# THREE DIMENSIONAL MODELLING OF GENERALIZED NEWTONIAN FLUIDS IN DOMAINS INCLUDING OBSTRUCTIONS.

**By**

Noel Rupert Thierry Boukanga

**A Doctoral Thesis**

**Submitted in partial fulfilment of the Requirements for the award of Doctor of Philosophy of Loughborough University.**

**2010**

**Department of Chemical Engineering**

**Loughborough University**

## ACKNOWLEDGEMENT

I have to say special thanks to Professor Vahid Nassehi for his strong support during the course of my study at Loughborough University. Thanks also to my co-supervisor Professor Richard Wakeman for his valuable advices.

I must thank my parents Mr Auguste Boukanga and Mrs Henriette Boukanga, my brothers, sisters, and cousins to be around during these though period of times.

My friends Ganiu Alogba, Ravinder Bhandari, Abu, Kenneth, Amale K. Manga Mabada, Mabiango Mbiangola, and Ahmed Turay are acknowledge for their helps during my time in United Kingdom.

But most of all, I have to say this: thanks to God.

I acknowledge receipt of financial support during the time of my PhD study at Loughborough University.

## ABSTRACT

Three dimensional flow regimes are encountered in many types of industrial flow processes such as filtration, mixing, reaction engineering, polymerization and polymer forming as well as environmental systems. Thus, the analyses of phenomena involved fluid flow are of great importance and have been subject of numerous ongoing research projects. The analysis of these important phenomena can be conducted in laboratory through experiments or simply by using the emerging computational fluid dynamics (CFD) techniques. But when dealing with three dimensional fluid flow problems, the complexities encountered make the analysis via the traditional experimental techniques a daunting task. For this reason, researchers often prefer to use the CFD techniques which with some care taken, often produce accurate and stable results while maintaining cost as low as possible.

Many CFD codes have been developed and tested in the past decades and the results have been successful and thus encouraging researchers to develop new codes and/or improve existing codes for the solutions of real world problems.

In this present project, CFD techniques are used to simulate the fluid flow phenomena of interest by solving the flow governing equations numerically through the use of a personal computer. The aim of this present research is to develop a robust and reliable technique which includes a novel aspect for the solution of three dimensional generalized Newtonian fluids in domains including obstructions, and this must be done bearing in mind that both accuracy and cost efficiency have to be achieved. To this end, the finite element method (FEM) is chosen as the CFD computational method. There are many existing FEM techniques namely the streamline upwind Petrov-Galerkin methods, the streamline diffusion methods, the Taylor-Galerkin methods, among others. But after a thorough analysis of the physical conditions (geometries, governing equations, boundary conditions, assumptions …) of the fluid flow problems to be solve in this project, the appropriate scheme chosen is the UVWP family of the mixed finite element methods. It is scheme originally developed to solve two dimensional fluid flow problems but since the scheme produced accurate and stable results for two dimensional problems, then attempt is made in this present study to develop a new version of the UVWP scheme for the numerical analysis of three dimensional fluid flow problems. But, after some initial results obtained using the developed three dimensional scheme, investigations were made during the course of this study on how to speed up solutions' convergence without affecting the cost efficiency of the scheme. The outcomes of these investigations yield to the development of a novel scheme named the modified three dimensional UVWP scheme. Thus a computer model based on these two numerical schemes (UVWP and the Modified UVWP) is developed, tested, and validated through some benchmark problems, and then the model is used to solve some complicated tests problems in this study. Results obtained are accurate, and stable, moreover, the cost efficiency of the computer model must be mentioned because all the simulations carried out are done using a simple personal computer.

**Keywords:** **C**omputational fluid dynamics, CFD, Computational methods, Fluid flow, Generalized Newtonian fluids, Mathematical modelling, Modelisation, simulation et analyse numerique, Navier-Stokes, Numerical analysis, Stokes.

## LIST OF TABLES

## LIST OF FIGURES

**List of figures**

**List of figures**

# NOMENCLATURE

| | |
|---|---|
| c | Speed of sound in the fluid. |
| g | Body force |
| L | Characteristic length |
| n | Power law index |
| $\hat{n}_x$ | Unit vector in the $x$ – direction |
| $\hat{n}_y$ | Unit vector in the $y$ – direction |
| $\hat{n}_z$ | Unit vector in the $z$ – direction |
| P, p | Pressure |
| R | Specific volume |
| Re | Reynolds number |
| $\dot{S}$ | Source |
| T | Temperature |
| t | Time |
| $\vec{V}$ | Velocity vector |
| U, u | Velocity component in the x-direction |
| V, v | Velocity component in the y-direction |
| W, w | Velocity component in the z-direction |
| $W_J$ | weight (test) function |

## Greek symbols

| | |
|---|---|
| $\alpha$ | Time increment parameter |
| $\alpha_i$ | Coefficient used in the derivation of the trial functions |
| $\Gamma$ | Boundary of the domain |
| $\Gamma_e$ | Boundary of an element domain |
| $\lambda$ | A l arg e number used in the scheme |
| $\delta$ | Kronec ker delta |
| $\varepsilon$ | cov ergencetolerance |
| $\phi_i$ | Shape (trial) function |
| $\dot{\gamma}$ | Shear rate |
| $\eta$ | Apparent vis cos ity |
| $\eta_o$ | Power law consistency index |

| | |
|---|---|
| $\rho$ | Density |
| $\sigma$ | Caushy stress tensor |
| $\tau$ | Stress tensor |
| $\Omega$ | Domain |
| $\Omega_e$ | An element domain within the domain ($\Omega$) |
| $\psi$ | Linear differencial operator |

| | |
|---|---|
| $\nabla$ | Operator nabla |
| $\nabla^2$ | Operator Laplacian |

## CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 Purpose of the present study

Complex three dimensional flow regimes are encountered in many types of industrial flow processes such as filtration, mixing, reaction engineering, polymerization and polymer forming as well as environmental systems. In this project after investigating the effectiveness of different finite element techniques for the modelling of three dimensional viscous flows in three dimensional domains with obstructions, a robust and reliable model has been developed. This model has been applied to solve a number of benchmark problems and its accuracy and validity have been evaluated.

Majority of commercially available CFD packages such as Phoenics, Star-CD, Star-CCM, Fluent, and Flow3D are based on the finite volume method. Despite the rigour of its underpinning concepts the finite volume method does not provide mathematical flexibility required to solve some specific types of problems unless it is essentially based on a finite element approximation on a computational cell level.

Traditionally relatively high cost of three dimensional finite element computations has been regarded as a drawback for this method which has provided a motive for using finite volume approach. One of the main achievements of this work has been to show that very effective low cost three dimensional finite element models of complex flow problems can be developed. Therefore it has been demonstrated that finite element schemes can be extended to complex cases such as those involving multiscale phenomena whilst maintaining computing economy. Another advantage of using the finite element method in this study is that it provides a straightforward way of dealing with non-linear terms in the model equations. This point has been elaborated in later chapters and has shown that a finite element based approach has wider applicability in the modelling of flow processes and can easily be used in cases which involve non-Newtonian fluids. Finite volume based CFD models are generally designed to solve Newtonian flow problems.

## 1.2 Outcome and method of study

As described the main focus of the current project has been the development of a robust, reliable and cost effective computer model for the solution of three-dimensional viscous flow problems in domains including obstructions. The deliverable product of this work is, therefore, a software which can be used in many types of industrial design involving flow processes. The software can be used to simulate fluid flow inside three dimensional domains and to visualize the results in form of plots such as contour, and vector plots.

In order to reach this end, the work undertaken in this current study has involved the following steps:

(i) Formulation of a well posed mathematical problem for the analysis of time dependent generalized Newtonian fluid in three-dimensional domains with and without obstructions.

(ii) Selection of appropriate boundary conditions for simulating the problem of interest.

(iii) The creation of a user-friendly software, with a numerical approach capable of solving the problem defined in parts (i) and (ii).

(iv) Checking and validating the developed model using well known benchmark problems.

(v) Checking the developed computer code through different test cases of generalized Newtonian fluids, and then validating the code by the principle of mass balance.

## 1.3 Structure of the thesis

The present thesis is composed of a total of six chapters with additional parts devoted to a list of references cited within the text and appendices. The thesis begins with the introduction (chapter 1) in which preliminary explanations including processes being modelled, the objective and the significance of the project are given. Reviews of past research works undertook on the modelling of incompressible flow using computational fluid dynamics (CFD), and a brief details of the numerical methods used in CFD are given in chapter 2. In chapter 3 an overview of governing equations of a generalized fluid is given in general, and then from the assumptions made in the current study the governing equations characterizing the physics of the problem to be solved are derived. Chapter 4 provides a detailed discretization procedure of the flow governing equations using the two developed numerical methods. Results and discussions are presented in chapter 5, while the conclusion and recommendations for future work are summarised in the last chapter (chapter 6) of the thesis. A list of all the references used in this present study and the appendices are given after the sixth chapter.

Appendix section provides a detailed manual explaining the implementation of the software developed in this project, additional explanations regarding the structure of the main software and a list of the source code.

# CHAPTER 2

# LITERATURE REVIEW AND BACKGROUND OF THE USED NUMERICAL SCHEMES

## 2.1 Introduction

Processes involving fluid flow are of great importance in many branches of engineering, namely chemical, aeronautical, civil and biochemical engineering. However, some of the most modern technologies which depend on fluid systems such as bio-visco-elastic regimes encountered in medical engineering, non-isothermal elastomeric flows occurring in rubber products manufacturing and particle capturing in filtration are so complex that obtaining empirical design relationships for them by the traditional experimental techniques is not possible. Therefore the development of reliable, robust and cost effective computer models for the simulation of such flow processes have been subject of numerous ongoing research projects. Researchers from different backgrounds have carried out considerable work aiming to reach this end, and most of the works done have been carried out using computational fluid dynamics (CFD) techniques.

CFD is defined as a tool for analysing systems involving fluid, heat transfer and associated phenomena such as chemical reactions by means of computer based simulation (Versteeg and Malalasekera 1995). The main computational methods at the core of the CFD techniques are the finite difference methods (FDM), the finite element methods (FEM), and the finite volume methods (FVM). There are extensive literatures available for each of these techniques.

For FDM see Thomas (1995), Ciarlet *et al.*(1980), Ozi (1994), El-Nakkla (1987), Smith (1985), Wang (c1982), Biggings (1980), Forsythe et al. (1960), Duffy (2006), Bowen (2005), Shashkov and Steinberg (1996), and Voller (c2009), among others.

For FVM see LeVeque (2002), Versteeg and Malalasekera (1995, 2007), Schneider and Raw (1987), Masson and Baliga (1994), and Darbandi and Schneider (1999). For FEM see Masson *et al.*(1994), Darbandi and Schneider (1999), Donea and Huerta (2003), Lohner (2008), Wriggler (c2008), Nassehi (2002), Bochev and Gunzburger (2007), Pian and Wu ( 2006), Brenner (c2002), Chen and Shih (1996), Fenner (1996), Beer and Watson (c1992), Gunzburger (c1989), Girault and Raviart (c1986), Kikuchi (1986), AT Luri *et al.* (c1983), Baker (c1983), Akin (1982), Zienkiewick and Cheung (1965), Zienkiewick and Taylor (1991), Zienkiewick and Codina (1995), Pironneau (1989), and Oden (1972), among others.

FVM is the most commonly used CFD tool for three dimensional fluid flow modelling, however, as mentioned earlier FEM which was primarily regarded as too costly can now be used to generate more flexible and reliable results. Further details about how these computational methods are formulated will be given later in this chapter. In the following section, however, the focus is the review of some of relevant publications on the modelling of fluid flows.

## 2.2 A survey of published literature

In 2005 Nassehi *et al.* (2005) modelled fluid flow through pleated cartridge filter using finite element method (FEM), the computational domain of interest in their work consisted of a combination of free and porous regions and the flow was assumed to be governed by the Stokes equation in the free regime and by the Darcy equation in the porous region. They developed two 2D finite element schemes using perturbed continuity method and mixed formulation for obtaining the solutions of the described problem. These models were in conjunction with equal order and Taylor-Hood interpolation functions, respectively. These two schemes were tested on different problems (using simple and complex geometries) and the results obtained showed that the mixed formulation scheme provided accurate and stable solutions to the problems no matter how complex the selected domain geometry became whilst the perturbed scheme, even though provided accurate and stable solutions for simple geometry cases, yielded spurious and oscillatory pressure solutions for complex geometries.

Four years later, Hanspal *et al*. (2009) investigated fluid flows in cross-flow membrane filtration, the domain of interest in their study was modelled as coupled free/porous regimes and the fluid was governed by Stokes/Darcy equations. In their approaches to the problem, they used the U-V-P family of the mixed finite element method in conjunction with unequal order interpolation functions for velocity and pressure, and then developed a two-dimensional numerical scheme capable of solving the problem. The developed scheme was tested on two problems, the computational domain of the first problem consisted of a rectangular cross-flow membrane filtration with a flat interface between the two regimes (free and porous) whilst for the second problem, the flat interface was replaced by a curved interface placed at the same location than the one from the first problem. Using two different values of permeability ($10^{-6}$ *and* $10^{-12}\,m^2$) they concluded from the solutions obtained that the scheme yield stable and accurate solutions which were validated by calculating the mass balance in both domains. As explained in later chapters these works have been the main starting step of the present project. However, other works have also influenced the development of the present models and are discussed here.

