2.4 Assembly to Global stiffness matrix function "ASSEMBLE"

```
function [SysK SysR]=ASSEMBLE(iel,nodes,Akele,Rele,
SysK,SysR,Npoiv,Neq,Nelem)
Contribution of coeficients assocoated with U, and V velocity
for i=1:6
for j=1:6
II=nodes(iel,i);
JJ=nodes(iel,j);
k=i+6;
l=j+6;
KK=Npoiv+II;
LL=Npoiv+JJ;
SysK(II,JJ)=SysK(II,JJ)+Akele(i,j);
SysK(II,LL)=SysK(II,LL)+Akele(i,l);
SysK(KK,JJ)=SysK(KK,JJ)+Akele(k,j);
SysK(KK,LL)=SysK(KK,LL)+Akele(k,l);
end
end
```

```
Contribution of coeficients assocoated with P pressure
for i=1:6
for j=1:3
II=nodes(iel,i);
JJ=nodes(iel,j);
k=i+6;
l=j+12;
KK=Npoiv+II;
LL=2*Npoiv+JJ;
SysK(II,LL)=SysK(II,LL)+Akele(i,l);
SysK(KK,LL)=SysK(KK,LL)+Akele(k,l);
SysK(LL,II)=SysK(LL,II)+Akele(l,i);
SysK(LL,KK)=SysK(LL,KK)+Akele(l,k);
end
end
Assembly load vector system
Contribution of value with U, and V velocity
for i=1:6
II=nodes(iel,i);
k=i+6;
KK=Npoiv+II;
SysR(II)=SysR(II)+Rele(i);
SysR(KK)=SysR(KK)+Rele(k);
end
Contribution of value with P pressure
for i=1:3
```

```
II=nodes(iel,i);
k=i+12;
KK=2*Npoiv+II;
SysR(KK)=SysR(KK)+Rele(k);
end
Apply boundary conditions function "ApplyBC55"
function [SysM, SysN]=ApplyBC55(Npoiv,Npoip,Neq,Ibcu,SysM,
SysN,Uvel,Vvel,Pres);
Apply boundary condition for nodal U-velocity
IEQ1=1;
IEQ2=Npoiv;
for IEQ=IEQ1:IEQ2
IEQU=IEQ;
if(Ibcu(IEQU) =0)
for IR=1:Neq
if(IR==IEQ)
SysN(IR)=SysN(IR)-SysM(IR,IEQ)*Uvel(IEQU);
SysM(IR,IEQ)=0;
end
end
for IC=1:Neq
SysM(IEQ,IC)=0;
end
SysM(IEQ,IEQ)=1;
SysN(IEQ)=Uvel(IEQU);
end
```

```
end
Apply boundary condition for nodal V-velocity
IEQ1=Npoiv+1;
IEQ2=2*Npoiv;
for IEQ=IEQ1:IEQ2
IEQV=IEQ-Npoiv;
if(Ibcv(IEQV) =0)
for IR=1:Neq
if(IR==IEQ)
SysN(IR)=SysN(IR)-SysM(IR,IEQ)*Vvel(IEQV);
SysM(IR,IEQ)=0;
end
end
for IC=1:Neq
SysM(IEQ,IC)=0;
end
SysM(IEQ,IEQ)=1;
SysN(IEQ)=Vvel(IEQV);
end
end
Apply boundary condition for nodal P-pressure
IEQ1=2*Npoiv+1;
IEQ2=Neq;
for IEQ=IEQ1:IEQ2
IEQP=IEQ-2*Npoiv;
if(Ibcp(IEQP) =0)
```

```
for IR=1:Neq
if(IR==IEQ)
SysN(IR)=SysN(IR)-SysM(IR,IEQ)*Pres(IEQP);
SysM(IR,IEQ)=0;
end
end
for IC=1:Neq
SysM(IEQ,IC)=0;
end
SysM(IEQ,IEQ)=1;
SysN(IEQ)=Pres(IEQP);
end
end
```

REFERENCES

[1] A. J. Chorin, "A numerical method for solving incompressible viscous flow problems," *COMPUTATIONAL PHYSICS*, vol. 135, pp. 118–125, 1997.

[2] M. Matyka, "Solution to two-dimensional incompressible navier-stokes equations with simple, simpler and vorticity-stream function approaches," *Computational Physics Section of Theoretical Physics*, vol. 7, pp. 1–13, 2004.

[3] P. A. Madsen and H. A. Schaffer, "A discussion of artificial compressibility," *Coastal Engineering*, vol. 53, pp. 93–98, 2006.

[4] R. Peyret and T. D. Taylor, *Computational methods for Fluid Flow.* New York: Springer Verlag, 1983.

[5] P. H. Kao and R. J. Yang, "A segregated-implicit scheme for solving the incompressible navier-stokes equations," *Computer and Fluids*, vol. 36, pp. 1159–1161, 2007.

[6] J. L. Steger and P. Kutler, "Implicit finite-difference procedures for the computation of vortex wakes," *AIAA*, vol. 15, pp. 581–590, 1977.

[7] J. L. Chang and D. Kwak, "On the method of pseudo-compressibility for numerically solving incompressible flows," *AIAA*, vol. 15, pp. 84–0252, 1984.

[8] D. Choi and C. L. Merkle, "Application of time-iterative schemes to incompressible flow," *AIAA*, vol. 23, pp. 1518–1524, 1985.

[9] A. Rizzi and L. E. Eliksson, "Computational of inviscid incompressible flow with rotation," *Fluid Mech*, vol. 153, pp. 275–312, 1985.

[10] S. E. Rogers, D. Kwak, and U. Kaul, "On the accuracy of the pseudocompressibility method in solving the incompressible navier-stokes equations." *Appl. Math. Model.*, vol. 11, pp. 35–44, 1987.

[11] N. Massarotti, F. Arpino, and P. Nithiarasu, "Fully explicit and semi-implicit cbs procedures for incompressible flows," *ECCOMAS*, vol. 10, pp. 1–15, 2004.

[12] U. Ghia, K. N. Ghia, and C. T. Shin, "High reynolds number solutions for incompressible flow using the navier-stokes equation and the multigrid method," *Computational Physics*, vol. 48, pp. 387–411, 1982.

[13] J. N. Reddy and D. K. Gatling, *the finite element method in heat transfer and fluid dynamics.* USA: CRC Press, 1994.

[14] J. Donea and A. Huerta, *Finite Element Methods for Flow Problems.* England: John Wiley and Sons Ltd., 2003.

[15] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Dynamics and Heat Transfer.* Washington DC: Taylor and Francis, 1798.

[16] K. H. Huebner, D. L. Dewhirst, D. E. Smith, and T. G. Byrom, *The Finite Element Method For Engineers: Fouth Edition.* New York: John Willey and Sons, 2001.

[17] C. G. du Toit, "Finite element solution of the navier-stokes euations for incompressible flow using a segregated algorithm," *Comput. Methods Appl. Mech. Engrg.*, vol. 151, pp. 131–141, Mar. 1998.

BIOGRAPHICAL STATEMENT

Wanchai Jiajan was born in Mahasarkham, Thailand, in 1983. He received his B.S. degree from Royal Thai Air force Academy, Thailand, in 2005, ($2^n d$ class honor) From 2005 to 2008, he was with the department of Aeronautical Engineering, Royal Thai Air Force Academy as an Instructor. In 2008, he won the scholarship to pursue his first master at The University of Texas at Arlington in Aerospace Engineering.