In the late 1990's, Kumar and Naidu (1998) who were interested in simulating nonlinear pulsatile flow of a viscous fluid through a stenosed vessel used the U-V-P scheme with a completely different approach. In their approach, they used the Galerkin weighted residual method to discretize the spatial variables while the temporal variable was discretized through a combination of the explicit Adams-Basuforth formula as predictor and the A-stable implicit trapezoidal rule as the corrector. The computational domain in their study was discretized using a 9-noded Lagrange element and the simulations were carried out for different time steps. Solutions obtained for these time steps were compared with results obtained by previous researchers (O'Brien 1985 and Sako 1962) and the comparison were in good agreement. Zhang (2006) used a different scheme of the FEM, namely the modified pressure correction method to solve incompressible and viscous flow problems on an unstructured Chimera grid. To reach this end, he divided the computational domain into sub domains then solved the governing flow equations (Navier-Stokes) independently before transferring information across the interior boundaries via Scharz method to couple the solutions of each sub domain. In his approach, he discretized the spatial variable using second order upwind scheme while the temporal variable was discretized via the Crank-Nicholson scheme. The field unknowns in his work were interpolated using Rhie-Chow interpolation functions (Demirdzic and Muzaferija 1995). This scheme prevents the unphysical decoupling of the pressure field in the overlapping regions and yields a smoother result. The only drawback of this scheme is an increase in the total number of grid points, which can eventually affect the cost effectiveness of the scheme.

## 2.3 Selection of the computational scheme

Samples described in the previous section are just few among many of the relevant historical papers related to the current project. The aim of the present project is not to make judgments on these different existing numerical methods because each scheme

has its strengths and weaknesses, and as it is proved in the literature, the choice of a particular scheme is problem dependent, that is the physical situation of the fluid will dictate the governing equations to be used. Therefore the selection of a particular methodology for the development of new scheme should be based on the information from previous works plus considerations regarding the main physical features and associated boundary conditions of the problem which needs to be solved.

For the solutions of incompressible viscous flows of generalized Newtonian fluids, many authors, (e.g. Chung, 2002) have shown that the appropriate computational method is the FEM. However the most successful schemes of the FEM are based on the mixed methods, the penalty methods, and the vortex methods. All of these schemes may, however, produce unstable results. To circumvent this instability a condition known as the LBB (Ladyzhenskaya-1969, Babuska-1973, and Brezzi -1974) condition need to be satisfied. Many different strategies for the satisfaction of this condition in the context of the mixed, penalty, and the vortex methods have been developed.

The most common technique adopted in the mixed methods formulation to satisfy the continuity constraint is to use unequal order interpolation function for velocity and pressure. The strategy here consists of choosing the shape functions for pressure one order lower than those for the velocity and to choose the shape functions for pressure to be identical to the test function for the continuity equation. For instance, if the pressure is approximated using linear function then the velocity must be approximated using quadratic function. This yields stable solution but it is computationally expensive.

The main strategy of penalty methods is based on eliminating the pressure term from the momentum equation by setting the pressure as $p = -\lambda\left(\nabla \cdot \vec{V}\right)$ where $\lambda$ is a very large number called the penalty parameter, and then substituting $p = -\lambda\left(\nabla \cdot \vec{V}\right)$ into the momentum equation so that the pressure term will vanish. This will provide a more compact set of working equations from which one will have to firstly solve for
velocity alone, and when all velocity field calculated then the pressure field can be obtained by means of $p = -\lambda\left(\nabla \cdot \vec{V}\right)$. The computational cost of the penalty methods

is less than the computational cost of the mixed methods but when $\lambda$ becomes large, the penalty term will dominate, and this will generate ill-conditioned equations. This will end up by producing instable results, that is, the LBB criterion is violated. Further information about these methods can be found from the following authors Hughes *et al.* (1972), Gunzburger (1989), Bercover (1978), Falk (1975), Cuvelier *et al.* (1986), Teman (1975), and Girault and Raviart (1979).

It is important to mention that the penalty parameter $\lambda$ originates from the stress relation $\tau = \eta D$ from which $\tau$ is pressure, that is force per unit area and D denotes the velocity gradient. Thus $\tau = \eta D$ can be rewritten as

$$p \times \frac{1}{A} = \eta(\nabla \cdot \vec{V})$$

Or $p = A\eta(\nabla \cdot \vec{V})$

Or $p = \lambda(\nabla \cdot \vec{V})$ where $\lambda = A\eta$ with A denoting the cross section area.

The vortex method's strategy is somewhat similar to the penalty method's strategy in the sense that the pressure term is removed from the momentum equation but this time by taking the curl of the momentum equation. This provides a momentum equation in terms of velocity and vorticity vector and/or stream function instead of primitive variables (u, v, w, and p). Hence one can solve the system in the absence of a pressure term. After obtaining a solution the divergence of the original momentum equation is used in order to compute the pressure. The vortex method provide numerical stability but the drawback is that the velocity is coupled with the vorticity vector and this yields a system with seven equations and seven unknowns ($w_1, w_2, w_3, u, v, w, and\ p$) for three-dimensional problems. With an increased number of unknowns the cost efficiency of the method is not good. To alleviate this, one can take the double curl of the momentum equation, that is, when performing the first curl operation on the momentum equation, the result is a momentum equation in term of velocity and vorticity vectors but when a second curl operation is performed on this momentum equation, the vorticity vector will vanish so that the momentum equation will be in term of a single variable (velocity).

Attempt was made to apply this strategy to the governing equations in the present project but after performing the two curl operations, these yielded a momentum equation of high-order (4th order) derivatives for velocity and this required the use of $C^1$ continuous Hermite interpolation functions. Unfortunately, as reported by Nassehi and Petera (1994), these elements lack flexibility and their application in geometrically complex domains

involve elaborate schemes. For this reason it was judged to avoid the use of the vortex method for the discretization for the governing equations of the current project.
Based on Chung's suggestion (and results obtained by researchers like Nassehi *et al* (2004) and Hanspal *et al* (2009)) and bearing in mind that the goal in computational fluid dynamics modelling is to obtain accurate and stable results while minimizing cost finally an FEM based U-V-W-P scheme was developed the governing equations of the present modelling effort. In order to circumvent the problems of spurious and oscillatory pressure field, as mentioned by Nassehi *et al* (2004), which is due to the failure of the enforcement of the incompressibility condition ($\nabla \cdot \vec{V} = 0$), a perturbed form of the incompressible condition $\frac{\partial p}{\rho c^2} + \nabla \cdot \vec{V} = 0$ in conjunction with use of equal order hexahedral isoparametric $C^0$ interpolation functions (figure 2.6) for velocity and pressure field is used.
As it can be noted from the review of the past papers, most examples of finite element based mixed formulations were done for two-dimensional problems. In the current study a new three-dimensional Velocity/Pressure based model capable of providing stable, accurate solutions of fluid flow problems.

Two different schemes of the U-V-W-P have been developed in this work; the first one is based on the direct extension of a two-dimensional scheme to a three-dimensional form, the second scheme is based on a new concept and differs from the usual U-V-P scheme by the addition of a penalty parameter $\lambda$ to the continuity equation. The idea is originated from a method developed by Chang (2002). To solve incompressible viscous flows via FEM, Chung proposed a scheme which is based on the combination of the penalty methods with the mixed methods. He achieved such formulation by replacing the continuity equation with the Galerkin integral of the penalty term $p = -\lambda\left(\nabla \cdot \vec{V}\right)$ and ended up with a Galerkin integral of the continuity equation of the form $\int_{\Omega} \phi_{\alpha}\left( \nabla \cdot \vec{V} + \frac{p}{\lambda} \right) d\Omega = 0$.

Where $\lambda$ is the penalty parameter (a large number)

$\vec{V}$ denotes the velocity vector

p represent the pressure

and $\nabla$ is the gradient operator (nabla operator).

When solving the system of equations consisting of the Galerkin integral of the continuity equation together with the Galerkin integral of the momentum equation, Chung noticed that this scheme provided additional computational stability in

comparison with the solutions obtained from the penalty and mixed methods. Thus based on this idea, the perturbed continuity equation $\frac{\partial p}{\rho c^2} + \nabla \cdot \vec{V} = 0$ will be slightly modified and take the form $\frac{\partial p}{\lambda \rho c^2} + \nabla \cdot \vec{V} = 0$ (variables are as defined above) for the second U-V-W-P scheme developed in this project work.

### 2.3.1 Comparison of the two schemes used in this project

Before comparing the schemes developed in this project a question that must be answered is whether the modified scheme yields stable and accurate results? The U-V-P scheme has been tested on many two-dimensional problems and has provided stable and accurate solutions. The scheme based on its extension to three-dimensional cases (U-V-W-P) is expected to provide similar stable results. Therefore only the stability of the modified scheme needs rigorous investigation. In addition to the stability considerations, during the course of this project, investigations have been made on how to speed up convergence of each scheme without affecting its cost efficiency. One important conclusion of these investigations which can be stated here to prove the validity of effort made to develop an alternative scheme is that the scheme based on the incorporation of the parameter $\lambda$ with the mass balance equation converges much faster than the normal scheme. To demonstrate this point a comparison between the solutions obtained by the U-V-W-P method and the modified U-V-W-P method tested using the same fluid properties is shown below.

Full details about the fluid properties and the boundary conditions are given in chapter 5. For this benchmark problem, it can be noted that after only few iterations, the modified U-V-W-P method yields a converged solution while the U-V-W-P method solution has not yet converged.

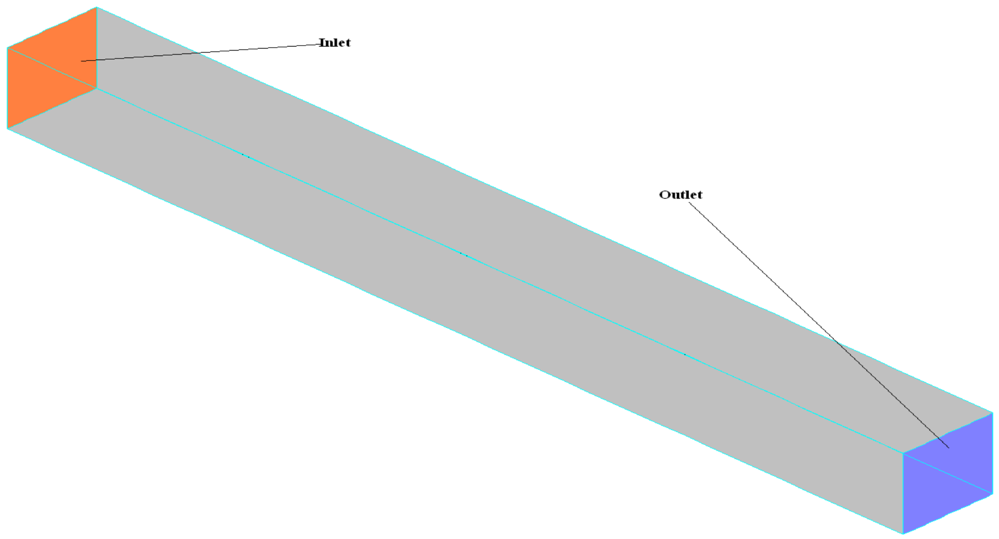


**Figure 2.1:** Computational domain for comparison case.

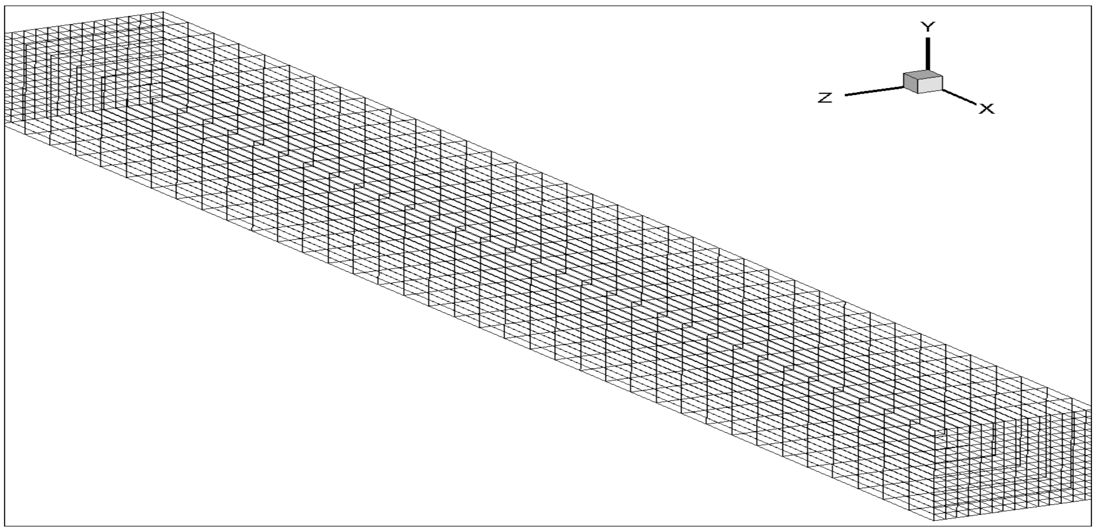


**Figure 2.2:** Finite element mesh for comparison case.

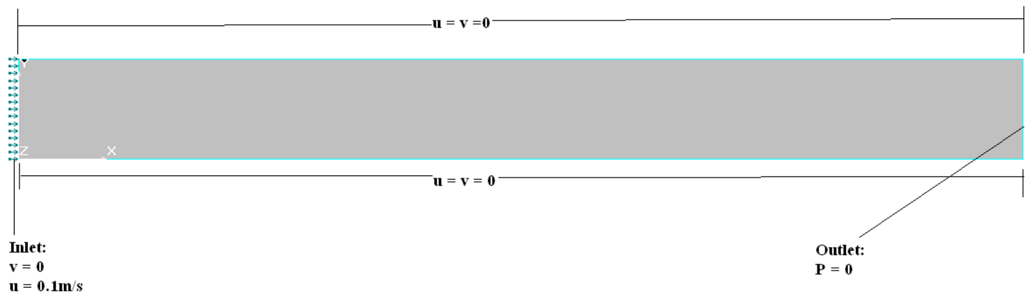


**Figure 2.3a:** 2-D schematic representation of the boundary condition in the xy plane for the comparison case.

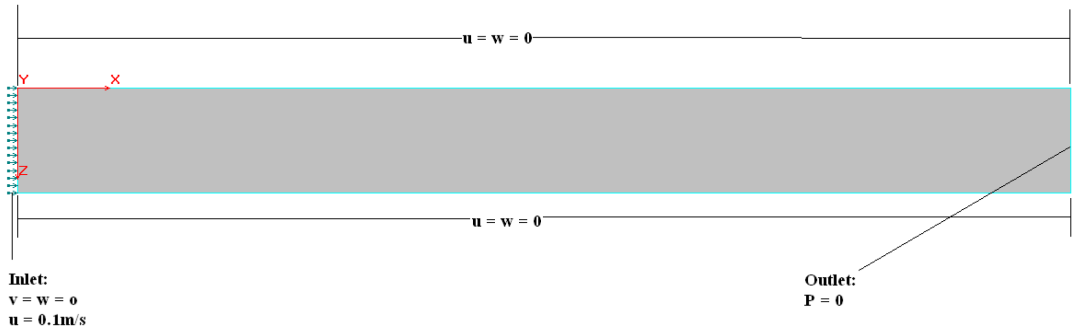


**Figure 2.3b:** 2-D schematic representation of the boundary condition in the xz plane.

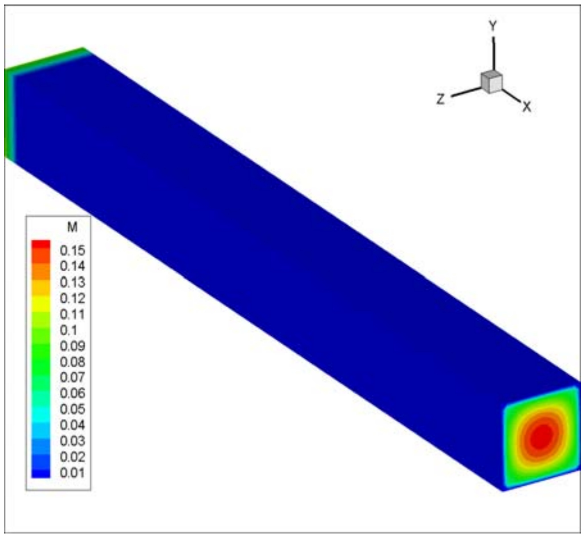


**Figure 2.4a:** Velocity contour plot (Modified U-V-W-P scheme).

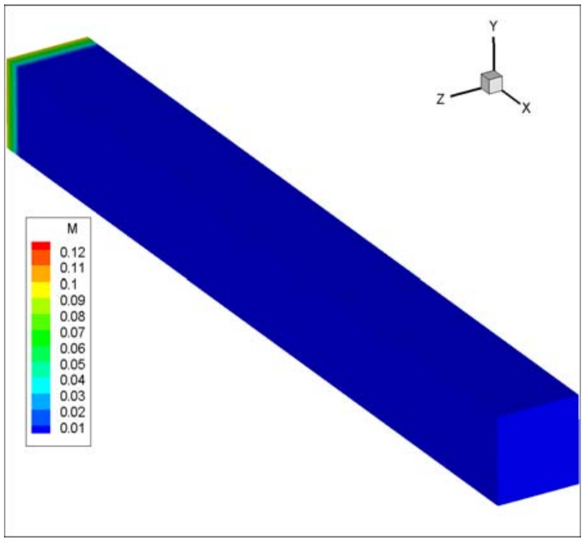


**Figure 2.4b:** Velocity contour plot (U-V-W-P scheme).

Figures 2.4 a, and b represent the contour plots of the velocity for both schemes, and as it can seen from figure 2.4a (Modified U-V-W-P scheme), there is movement of the fluid as excepted at the outlet (multicoloured region representing the expected developed flow profile) while in the case of the U-V-W-P scheme (figure 2.4a), it seems that no fluid is coming out of the exit to the computational domain. This can be interpreted as the solution of the U-V-W-P scheme has yet to reach convergence. This interpretation is confirmed by the plots from figures 2.5.a, and b representing the cross sectional velocity profiles.

These cross sectional plots show the velocity vectors magnitude at a location of z equal 0.05m of the domain. Overall mass balance in both cases has also been checked. Note that both schemes are stable as shown by the expected pressure contours (Figs 2.6a and 2.6b)

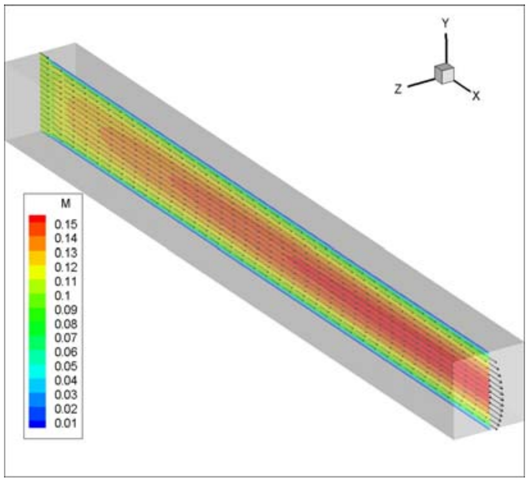


**Figure 2.5a:** Velocity section plot (Modified U-V-W-P scheme)

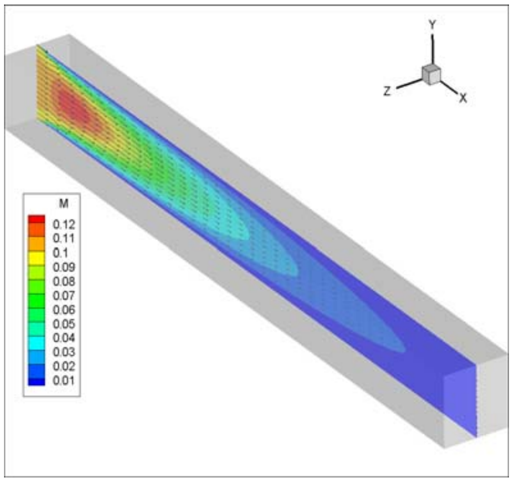


**Figure 2.5b:** Velocity section plot (U-V-W-P scheme)

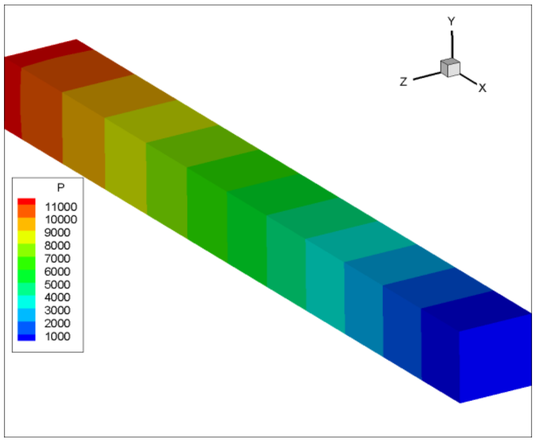


**Figure 2.6a:** Pressure contour plot (Modified U-V-W-P scheme).

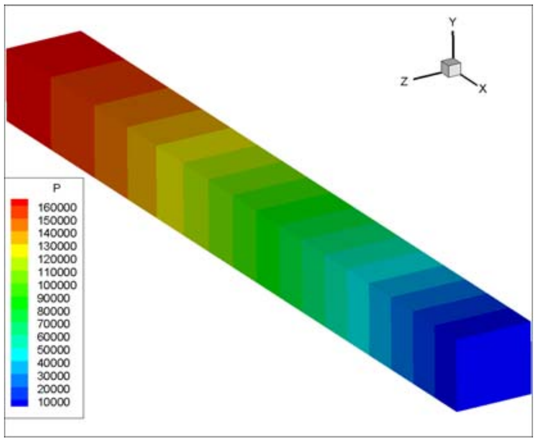


**Figure 2.6b:** Pressure contour plot (U-V-W-P scheme).

Both of the developed three-dimensional schemes are tested extensively using different complex problems in chapter 5. To the best knowledge of the author, none of these two schemes have been used previously to model three-dimensional incompressible highly viscous flow of generalized Newtonian fluids in domains including obstructions.

## 2.4 Mathematical background of the developed schemes

FEM was developed in the late 1950s and was mainly based on the variational formulation and was mainly used for structural analysis. Since then, considerable efforts have been made and especially by mathematicians, engineers, and physicists to extend the use of the FEM to a broad field of continuum mechanics. FEM can now be formulated either using the variational methods or the weighted residual methods.
The variational formulation of the FEM is based on the minimisation of the variational principle of the governing differential equations. This formulation work well for structural analysis but unfortunately cannot be applied to nonlinear fluid mechanics problems due to the non availability of variational principles in exact forms for nonlinear fluid mechanics equations. Due to this reason, the variational approach will not be attempted in this study. Interested reader can obtained further information about this formulation from Curant (1943, 1953), Mura and Koya (1992), and Reddy (1986).

In the weighted residual formulation on the other hand, the strategy is to minimize to zero the residual of the governing equation (minimizing the difference between external forces applied and the internal forces caused by the flow) , and this can be achieved by constructing the inner product of the weighting or test function and the residual. To illustrate this, let the residual R defined as

$$R = \nabla^2 \cdot u(x) + g$$

Where u(x) is the unknown variable function of independent spatial variables
and g is the source/sink term.

Then the weighted residual statement is given as follows

$$\left(W_J, R\right) = \int_{\Omega} W_J \, R \, d\Omega = 0 \qquad j = 1,2,3,...m$$

Or $$\left(W_J, R\right) = \int_{\Omega} W_J \left(\nabla^2 u(x) + g\right) d\Omega = 0 \quad j = 1,2,3,...m$$

Where $W_J$ are linearly independent weights or test functions.

$\Omega$ is a sufficiently smooth closed domain surrounded by a continuous boundary $\Gamma$.

If u(x) is approximated as $u(x) \approx \tilde{u}(x) = \sum_{i=1}^{m} \alpha_i \phi_i(x)$

Where $\alpha_i (i = 1, m)$ are a set of constant coefficients and $\phi_i (i = 1, m)$ denote the trial (interpolation, shape, or basis) function then the weighted residual statement can be written as

$$\left(W_J, R\right) = \int_{\Omega} W_J \left( \nabla^2 \left( \sum_{i=1}^{m} \alpha_i \phi_i(x) \right) + g \right) d\Omega = 0 \quad j = 1,2,3,...m$$

If the test function $W_J$ is chosen to be identical to the shape function $\phi_i$ then the weighted residual method is known as the standard Galerkin method (Zienkiewicz and Morgan 1983). But if it is chosen differently from the shape function then this yields different schemes of the weighted residual methods namely the streamline upwind method (Brooks and Hughes 1982), the streamline upwind Petrov-Galerkin method (Heinrich *et al.* 1977), etc.

For time dependent problems as is the case in the present project, the discretization procedure mentioned above must be preceded, followed, or executed simultaneously with a temporal discretization. In this present work, the temporal discretrization is carried out prior to the spatial discretization and the two procedures are explained in detail below

### 2.4.1 Temporal discretization

Temporal discretization have been subject of a flood of researches and many numerical time integrations techniques have been developed, among them are the continuous space-time method and the discontinuous space-time method (Chung 2002), the θ family of methods (Donea and Huerta 2003, Ames 1992, Lambert 1991, Wait and Mitchell 1985, Zienkiewicz and Taylor 2000, Mitchell and Griffiths 1980, Johnson 1987, and Reddy and Gartling 2000), the Lax-Wendroff method, and the Leap-Frog method (Donea and Huerta 2003) . It is necessary to integrate the temporal variable in order to ensure that information are accurately transported in time to trace transient respond.

In this present work, Taylor-Galerkin discretization (Nassehi 2002, Donea and Huerta 2003, Townsend and Webster 1987) is chosen for the numerical time integration. The technique is based on a truncated Taylor series expansion, and is illustrated by the following example.

Let consider a time-dependent differential equation of the form

$$\frac{\partial \psi(x,t)}{\partial t}+\chi[\psi(x,t)]-\beta=0 \qquad (2.2)$$

Where $\psi$ is a linear differential operator with the respect of the special variables

Then Taylor series expansion of the field unknown $\psi(x,t)$ within the time steps n and n+1 gives

$$\psi^{n+1}=\psi^{n}+\Delta t\left.\frac{\partial \psi}{\partial t}\right|^{n}+\frac{1}{2}\left(\Delta t^{2}\right)\left.\frac{\partial^{2} \psi}{\partial t^{2}}\right|^{n}+\ldots \qquad (2.2)$$

The first order time derivatives term in expansion (2.2) can be found from equation (2.2) as

$$\frac{\partial\psi(x,t)}{\partial t} = \beta - \chi[\psi(x,t)] \qquad (2.3)$$

Differentiating equation (2.3) with respect to time gives the second order time derivatives term in expansion (2.2) as

$$\frac{\partial^2\psi}{\partial t^2} = \frac{\partial}{\partial t}\left(\frac{\partial\psi(x,t)}{\partial t}\right) = \frac{\partial}{\partial t}\{\beta - \chi[\psi(x,t)]\} \qquad (2.4)$$

With a similar procedure all the other order time derivatives term in expansion (2.2) can be found then substituted into equation (2.2) bearing in mind that any first-order temporal term of $\psi(x,t)$ has to be substituted from equations (2.3). This will result by producing a differential equation in terms of spatial variables only which can be discretized using the weighted residual method described previously and summarised by step 1 through step 8 below.

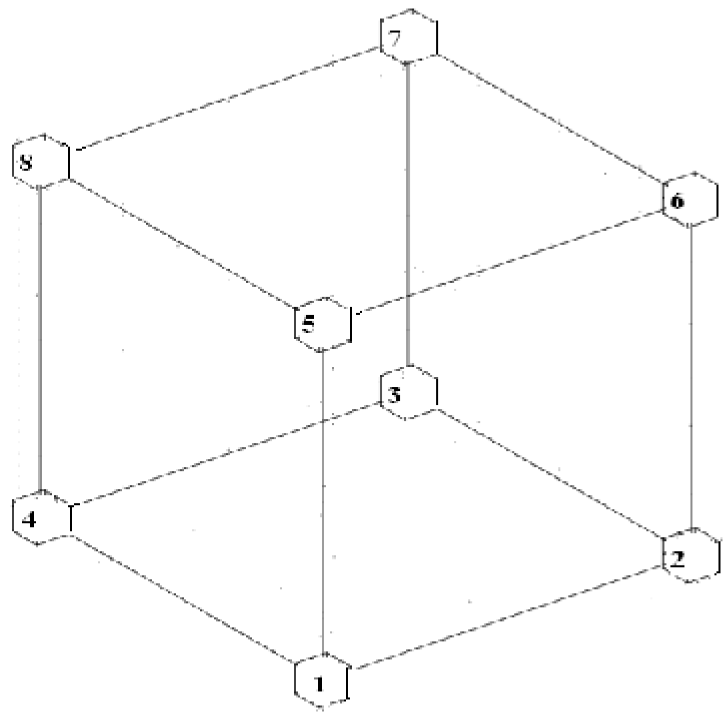


**Figure 2.7:** 8 – nodes **i**soparametric hexahedral element

### 2.4.2 Spatial discretization

Regardless of the scheme used to formulate the finite element methods, the spatial discretization of the governing equation follows these steps.

Step 1. Discretization of the problem domain: The domain $\Omega$ is discretized into Elements limited by a boundary $\Gamma$ is discretised into finite element in the following forms $\Omega = \sum_{e}^{n} \Omega_e$ and $\Gamma = \sum_{e}^{n} \Gamma_e$ .

Step 2. Approximation using trial functions: In this step, one needs to assign nodes to each element, and then selects the appropriate trial function to represent the variation of the unknown functions (pressure, velocity, etc) over the elements. The unknown functions are approximated using the following forms

$$u \approx \tilde{u} = \sum_{i=1}^{n} \phi_i u_i \quad v \approx \tilde{v} = \sum_{i=1}^{n} \phi_i v_i \quad w \approx \tilde{w} = \sum_{i=1}^{n} \phi_i w_i \quad p \approx p = \sum_{i=1}^{n} \phi_i p_i$$

Where $\phi_i \, (i = 1, n)$ denote the trial (interpolation, shape, or basis) function (see figure 2.7).

Step 3. Formulation of the weighted residual statement: in this step, one needs to substitute the interpolated values of the unknown functions found in step2 into the residual of the governing equations, and then construct the inner product of the test function with the residual.

Step 4. Application of Green's theorem: At this stage, one has to apply Green's theorem to all second-order derivatives from the weighted residual statement obtained in step 3 in order to reduce the second-order derivatives to first-order derivatives so that $C^0$ elements can generate an acceptable solution. This process will produce the weak form of the weighted residual statement.

Step 5. Formulation of the elemental stiffness equations: Here, one needs to write an equation corresponding to the weak statement (step 4) for each test function.

Step 6. Assembly of the elemental stiffness equations into a global system of equations: to get the solution of the global system, all the elemental weak statement equations obtained in step 5 must be assemble over their common nodes to form a global system of algebraic equations to be solved.

Step 7. Imposition of the boundary conditions: At this stage, one needs to insert the prescribed values of the unknown functions at the boundaries of $\Omega$ into the global system of algebraic equations obtained in step 6. Redundant equations corresponding to the boundary nodes must be eliminated from the set.

Step 8. Solve the global algebraic system: The global system of algebraic equations obtained in step 7 can now be solved in order to obtained the unknown nodal values of the problem.

## 2.5 Conclusion

In this chapter, a review of past papers on the modelling of generalized Newtonian fluids has been presented and after thorough analysis of them, a choice about which computational method to use in the present project has been made. The chosen computational method is used to discretize the governing flow governing equations representative of the flow regimes considered here in chapter 4 of this thesis.

# CHAPTER 3

# GOVERNING EQUATIONS AND BOUNDARY CONDITIONS

## 3.1 Flow model

Mathematical modelling of fluid flows is based on the solution of partial differential equations governing the physical behaviour of the flows. These governing equations represent mathematical statements of the conservation law of physics, which can be stated as

a) The mass of a fluid is conserved across the entire domain (Conservation of mass).
b) The rate of change of momentum equals the sum of the forces on a fluid particle. This comes from Newton's second law.
c) The rate of change of energy is equal to the sum of the rate of heat addition to and the rate of work done on a fluid particle. This is simply the first law of thermodynamics.

Combining these three statements together with the equation of state and the specified boundary conditions will make the problem to be solved a well-posed mathematical problem representing the physics of the fluid.

In addition to the statements above, in the present study, it is necessary to include a rheological relationship that describes the constitutive behaviour of the fluid. Thus, with the continuum assumption made, that is scalars like density, temperature, and pressure and a vector like velocity vary smoothly and continuously in space and time and adopting a macroscopic viewpoint, in a fixed (stationary or Eulerian) coordinate system (using vector notations), the following equations can be derived.

### 3.1.1 Continuity (Mass balance) equation

The continuity or mass balance equation is the mathematical representation of the statement that: the rate of increase of mass in fluid element equals the net rate of flow of mass into fluid element. This can be mathematically written as

$$\frac{\partial \rho}{\partial t} + \nabla\left(\rho \vec{V}\right) = 0 \qquad (3.1)$$

Where $\nabla$ is the operator nabla (gradient operator)

$\vec{V}$ denotes the velocity vector having u, v, and w as component in the x, y, and z direction respectively.

$\rho$ is the density of the fluid

and t is time.

### 3.1.2 Equation of motion (Momentum equation)

The momentum equation which is Newton's second law states that the rate of change of momentum of a fluid particle equals the sum of the forces acting on the particle. This statement can be mathematically written as

$$\rho \frac{\partial \vec{V}}{\partial t} + \rho \vec{V} \cdot \nabla \vec{V} = \nabla \cdot \vec{\vec{\sigma}} + \rho g \qquad (3.2)$$

Where $\vec{\vec{\sigma}}$ denotes the Cauchy stress tensor

g is the body force per unit volume of fluid.

and $\rho, \vec{V}, \nabla$ are as defined previously.

The Cauchy stress tensor is given as

$$\vec{\vec{\sigma}} = -p\vec{\vec{\delta}} + \vec{\vec{\tau}} \qquad (3.3)$$

Where p is the hydrostatic pressure

$\vec{\vec{\delta}}$ is the unit second-order tensor (Kronecker delta)

and $\vec{\vec{\tau}}$ is the extra stress tensor.

Substituting the expression of the Cauchy stress tensor equation (3.3) in to equation (3.2) yield

$$\rho\frac{\partial \vec{V}}{\partial t} + \rho\vec{V}\cdot\nabla\vec{V} = -\nabla p\delta_{ij} + \nabla\cdot\tau_{ij} + \rho g \qquad (3.4)$$

### 3.1.3 Thermal energy equation

The energy equation is based on the first law of thermodynamics stating that the rate of change of energy of a fluid particle is equal to the rate of heat addition to the fluid particle plus the rate of work done on the particle. This statement can be mathematically written as.

$$\rho c\frac{DT}{Dt} = k\nabla^2 T + \tau : \nabla v + \dot{S} \qquad (3.5)$$

Where c is the specific heat

k is the thermal conductivity

T denotes the temperature

$\nabla^2$ is the Laplacian operator

$\frac{D}{Dt}$ represents the substantial or material derivative

and $\dot{S}$ is the source.

### 3.1.4 Equation of state

It is useful to add the equation of state to the system of equations (mass balance, momentum, and energy) because it allows a linkage between the thermodynamic variables p, $\rho$, and T. It has been observed that in practice most fluid follow the perfect gas law, and that in general, pressure is a function of both density and temperature except in the case of baratropic fluids where pressure is function of density only.

The equation of state is given as

$$p = \rho RT \qquad (3.6)$$

Where R is the specific volume

and p, and T are as defined previously.

### 3.1.5 Constitutive equation

The constitutive equation is a relation between the extra stress $(\tau)$ and the rate of deformation that a fluid experiences as it flows.

But the derivation of a universally applicable constitutive model for non-Newtonian fluids is generally not accepted as it is extremely difficult due to the difficulty that arises in establishing exact quantitative relationship between the microscopic structure of non-Newtonian fluids and their macroscopic properties (Nassehi 2002). There are various formulae used to represent the constitutive equation, see for instance Middleman (1977), Pittman, and Nakazawa (1984), and Carreau (1968), but the one adopted in this study to calculate and update the value of the apparent viscosity is the power law model proposed by Waele (1923), and Ostwald (1925). The power law model is chosen because it is able to describe both shear thinning and shear thickening fluids behaviour. The power law formula is given by

$$\eta = \eta_o (\dot{\gamma})^{n-1} \qquad (3.7)$$

Where $\eta_o$ is the consistency coefficient.

$\eta$ is the apparent viscosity

n is the power law index

$\dot{\gamma}$ denotes the shear rate

For n < 1, the fluid exhibits shear thinning properties.
For n = 1, the fluid shows Newtonian behaviour.
For n >1, the fluid shows shear thickening behaviour.

The shear rate ($\dot{\gamma}$) is calculated using the following relation

$$\dot{\gamma} = \sqrt{2\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)^2 + 2\left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)^2 + 2\left(\frac{\partial w}{\partial z}\right)^2}$$

Where u, v, and w are the three components of the velocity vector $\vec{V}$.

The shear rate ($\dot{\gamma}$) is calculated after each iteration, then the value obtained is used to calculate the apparent viscosity ($\eta$) through the constitutive equation (3.7). This process is repeated until convergence is reached.

Shear-thinning (or pseudoplastic), viscoplastic, shear-thickening (or dilatant) are all the characteristics of the time independent non-Newtonian fluid. It is well established that non-Newtonian fluid can be conveniently grouped into three general classes:

1) The time independent, or purely viscous, or inelastic, or generalized Newtonian fluids in which the rate of shear at any point is determined only by the value of the shear stress at that point at that instant.

2) Time dependent fluids, in which in addition of the criterion mentioned in 1), the relation between shear stress and shear rate depends upon the duration of shearing and their kinematic history.

3) The visco-elastics fluids, in which substances exhibiting characteristics of both ideal fluids and elastic solids and showing partial elastic recovery after deformation.

Interested reader about this classification may refer to the following authors for further information Hou-Cheng Huang *et al.* (1999), Harris (1977), Chhabra and Richardson (1999), Crochet *et al.* 1984, and Nassehi (2002).

## 3.2 Assumptions

The necessary assumptions made in this work are as stated below:

- The incompressible assumption; that is the fluid undergo no changes in volume or density.

- The computational domain is assumed to be isothermal.

- The flow is laminar

- And since the Re is very small ($\mathrm{Re} \langle\langle 1$) in this study, then the convective term (i.e. $\left(\vec{V} \cdot \nabla \vec{V}\right)$) and the body force g from the equation of the motion (3.4) are small and can be omitted Bird., et al (2002), and Nassehi (2002).

Taking these assumptions into consideration, the governing equations (3.1 through 3.7) reduced to

### 3.2.1 Continuity equation

$$\nabla \cdot \vec{\mathrm{V}} = 0 \qquad (3.8)$$

### 3.2.2 Momentum equation

$$\rho \frac{\partial \vec{\mathrm{V}}}{\partial t} = -\nabla p \delta_{ij} + \nabla \cdot \tau_{ij} \qquad (3.9)$$

### 3.2.3 Constitutive equation

$$\eta = \eta_o \left(\dot{\gamma}\right)^{n-1} \qquad (3.10)$$

The first remark one can made from the system of equation given by 3.8 through 3.10 is that there is no pressure term in the continuity equation (3.8) this will make the solution of such system difficult to obtain because the enforcement of the incompressibility conditions (conservation of mass) is difficult. As a result, the computed pressure (p) may be spurious and oscillatory, known as checkerboard type oscillations.

To circumvent this difficulty and satisfy the Ladyzhenskaya (1969), Babuska (1971), and Brezzi (1974) stability condition or simply the LBB condition, the approach adopted in the present project is to replace the continuity equation (3.8) by an equation corresponding to slightly compressible fluids, and it is given as

$$\frac{1}{\rho c^2}\frac{\partial p}{\partial t} + \nabla \cdot \vec{V} = 0 \qquad (3.11)$$

Where c is the speed of sound in the fluid

Equations 3.9 through 3.11 represent the flow model governing equations solved in this project. Using three-dimensional Cartesian coordinates, equations 3.9 and 3.11 can be written as

$$\frac{1}{\rho c^2}\frac{\partial p}{\partial t}+\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \quad (\text{continuity}) \qquad (3.12)$$

Before writing the momentum equation in Cartesian coordinates, it is important to write the component of the Cauchy stress tensor. As mentioned previously, the Cauchy stress tensor is related to the extra stress through the relation $\vec{\vec{\sigma}}=-p\vec{\vec{\delta}}+\vec{\vec{\tau}}$

This can be written in three-dimensional Cartesian coordinates as

$$\vec{\vec{\sigma}}=-p+\vec{\vec{\tau}} \qquad (3.13a)$$

$$\vec{\vec{\sigma}}=-p+\vec{\vec{\tau}} \qquad (3.13b)$$

$$\vec{\vec{\sigma}}=-p+\vec{\vec{\tau}} \qquad (3.13c)$$

Where the normal stresses are given by

$$\tau_{xx}=2\eta\frac{\partial u}{\partial x} \qquad (3.14a)$$

$$\tau_{yy}=2\eta\frac{\partial v}{\partial y} \qquad (3.14b)$$

$$\tau_{zz}=2\eta\frac{\partial w}{\partial z} \qquad (3.14c)$$

and the shear stresses given by

$$\tau_{xy} = \tau_{yx} = \eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \qquad (3.15a)$$

$$\tau_{yz} = \tau_{zy} = \eta\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \qquad (3.15b)$$

$$\tau_{zx} = \tau_{xz} = \eta\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) \qquad (3.15c)$$

Substituting theses expressions of the normal and shear stresses into the momentum equation (3.9) and expanded the result yield

$$\rho\frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(2\eta\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\eta\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] \qquad (3.16a)$$

$$\rho\frac{\partial v}{\partial t} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left(2\eta\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left[\eta\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right] \quad (\text{momentum}) \qquad (3.16b)$$

$$\rho\frac{\partial w}{\partial t} = -\frac{\partial p}{\partial z} + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right)\right] + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right] + \frac{\partial}{\partial z}\left(2\eta\frac{\partial w}{\partial z}\right) \qquad (3.16c)$$

The mass balance, momentum, and constitutive equations as given by 3.12, 3.16a,b,c, and 3.10 respectively represent the final system of equations which will be discretized in chapter 4 and solve in chapter 5.

## 3.3 Boundary and initial conditions

The system of equations representing the flow governing equations (continuity, momentum, and constitutive) as given in section 3.2 does not constitute a consistent system in mathematical viewpoint. To make this system of equations consistent or a well-posed mathematical model, initial and boundary conditions must be specified.

There are three type of boundary conditions; Dirichlet or essential boundary conditions (values of variable specified at boundaries), Neumann or natural boundary conditions (derivatives of variables specified at boundaries), and Cauchy or Robin boundary conditions which is a combination of Dirichlet and Neumann conditions.
The specification of Neumann boundary conditions is the unique future in finite element method (FEM) since Neumann boundary conditions naturally arises in the formulations process of FEM.
The choice of which boundary conditions to apply depends on the type of partial differential equations which can be elliptic, parabolic, hyperbolic or a combination of two or three of them, and the type of flows that can be compressible, incompressible, turbulent, laminar, irrotational, vertical, etc…
In general scalar like pressure may not be specified at the boundaries as it is an implicit variable in an incompressible flow (Lewis., et al 1995) which adjusts itself to deliver a solenoidal velocity field. However, in the case of contained flow, that is specified velocities on all boundaries, the pressure becomes indeterminate and it must be specified at least at one point as a datum.

The initial conditions on the other hand must be specified at time $t = t_o$ in the domain $\Omega$ as mentioned by Hou-Cheng Hang *et al.* (1999) and can take the following form

$$v_i(x_i, t=0)\, v_i^0(x_i) \qquad \text{in } \Omega$$

Taking these remarks into consideration, in the present project, the flow governing equations are solved in conjunction with the following boundary conditions.

### 3.3.1 Inlet boundary conditions

The inlet is placed perpendicular to the x direction for all the simulations carried out in this project. And at the inlet, only Dirichlet type boundary conditions are specified and for velocity variable only. The three component (u, v, and w) of the vector velocity are specified as follow

$u = a$

$v = w = 0$

Where a is small number ( $m \cdot s^{-1}$ )

### 3.3.2 Outlet Boundary conditions

Although there are three type of outlets used for the problems in this project, care was taken that they are all placed far away from geometrical disturbance allowing the flow to reach a fully developed state where no change occurs in the flow directions. Researchers like Nassehi (1998), and Das. et al (2002) have suggested that the imposition of artificial boundary conditions at the outlet might lead to unrealistic numerical results in simulations. Hence based on their suggestions, in this work, no velocity conditions will be specified at the exit, and only a zero datum pressure condition will be specified.

### 3.3.3 Solid walls and blockages

The remaining sides of the geometries and all the faces of the blockages (rectangular or cylindrical) are considered as solid and non permeable walls, on which perfect no-slip conditions are specified, that is the three components (u, v, and w) of the velocity vector are all set equal zero.

### 3.3.4 Initial conditions

For all the problems solved in this project, the inlet boundary conditions will be used as initial conditions for all nodes.

## 3.4 Conclusion

A summary of the derivation of the partial differential equations governing the flow model was presented in this chapter together with the problem of incompressibility enforcement that may occur if care is not taken. Many methods have been developed to overcome this incompressibility enforcement problem and among them are the penalty methods, the vortex transport method, and the mixed finite element method.

In this present project two different finite element techniques are used which differ from the traditional mixed finite element method, penalty, and vortex transport method in the way that the pressure term is not eliminated from the momentum equation but instead an artificial pressure term is added to the continuity equation. The discretization of the derived governing equations will be subject of the next chapter.

# CHAPTER 4

# WORKING EQUATIONS

This chapter is dedicated to the discretization of the governing equations given in the previous chapter. The discretization procedures are followed by briefs description of the solution techniques, the convergence criteria, and the mesh refinements adopted in the present study.

## 4.1 U-V-W-P discretization of the governing equations

The flow governing equations given in chapter 3 are as follow

$$\frac{\partial p}{\partial t} = -\rho c^2\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right) \qquad \text{(continuity)} \qquad (4.1)$$

$$\rho\frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(2\eta\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)\right] \qquad (4.2a)$$

$$\rho\frac{\partial v}{\partial t} = -\frac{\partial p}{\partial y}+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left(2\eta\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right] \quad (\text{momentum}) \quad (4.2b)$$

$$\rho\frac{\partial w}{\partial t} = -\frac{\partial p}{\partial z}+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)\right]+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right]+\frac{\partial}{\partial z}\left(2\eta\frac{\partial w}{\partial z}\right) \qquad (4.2c)$$

All variables are as previously defined.

The first step in the U-V-W-P discretization technique is to normalized the governing equations by setting

$U = u$

$V = v$ for the components of the velocity vector

$W = w$

And $P = \dfrac{p}{\rho}$ for pressure.

Thus one obtains after substitution of these terms into equations (4.1) and (4.2)

For the mass balance equation

$$\frac{\partial P}{\partial t} = -c^2\left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}\right) \tag{4.3}$$

And for the momentum equation

$$\frac{\partial U}{\partial t} = -\frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left(2\eta\frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\eta\left(\frac{\partial W}{\partial x} + \frac{\partial U}{\partial z}\right)\right] \tag{4.4a}$$

$$\frac{\partial V}{\partial t} = -\frac{\partial P}{\partial y} + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left(2\eta\frac{\partial V}{\partial y}\right) + \frac{\partial}{\partial z}\left[\eta\left(\frac{\partial V}{\partial z} + \frac{\partial W}{\partial y}\right)\right] \tag{4.4b}$$

$$\frac{\partial W}{\partial t} = -\frac{\partial P}{\partial z} + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial W}{\partial x} + \frac{\partial U}{\partial z}\right)\right] + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial V}{\partial z} + \frac{\partial W}{\partial y}\right)\right] + \frac{\partial}{\partial z}\left(2\eta\frac{\partial W}{\partial z}\right) \tag{4.4c}$$

Then the discretization continues with the numerical time integration as explained in chapter 2 as follow. The application of the Taylor-Galerkin method to the temporal terms in equation (4.3) and (4.4a, b, and c) gives

$$\frac{\Delta U}{\Delta t}=\frac{U\big|_{n+1}-U\big|_{n}}{\Delta t}=\left.\frac{\partial U}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 U}{\partial t^2}\right|_{n+\alpha\Delta t} \tag{4.5}$$

$$\frac{\Delta V}{\Delta t}=\frac{V\big|_{n+1}-V\big|_{n}}{\Delta t}=\left.\frac{\partial V}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 V}{\partial t^2}\right|_{n+\alpha\Delta t} \tag{4.6}$$

$$\frac{\Delta W}{\Delta t}=\frac{W\big|_{n+1}-W\big|_{n}}{\Delta t}=\left.\frac{\partial W}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 W}{\partial t^2}\right|_{n+\alpha\Delta t} \tag{4.7}$$

$$\frac{\Delta P}{\Delta t}=\frac{P\big|_{n+1}-P\big|_{n}}{\Delta t}=\left.\frac{\partial P}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 P}{\partial t^2}\right|_{n+\alpha\Delta t} \tag{4.8}$$

Where the value of the time increment parameter $\alpha$ must be as $0 \le \alpha \le 1$

The first order derivatives terms ( $\left.\frac{\partial P}{\partial t}\right|_{n+\alpha\Delta t}, \left.\frac{\partial U}{\partial t}\right|_{n+\alpha\Delta t}, \left.\frac{\partial V}{\partial t}\right|_{n+\alpha\Delta t}$, and $\left.\frac{\partial W}{\partial t}\right|_{n+\alpha\Delta t}$ ) from

equations (4.5) through (4.8) can be readily found from the governing equations (4.3) and (4.4a, b, and c) as

For pressure term

$$\left.\frac{\partial P}{\partial t}\right|_{n+\alpha\Delta t}=-c^2\left.\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t} \tag{4.9}$$

And for the velocity components

$$\left.\frac{\partial U}{\partial t}\right|_{n+\alpha\Delta t} = -\left.\frac{\partial P}{\partial x}\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial x}\left(2\eta\frac{\partial U}{\partial x}\right)\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\right|_{n+\alpha\Delta t} \quad (4.10)$$

$$\left.\frac{\partial V}{\partial t}\right|_{n+\alpha\Delta t} = -\left.\frac{\partial P}{\partial y}\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial y}\left(2\eta\frac{\partial V}{\partial y}\right)\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\right|_{n+\alpha\Delta t} \quad (4.11)$$

$$\left.\frac{\partial W}{\partial t}\right|_{n+\alpha\Delta t} = -\left.\frac{\partial P}{\partial z}\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\right|_{n+\alpha\Delta t} + \left.\frac{\partial}{\partial z}\left(2\eta\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t} \quad (4.12)$$

The second-order derivatives terms ($\left.\frac{\partial^2 P}{\partial t^2}\right|_{n+\alpha\Delta t}$, $\left.\frac{\partial^2 U}{\partial t^2}\right|_{n+\alpha\Delta t}$, $\left.\frac{\partial^2 V}{\partial t^2}\right|_{n+\alpha\Delta t}$, and $\left.\frac{\partial^2 W}{\partial t^2}\right|_{n+\alpha\Delta t}$) from equations (4.5) through (4.8) can now be obtained from equations (4.9) through (4.12) as

For pressure term

$$\left.\frac{\partial^2 P}{\partial t^2}\right|_{n+\alpha\Delta t} = \left.\frac{\partial}{\partial t}\left(\frac{\partial P}{\partial t}\right)\right|_{n+\alpha\Delta t} = \left.\frac{\partial}{\partial t}\left[-c^2\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\right]\right|_{n+\alpha\Delta t}$$

$$= -c^2\left.\left[\frac{\partial}{\partial x}\left(\frac{\partial U}{\partial t}\right)+\frac{\partial}{\partial y}\left(\frac{\partial V}{\partial t}\right)+\frac{\partial}{\partial z}\left(\frac{\partial W}{\partial t}\right)\right]\right|_{n+\alpha\Delta t} \quad (4.13)$$

Substituting the expressions of $\frac{\partial U}{\partial t}$, $\frac{\partial V}{\partial t}$, and $\frac{\partial W}{\partial t}$ from equations (4.4a, b, and c) into equation (4.13) gives

$$\begin{aligned}\left.\frac{\partial^2 P}{\partial t^2}\right|_{n+\alpha\Delta t} &= -c^2\left\{\frac{\partial}{\partial x}\left[\left(-\frac{\partial P}{\partial x}\right)+\frac{\partial}{\partial x}\left(2\eta\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\right]\right\}\\ &\quad -c^2\left\{\frac{\partial}{\partial y}\left[\left(-\frac{\partial P}{\partial y}\right)+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left(2\eta\frac{\partial V}{\partial y}\right)+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial W}{\partial y}+\frac{\partial V}{\partial z}\right)\right]\right]\right\} \qquad (4.14)\\ &\quad -c^2\left\{\frac{\partial}{\partial z}\left[\left(-\frac{\partial P}{\partial z}\right)+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial W}{\partial y}+\frac{\partial V}{\partial z}\right)\right]+\frac{\partial}{\partial z}\left(2\eta\frac{\partial W}{\partial z}\right)\right]\right\}\end{aligned}$$

For the first component (U) of the velocity vector one obtains

$$\begin{aligned}\left.\frac{\partial^2 U}{\partial t^2}\right|_{n+\alpha\Delta t} &= \left.\frac{\partial}{\partial t}\left(\frac{\partial U}{\partial t}\right)\right|_{n+\alpha\Delta t} = \frac{\partial}{\partial t}\left\{-\frac{\partial P}{\partial x}+\frac{\partial}{\partial x}\left(2\eta\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial U}{\partial z}+\frac{\partial W}{\partial x}\right)\right]\right\}\\ &= -\frac{\partial}{\partial x}\left(\frac{\partial P}{\partial t}\right)+\frac{\partial}{\partial x}\left[2\eta\frac{\partial}{\partial x}\left(\frac{\partial U}{\partial t}\right)\right]+\frac{\partial}{\partial y}\left\{\eta\left[\frac{\partial}{\partial y}\left(\frac{\partial U}{\partial t}\right)+\frac{\partial}{\partial x}\left(\frac{\partial V}{\partial t}\right)\right]\right\} \qquad (4.15)\\ &\quad +\frac{\partial}{\partial z}\left\{\eta\left[\frac{\partial}{\partial x}\left(\frac{\partial W}{\partial t}\right)+\frac{\partial}{\partial z}\left(\frac{\partial U}{\partial t}\right)\right]\right\}\end{aligned}$$

Using a similar approach, one can obtain for the second component (V) of the velocity vector

$$\begin{aligned}\left.\frac{\partial^2 V}{\partial t^2}\right|_{n+\alpha\Delta t} &= \left.\frac{\partial}{\partial t}\left(\frac{\partial V}{\partial t}\right)\right|_{n+\alpha\Delta t} = \frac{\partial}{\partial t}\left\{-\frac{\partial P}{\partial y}+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left(2\eta\frac{\partial V}{\partial y}\right)+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\right\}\\ &= -\frac{\partial}{\partial y}\left(\frac{\partial P}{\partial t}\right)+\frac{\partial}{\partial x}\left\{\eta\left[\frac{\partial}{\partial y}\left(\frac{\partial U}{\partial t}\right)+\frac{\partial}{\partial x}\left(\frac{\partial V}{\partial t}\right)\right]\right\}+\frac{\partial}{\partial y}\left[2\eta\frac{\partial}{\partial y}\left(\frac{\partial V}{\partial t}\right)\right] \qquad (4.16)\\ &\quad +\frac{\partial}{\partial z}\left\{\eta\left[\frac{\partial}{\partial y}\left(\frac{\partial W}{\partial t}\right)+\frac{\partial}{\partial z}\left(\frac{\partial V}{\partial t}\right)\right]\right\}\end{aligned}$$

And finally for the third component (W) of the velocity vector, one obtains

$$\left.\frac{\partial^2 W}{\partial t^2}\right|_{n+\alpha\Delta t} = \frac{\partial}{\partial t}\left(\frac{\partial W}{\partial t}\right)\Bigg|_{n+\alpha\Delta t} = \frac{\partial}{\partial t}\left\{-\frac{\partial P}{\partial z} + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial V}{\partial z} + \frac{\partial W}{\partial y}\right)\right] + \frac{\partial}{\partial z}\left(2\eta\frac{\partial W}{\partial z}\right)\right\}$$
$$= -\frac{\partial}{\partial z}\left(\frac{\partial P}{\partial t}\right) + \frac{\partial}{\partial x}\left\{\eta\left[\frac{\partial}{\partial z}\left(\frac{\partial U}{\partial t}\right) + \frac{\partial}{\partial x}\left(\frac{\partial W}{\partial t}\right)\right]\right\} \qquad (4.17)$$
$$+\frac{\partial}{\partial y}\left\{\eta\left[\frac{\partial}{\partial y}\left(\frac{\partial W}{\partial t}\right) + \frac{\partial}{\partial z}\left(\frac{\partial V}{\partial t}\right)\right] + \frac{\partial}{\partial z}\left[2\eta\frac{\partial}{\partial z}\left(\frac{\partial W}{\partial t}\right)\right]\right\}$$

Substituting the expressions of $\frac{\partial P}{\partial t}, \frac{\partial U}{\partial t}, \frac{\partial V}{\partial t}$, and $\frac{\partial W}{\partial t}$ from equations (4.3) and (4.4a, b, and c) into equations (4.15) through (4.17), yield equations containing high-order derivative terms. However, as previously published works show (e.g. see Nassehi (2002)), the contributions of these terms with high-order derivatives (that is 3rd or above in the present work) are negligible and hence they can be omitted from equations (4.13) through (4.17). Thus one obtains

$$\left.\frac{\partial^2 P}{\partial t^2}\right|_{n+\alpha\Delta t} = c^2\left(\frac{\partial}{\partial x}\frac{\partial P}{\partial x} + \frac{\partial}{\partial y}\frac{\partial P}{\partial y} + \frac{\partial}{\partial z}\frac{\partial P}{\partial z}\right)\Bigg|_{n+\alpha\Delta t} \qquad (4.18)$$

$$\left.\frac{\partial^2 U}{\partial t^2}\right|_{n+\alpha\Delta t} = c^2\frac{\partial}{\partial x}\left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}\right)\Bigg|_{n+\alpha\Delta t} \qquad (4.19)$$

$$\left.\frac{\partial^2 V}{\partial t^2}\right|_{n+\alpha\Delta t} = c^2\frac{\partial}{\partial y}\left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}\right)\Bigg|_{n+\alpha\Delta t} \qquad (4.20)$$

$$\left.\frac{\partial^2 W}{\partial t^2}\right|_{n+\alpha\Delta t} = c^2\frac{\partial}{\partial z}\left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}\right)\Bigg|_{n+\alpha\Delta t} \qquad (4.21)$$

Substituting the first-order time derivatives from equations (4.9 through 4.12) and the second-order time derivatives from equations (4.18 through 4.21) into equations (4.5 through 4.8) yield

For pressure term

$$\frac{\Delta P}{\Delta t}=\frac{P\big|_{n+1}-P\big|_{n}}{\Delta t}=\left.\frac{\partial P}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 P}{\partial t^2}\right|_{n+\alpha\Delta t}$$
$$=-c^2\left.\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t c^2\left.\left(\frac{\partial}{\partial x}\frac{\partial P}{\partial x}+\frac{\partial}{\partial y}\frac{\partial P}{\partial y}+\frac{\partial}{\partial z}\frac{\partial P}{\partial z}\right)\right|_{n+\alpha\Delta t} \qquad (4.22)$$

For the first component (U) of the velocity vector

$$\frac{\Delta U}{\Delta t}=\frac{U\big|_{n+1}-U\big|_{n}}{\Delta t}=\left.\frac{\partial U}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 U}{\partial t^2}\right|_{n+\alpha\Delta t}$$
$$=\ -\left.\frac{\partial P}{\partial x}\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial x}\left(2\eta\frac{\partial U}{\partial x}\right)\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\right|_{n+\alpha\Delta t}$$
$$+\frac{1}{2}\alpha\Delta t\, c^2\left.\frac{\partial}{\partial x}\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t} \qquad (4.23)$$

For the second component (V) of the velocity vector

$$\frac{\Delta V}{\Delta t}=\frac{V\big|_{n+1}-V\big|_{n}}{\Delta t}=\left.\frac{\partial V}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 V}{\partial t^2}\right|_{n+\alpha\Delta t}$$
$$=-\left.\frac{\partial P}{\partial y}\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial y}\left(2\eta\frac{\partial V}{\partial y}\right)\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\right|_{n+\alpha\Delta t}$$
$$+\frac{1}{2}\alpha\Delta t\,c^2\left.\frac{\partial}{\partial y}\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t} \qquad (4.24)$$

And finally for the third component (W) of the velocity vector

$$\frac{\Delta W}{\Delta t}=\frac{W\big|_{n+1}-W\big|_{n}}{\Delta t}=\left.\frac{\partial W}{\partial t}\right|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\left.\frac{\partial^2 W}{\partial t^2}\right|_{n+\alpha\Delta t}$$
$$=-\left.\frac{\partial P}{\partial z}\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\right|_{n+\alpha\Delta t}+\left.\frac{\partial}{\partial z}\left(2\eta\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t}$$
$$+\frac{1}{2}\alpha\Delta t\,c^2\left.\frac{\partial}{\partial z}\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\right|_{n+\alpha\Delta t} \qquad (4.25)$$

These mark the end of the temporal discretization procedure, and hence one can proceed with the spatial discretization explained in chapter 2. Recall that in the U-V-W-P scheme, velocity and pressure are considered as primitive variables and are discretized as unknowns. In this work, the primitive variables (U, V, W, and P) from equations (4.22) through (4.25) are approximated using 8-noded isoparametric hexahedral element and the approximated variable can be written using the following statements

$$
\begin{aligned}
\mathrm{U} &= \sum_{j=1}^{n} \mathrm{N_j U_j} \cong \mathrm{N_j}\left\{\overset{\bullet}{\mathrm{U}}\right\} \\
\mathrm{V} &= \sum_{j=1}^{n} \mathrm{N_j V_j} \cong \mathrm{N_j}\left\{\overset{\bullet}{\mathrm{V}}\right\} \\
\mathrm{W} &= \sum_{j=1}^{n} \mathrm{N_j W_j} \cong \mathrm{N_j}\left\{\overset{\bullet}{\mathrm{W}}\right\} \qquad (4.26) \\
\mathrm{P} &= \sum_{j=1}^{n} \mathrm{N_j P_j} \cong \mathrm{N_j}\left\{\overset{\bullet}{\mathrm{P}}\right\}
\end{aligned}
$$

Where $\mathrm{N_j}$ (j = 1 …n) are the shape functions (8-noded isoparametric hexahedral element used for the approximation of both velocity and pressure) and n the number of nodes per elements

Substituting the approximated values from the relations given by (4.26) into equations (4.22) through (4.26) and writing the weighted residual finite element statement give

For the first component (U) of the velocity vector

$$
\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n} = - \int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((2\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \qquad (4.27)
\end{aligned}
$$

Where $\mathrm{N_i}$ denote the test function.

Similarly, for the second component (V) of the velocity vector one obtains

$$\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{V}\right\}_{n} = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((2\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \qquad (4.28)
\end{aligned}$$

And for the third component (W) of the velocity vector

$$\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{W}\right\}_{n} = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((2\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \qquad (4.29)
\end{aligned}$$

And with the same procedure the pressure term can be obtained from the mass balance equation as

$$
\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{P}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{P}\right\}_{n} = -\int_{\Omega_e} \Delta t c^2 N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \Delta t c^2 N_i \frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} - \int_{\Omega_e} \Delta t c^2 N_i \frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \frac{1}{2}\alpha \Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha \Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \frac{1}{2}\alpha \Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \qquad (4.30)
\end{aligned}
$$

Seeing that there are some terms of second-order derivatives in equations (4.27) through (4.30), it is necessary to apply Green's theorem to such terms to reduce them to first-order derivatives terms and thus one can ensure inter element continuity. It must be also noted that functions given at time level $n + \alpha\Delta t$ are interpolated using the relation $A\big|_{n+\alpha\Delta t} = \alpha A\big|_{n+1} + (1-\alpha)A\big|_{n}$ (Nassehi 2002).

Thus one obtains for the first component (U) of the velocity

$$
\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n} = \\
&\quad - \int_{\Omega_e} \alpha \Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{P}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{P}\right\}_{n} \\
&\quad - \int_{\Omega_e} \alpha \Delta t 2\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t 2\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n} \\
&\quad + \int_{\Gamma_e} (\alpha \Delta t 2\eta N_i \frac{\partial N_j}{\partial x}) \hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial x}) \hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n} \\
&\quad - \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{V}\right\}_{n} \\
&\quad + \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x}) \hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x}) \hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n}
\end{aligned}
$$

$$
\begin{aligned}
&- \int_{\Omega_e} \alpha\Delta t\eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \{\dot{U}\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t\eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \{\dot{U}\}_n \\
&+ \int_{\Gamma_e} (\alpha\Delta t\eta N_i \frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e \{\dot{U}\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t\eta N_i \frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e \{\dot{U}\}_n \\
&- \int_{\Omega_e} \alpha\Delta t\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \{\dot{U}\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \{\dot{U}\}_n \\
&+ \int_{\Gamma_e} (\alpha\Delta t\eta N_i \frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e \{\dot{U}\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t\eta N_i \frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e \{\dot{U}\}_n
\end{aligned}
$$

$$
\begin{aligned}
&- \int_{\Omega_e} \alpha\Delta t\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{W}\right\}_n \\
&+ \int_{\Gamma_e} (\alpha\Delta t\eta N_i \frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t\eta N_i \frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_n \\
&- \int_{\Omega_e} \frac{1}{2}\alpha^2\Delta t^2 c^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_n \\
&+ \int_{\Gamma_e} (\frac{1}{2}\alpha^2\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} (\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_n \\
&- \int_{\Omega_e} \frac{1}{2}\alpha^2\Delta t^2 c^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&+ \int_{\Gamma_e} (\frac{1}{2}\alpha^2\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} (\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e \left\{\dot{V}\right\}_n \\
&- \int_{\Omega_e} \frac{1}{2}\alpha^2\Delta t^2 c^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{W}\right\}_n \\
&+ \int_{\Gamma_e} (\frac{1}{2}\alpha^2\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z})\hat{n}_x d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} (\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z})\hat{n}_x d\Gamma_e \left\{\dot{W}\right\}_n \qquad (4.31)
\end{aligned}
$$

Similarly for the second velocity component (V) one obtains

$$\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{V}\right\}_{n} =$$

$$- \int_{\Omega_e} \alpha \Delta t N_i \frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{P}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t N_i \frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{P}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{V}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{V}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{U}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t 2\eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t 2\eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t 2\eta N_i \frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{W}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{V}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e \left\{\dot{V}\right\}_{n}$$

$$
\begin{aligned}
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial x})\hat{n}_y d\Gamma_e\left\{\dot{U}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial x})\hat{n}_y d\Gamma_e\left\{\dot{U}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial z})\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial z})\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_n \qquad (4.32)
\end{aligned}
$$

For the third component (W) of the velocity vector, one obtains

$$
\begin{aligned}
&\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{W}\right\}_n=\\
&\quad-\int_{\Omega_e}\alpha\Delta tN_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta tN_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{P}\right\}_n\\
&\quad-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{U}\right\}_n\\
&\quad+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_x d\Gamma_e\left\{\dot{U}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_x d\Gamma_e\left\{\dot{U}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{W}\right\}_n\\
&-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{V}\right\}_n\\
&+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_n\\
&-\int_{\Omega_e}\alpha\Delta t2\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t2\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\alpha\Delta t2\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t2\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e\left\{\dot{U}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e\left\{\dot{U}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_n \qquad (4.33)
\end{aligned}
$$

And finally one can obtain for the mass balance equation

$$
\begin{aligned}
&\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{P}\right\}_n=\\
&\qquad-\int_{\Omega_e}\alpha c^2\Delta tN_i\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)c^2\Delta tN_i\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&-\int_{\Omega_e}\alpha c^2\Delta t N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)c^2\Delta t N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n\\
&-\int_{\Omega_e}\alpha c^2\Delta t N_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)c^2\Delta t N_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{P}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{P}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{P}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{P}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{P}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{P}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{P}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{P}\right\}_n \qquad (4.34)
\end{aligned}
$$

Equations 4.31 through 4.34 can be written in matrix form as

$$
\begin{bmatrix} M_{ij}^{11} & M_{ij}^{12} & M_{ij}^{13} & M_{ij}^{14}\\ M_{ij}^{21} & M_{ij}^{22} & M_{ij}^{23} & M_{ij}^{24}\\ M_{ij}^{31} & M_{ij}^{32} & M_{ij}^{33} & M_{ij}^{34}\\ M_{ij}^{41} & M_{ij}^{42} & M_{ij}^{43} & M_{ij}^{44}\end{bmatrix}^{n+1}\begin{Bmatrix}\dot{U}\\ \dot{V}\\ \dot{W}\\ \dot{P}\end{Bmatrix}^{n+1}=\begin{bmatrix} K_{ij}^{11} & K_{ij}^{12} & K_{ij}^{13} & K_{ij}^{14}\\ K_{ij}^{21} & K_{ij}^{22} & K_{ij}^{23} & K_{ij}^{24}\\ K_{ij}^{31} & K_{ij}^{32} & K_{ij}^{33} & K_{ij}^{34}\\ K_{ij}^{41} & K_{ij}^{42} & K_{ij}^{43} & K_{ij}^{44}\end{bmatrix}^{n}\begin{Bmatrix}\dot{U}\\ \dot{V}\\ \dot{W}\\ \dot{P}\end{Bmatrix}^{n}
$$

$$
+\begin{Bmatrix}C_j^1\\ C_j^2\\ C_j^3\\ C_j^4\end{Bmatrix}^{n+1}+\begin{Bmatrix}D_j^1\\ D_j^2\\ D_j^3\\ D_j^4\end{Bmatrix}^{n} \qquad (4.35)
$$

Where the left hand sides are given as

$$M_{ij}^{11} = \iiint_{\Omega_e} \left\{ N_i N_j + \left( \Delta t \alpha 2\eta + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \right)\left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} \right) + \Delta t \alpha \eta \left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.36}$$

$$M_{ij}^{12} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} \right) \right\} dxdydz \tag{4.37}$$

$$M_{ij}^{13} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.38}$$

$$M_{ij}^{14} = \iiint_{\Omega_e} \left\{ \Delta t \alpha N_i \frac{\partial N_j}{\partial x} \right\} dxdydz \tag{4.39}$$

$$M_{ij}^{21} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} \right) \right\} dxdydz \tag{4.40}$$

$$M_{ij}^{22} = \iiint_{\Omega_e} \left\{ N_i N_j + \left( \Delta t \alpha 2\eta + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \right)\left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} \right) + \Delta t \alpha \eta \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.41}$$

$$M_{ij}^{23} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.42}$$

$$M_{ij}^{24} = \iiint_{\Omega_e} \left\{ \Delta t \alpha N_i \frac{\partial N_j}{\partial y} \right\} dxdydz \tag{4.43}$$

$$M_{ij}^{31} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} \right) \right\} dxdydz \tag{4.44}$$

$$M_{ij}^{32} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} \right) \right\} dxdydz \tag{4.45}$$

$$M_{ij}^{33} = \iiint_{\Omega_e} \left\{ N_i N_j + \left( \Delta t \alpha 2\eta + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \right)\left( \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) + \Delta t \alpha \eta \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} \right) \right\} dxdydz \tag{4.46}$$

$$M_{ij}^{34} = \iiint_{\Omega_e} \left\{ \Delta t \alpha N_i \frac{\partial N_j}{\partial z} \right\} dxdydz \tag{4.47}$$

$$M_{ij}^{41} = \iiint_{\Omega_e} \left\{ \Delta t \alpha c^2 N_i \frac{\partial N_j}{\partial x} \right\} dxdydz \tag{4.48}$$

$$M_{ij}^{42} = \iiint_{\Omega_e} \left\{ \Delta t \alpha c^2 N_i \frac{\partial N_j}{\partial y} \right\} dxdydz \tag{4.49}$$

$$M_{ij}^{43} = \iiint_{\Omega_e} \left\{ \Delta t \alpha c^2 N_i \frac{\partial N_j}{\partial z} \right\} dxdydz \tag{4.50}$$

$$M_{ij}^{44} = \iiint_{\Omega_e} \left\{ N_i N_j + \frac{1}{2} \alpha^2 \Delta t^2 c^2 \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.51}$$

$$\begin{aligned} C_j^1 = \int_{\Gamma_e} \Bigg\{ & \left( \alpha \Delta t 2 \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \\ & + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_x + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_x \\ & + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_y + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.52}$$

$$\begin{aligned} C_j^2 = \int_{\Gamma_e} \Bigg\{ & \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_y + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \alpha \Delta t 2 \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y \\ & + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_x + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_y \\ & + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.53}$$

$$\begin{aligned} C_j^3 = \int_{\Gamma_e} \Bigg\{ & \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x \\ & + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_x + \left( \alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \alpha \Delta t 2 \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \\ & + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_y + \left( \frac{1}{2} \alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.54}$$

$$C_j^4 = \int_{\Gamma_e} \left\{ \left( \frac{1}{2}\alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \frac{1}{2}\alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \frac{1}{2}\alpha^2 \Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \right\} d\Gamma_e \tag{4.55}$$

And the right hand sides as

$$K_{ij}^{11} = \iiint_{\Omega_e} \left\{ N_i N_j - \left( \Delta t(1-\alpha)2\eta + \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \right) \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} \right) - \Delta t(1-\alpha)\eta \left( \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.56}$$

$$K_{ij}^{12} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} \right) \right\} dxdydz \tag{4.57}$$

$$K_{ij}^{13} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial x} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.58}$$

$$K_{ij}^{14} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha) N_i \frac{\partial N_j}{\partial x} \right\} dxdydz \tag{4.59}$$

$$K_{ij}^{21} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \left( \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} \right) \right\} dxdydz \tag{4.60}$$

$$K_{ij}^{22} = \iiint_{\Omega_e} \left\{ N_i N_j - \left( \Delta t(1-\alpha)2\eta + \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \right) \left( \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) - \Delta t(1-\alpha)\eta \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.61}$$

$$K_{ij}^{23} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial y} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \left( \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.62}$$

$$K_{ij}^{24} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha) N_i \frac{\partial N_j}{\partial y} \right\} dxdydz \tag{4.63}$$

$$K_{ij}^{31}=\iiint_{\Omega_e}\left\{-\Delta t(1-\alpha)\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}-\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\left(\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x}\right)\right\}dxdydz \quad (4.64)$$

$$K_{ij}^{32}=\iiint_{\Omega_e}\left\{-\Delta t(1-\alpha)\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}-\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\left(\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}\right)\right\}dxdydz \quad (4.65)$$

$$K_{ij}^{33}=\iiint_{\Omega_e}\left\{N_iN_j-\left(\Delta t(1-\alpha)2\eta+\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\right)\left(\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right)-\Delta t(1-\alpha)\eta\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\right)\right\}dxdydz \quad (4.66)$$

$$K_{ij}^{34}=\iiint_{\Omega_e}\left\{-\Delta t(1-\alpha)N_i\frac{\partial N_j}{\partial z}\right\}dxdydz \quad (4.67)$$

$$K_{ij}^{41}=\iiint_{\Omega_e}\left\{-\Delta t(1-\alpha)c^2N_i\frac{\partial N_j}{\partial x}\right\}dxdydz \quad (4.68)$$

$$K_{ij}^{42}=\iiint_{\Omega_e}\left\{-\Delta t(1-\alpha)c^2N_i\frac{\partial N_j}{\partial y}\right\}dxdydz \quad (4.69)$$

$$K_{ij}^{43}=\iiint_{\Omega_e}\left\{-\Delta t(1-\alpha)c^2N_i\frac{\partial N_j}{\partial z}\right\}dxdydz \quad (4.70)$$

$$K_{ij}^{44}=\iiint_{\Omega_e}\left\{N_iN_j\,\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}+\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right)\right\}dxdydz \quad (4.71)$$

$$\begin{aligned}D_j^1=\int_{\Gamma_e}\Bigg\{&\left((1-\alpha)\Delta t2\eta N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_x+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_y+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_z\\&+\left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_x+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_x+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_x\\&+\left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_y+\left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_z\Bigg\}d\Gamma_e\end{aligned} \quad (4.72)$$

$$\begin{aligned}D_j^2=\int_{\Gamma_e}\Bigg\{&\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_y+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_x+\left((1-\alpha)\Delta t2\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_y\\&+\left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_x+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_z+\left((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_y\\&+\left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_y+\left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_z\Bigg\}d\Gamma_e\end{aligned} \quad (4.73)$$

$$D_j^3 = \int_{\Gamma_e} \left\{ \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x \right.$$
$$+ \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_x + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( (1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \quad (4.74)$$
$$\left. + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_y + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z \right\} d\Gamma_e$$

$$D_j^4 = \int_{\Gamma_e} \left\{ \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y \right.$$
$$\left. + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \right\} d\Gamma_e \quad (4.75)$$

## 4.2 Modified U-V-W-P discretization of the governing equations

The governing equations in the modified U-V-W-P discretization technique differ from those of the U-V-W-P scheme by the fact that a new parameter $\lambda$ (as defined in chapter 2) is introduced to the perturbed form of the continuity equation, and hence the governing equations in this scheme take the following form.

$$\frac{\partial p}{\partial t} = -\lambda \rho c^2 \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \qquad (continuity) \qquad (4.76)$$

The momentum equation remain as it was in the previous scheme and is written as

$$\rho \frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left( 2\eta \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y}\left[ \eta \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] + \frac{\partial}{\partial z}\left[ \eta \left( \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \right] \qquad (4.77a)$$

$$\rho \frac{\partial v}{\partial t} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x}\left[ \eta \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] + \frac{\partial}{\partial y}\left( 2\eta \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial z}\left[ \eta \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \right] \quad (momentum) \qquad (4.77b)$$

$$\rho \frac{\partial w}{\partial t} = -\frac{\partial p}{\partial z} + \frac{\partial}{\partial x}\left[ \eta \left( \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \right] + \frac{\partial}{\partial y}\left[ \eta \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \right] + \frac{\partial}{\partial z}\left( 2\eta \frac{\partial w}{\partial z} \right) \qquad (4.77c)$$

After the normalization of the primitive variables using the expressions given by

U = u

V = v for the components of the velocity vector

W = w

and $P = \frac{p}{\rho}$ for pressure.

One obtains for the continuity equation

$$\frac{\partial P}{\partial t} = -\lambda c^2 \left( \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z} \right) \tag{4.78}$$

and for the momentum equation

$$\frac{\partial U}{\partial t} = -\frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left( 2\eta \frac{\partial U}{\partial x} \right) + \frac{\partial}{\partial y}\left[ \eta \left( \frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} \right) \right] + \frac{\partial}{\partial z}\left[ \eta \left( \frac{\partial W}{\partial x} + \frac{\partial U}{\partial z} \right) \right] \tag{4.79a}$$

$$\frac{\partial V}{\partial t} = -\frac{\partial P}{\partial y} + \frac{\partial}{\partial x}\left[ \eta \left( \frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} \right) \right] + \frac{\partial}{\partial y}\left( 2\eta \frac{\partial V}{\partial y} \right) + \frac{\partial}{\partial z}\left[ \eta \left( \frac{\partial V}{\partial z} + \frac{\partial W}{\partial y} \right) \right] \tag{4.79b}$$

$$\frac{\partial W}{\partial t} = -\frac{\partial P}{\partial z} + \frac{\partial}{\partial x}\left[ \eta \left( \frac{\partial W}{\partial x} + \frac{\partial U}{\partial z} \right) \right] + \frac{\partial}{\partial y}\left[ \eta \left( \frac{\partial V}{\partial z} + \frac{\partial W}{\partial y} \right) \right] + \frac{\partial}{\partial z}\left( 2\eta \frac{\partial W}{\partial z} \right) \tag{4.79c}$$

Following the same procedure as applied for the U-V-W-P scheme (equations (4.5) through (4.21)); one obtains the following Taylor series expansion.
For pressure term

$$\frac{\Delta P}{\Delta t}=\frac{P\big|_{n+1}-P\big|_{n}}{\Delta t}=\frac{\partial P}{\partial t}\bigg|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\frac{\partial^2 P}{\partial t^2}\bigg|_{n+\alpha\Delta t}$$
$$=-\lambda c^2\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\bigg|_{n+\alpha\Delta t}+\frac{1}{2}\lambda\alpha\Delta t c^2\left(\frac{\partial}{\partial x}\frac{\partial P}{\partial x}+\frac{\partial}{\partial y}\frac{\partial P}{\partial y}+\frac{\partial}{\partial z}\frac{\partial P}{\partial z}\right)\bigg|_{n+\alpha\Delta t} \qquad (4.80)$$

For the first component (U) of the velocity vector

$$\frac{\Delta U}{\Delta t}=\frac{U\big|_{n+1}-U\big|_{n}}{\Delta t}=\frac{\partial U}{\partial t}\bigg|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\frac{\partial^2 U}{\partial t^2}\bigg|_{n+\alpha\Delta t}$$
$$=-\frac{\partial P}{\partial x}\bigg|_{n+\alpha\Delta t}+\frac{\partial}{\partial x}\left(2\eta\frac{\partial U}{\partial x}\right)\bigg|_{n+\alpha\Delta t}+\frac{\partial}{\partial y}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]\bigg|_{n+\alpha\Delta t}+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\bigg|_{n+\alpha\Delta t}$$
$$+\frac{1}{2}\lambda\alpha\Delta t\, c^2\frac{\partial}{\partial x}\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\bigg|_{n+\alpha\Delta t} \qquad (4.81a)$$

For the second component (V) of the velocity vector

$$\frac{\Delta V}{\Delta t}=\frac{V\big|_{n+1}-V\big|_{n}}{\Delta t}=\frac{\partial V}{\partial t}\bigg|_{n+\alpha\Delta t}+\frac{1}{2}\alpha\Delta t\frac{\partial^2 V}{\partial t^2}\bigg|_{n+\alpha\Delta t}$$
$$=-\frac{\partial P}{\partial y}\bigg|_{n+\alpha\Delta t}+\frac{\partial}{\partial x}\left[\eta\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)\right]\bigg|_{n+\alpha\Delta t}+\frac{\partial}{\partial y}\left(2\eta\frac{\partial V}{\partial y}\right)\bigg|_{n+\alpha\Delta t}+\frac{\partial}{\partial z}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\bigg|_{n+\alpha\Delta t}$$
$$+\frac{1}{2}\lambda\alpha\Delta t\, c^2\frac{\partial}{\partial y}\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\bigg|_{n+\alpha\Delta t} \qquad (4.81b)$$

And finally for the third component (W) of the velocity vector

$$
\begin{aligned}
\frac{\Delta W}{\Delta t} = \frac{W\big|_{n+1} - W\big|_{n}}{\Delta t} &= \frac{\partial W}{\partial t}\bigg|_{n+\alpha\Delta t} + \frac{1}{2}\alpha\Delta t \frac{\partial^2 W}{\partial t^2}\bigg|_{n+\alpha\Delta t} \\
&= -\frac{\partial P}{\partial z}\bigg|_{n+\alpha\Delta t} + \frac{\partial}{\partial x}\left[\eta\left(\frac{\partial W}{\partial x}+\frac{\partial U}{\partial z}\right)\right]\bigg|_{n+\alpha\Delta t} + \frac{\partial}{\partial y}\left[\eta\left(\frac{\partial V}{\partial z}+\frac{\partial W}{\partial y}\right)\right]\bigg|_{n+\alpha\Delta t} + \frac{\partial}{\partial z}\left(2\eta\frac{\partial W}{\partial z}\right)\bigg|_{n+\alpha\Delta t} \\
&\quad + \frac{1}{2}\lambda\alpha\Delta t\, c^2 \frac{\partial}{\partial z}\left(\frac{\partial U}{\partial x}+\frac{\partial V}{\partial y}+\frac{\partial W}{\partial z}\right)\bigg|_{n+\alpha\Delta t} \qquad (4.81c)
\end{aligned}
$$

Substituting the approximation expressions of the primitive variables given by (4.26) in to equations (4.80) through (4.81a,b, and c) and writing the weighted residual statement yield

For the first component (U) of the velocity vector

$$
\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n} = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((2\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} \\
&\quad + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \qquad (4.82a)
\end{aligned}
$$

Similarly, for the second component (V) of the velocity vector one obtains

$$\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{V}\right\}_{n} = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left(\frac{\partial(\eta) N_j}{\partial x}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left(\frac{\partial(\eta) N_j}{\partial y}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left(\frac{\partial(2\eta) N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left(\frac{\partial(\eta) N_j}{\partial y}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left(\frac{\partial(\eta) N_j}{\partial z}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \tag{4.82b}$$

For the third component (W) of the velocity vector

$$\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{W}\right\}_{n} = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial x}\left((\eta)\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial y}\left((\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \Delta t N_i \frac{\partial}{\partial z}\left((2\eta)\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\lambda\alpha\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t} \tag{4.82c}$$

And one gets from the continuity equation

$$\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{P}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{P}\right\}_{n} = -\int_{\Omega_e} \lambda\Delta t c^2 N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+\alpha\Delta t}$$
$$- \int_{\Omega_e} \lambda\Delta t c^2 N_i \frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_{n+\alpha\Delta t} - \int_{\Omega_e} \lambda\Delta t c^2 N_i \frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{W}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \frac{1}{2}\alpha\lambda\Delta t^2 c^2 N_i \frac{\partial}{\partial x}\left(\frac{\partial N_j}{\partial x}\right) d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} + \int_{\Omega_e} \frac{1}{2}\alpha\lambda\Delta t^2 c^2 N_i \frac{\partial}{\partial y}\left(\frac{\partial N_j}{\partial y}\right) d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t}$$
$$+ \int_{\Omega_e} \frac{1}{2}\alpha\lambda\Delta t^2 c^2 N_i \frac{\partial}{\partial z}\left(\frac{\partial N_j}{\partial z}\right) d\Omega_e \left\{\dot{P}\right\}_{n+\alpha\Delta t} \tag{4.83}$$

The application of Greens' theorem to the second-order derivatives contained in equations (4.82a, b, and) and (4.83) yield

For the first component (U) of the velocity

$$
\begin{aligned}
&\int_{\Omega_e} N_i N_j d\Omega_e \{\dot{U}\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \{\dot{U}\}_n = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \{\dot{P}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \Delta t 2\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \{\dot{U}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \Delta t 2\eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x d\Gamma_e \{\dot{U}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} d\Omega_e \{\dot{V}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_y d\Gamma_e \{\dot{V}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \{\dot{U}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_y d\Gamma_e \{\dot{U}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \{\dot{U}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z d\Gamma_e \{\dot{U}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \{\dot{W}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z d\Gamma_e \{\dot{W}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \frac{1}{2}\alpha \Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \{\dot{U}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \frac{1}{2}\alpha \Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x d\Gamma_e \{\dot{U}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \frac{1}{2}\alpha \Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \{\dot{V}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \frac{1}{2}\alpha \Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_x d\Gamma_e \{\dot{U}\}_{n+\alpha\Delta t} \\
&\quad - \int_{\Omega_e} \frac{1}{2}\alpha \Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} d\Omega_e \{\dot{W}\}_{n+\alpha\Delta t} + \int_{\Gamma_e} \left( \frac{1}{2}\alpha \Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_x d\Gamma_e \{\dot{U}\}_{n+\alpha\Delta t} \qquad (4.84a)
\end{aligned}
$$

The expression of the second component (V) of the velocity vector can be obtained in the similar way as

$$
\int_{\Omega_e} N_i N_j d\Omega_e \{\dot{V}\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \{\dot{V}\}_n = -\int_{\Omega_e} \Delta t N_i \frac{\partial N_j}{\partial y} d\Omega_e \{\dot{P}\}_{n+\alpha\Delta t}
$$

$$
\begin{aligned}
&-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_x d\Gamma_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_x d\Gamma_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\Delta t2\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t2\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\frac{1}{2}\alpha\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\frac{1}{2}\alpha\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_y d\Gamma_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\frac{1}{2}\alpha\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\frac{1}{2}\alpha\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}\\
&-\int_{\Omega_e}\frac{1}{2}\alpha\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\frac{1}{2}\alpha\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_{n+\alpha\Delta t} \qquad (4.84b)
\end{aligned}
$$

And we have for the third component (W) of the velocity vector

$$
\begin{aligned}
&\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{W}\right\}_{n}=-\int_{\Omega_e}\Delta tN_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_{n+\alpha\Delta t}\\
&\quad-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_x d\Gamma_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}\\
&\quad-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_x d\Gamma_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}\\
&\quad-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}
\end{aligned}
$$

$$-\int_{\Omega_e}\Delta t\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\{\dot{V}\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_y d\Gamma_e\{\dot{V}\}_{n+\alpha\Delta t}$$
$$-\int_{\Omega_e}\Delta t2\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\{\dot{W}\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\Delta t2\eta N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_z d\Gamma_e\{\dot{W}\}_{n+\alpha\Delta t}$$

$$-\int_{\Omega_e}\frac{1}{2}\alpha\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x}d\Omega_e\{\dot{U}\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\frac{1}{2}\alpha\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x}\right)\hat{n}_z d\Gamma_e\{\dot{U}\}_{n+\alpha\Delta t}$$
$$-\int_{\Omega_e}\frac{1}{2}\alpha\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\{\dot{V}\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\frac{1}{2}\alpha\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y}\right)\hat{n}_z d\Gamma_e\{\dot{V}\}_{n+\alpha\Delta t}$$
$$-\int_{\Omega_e}\frac{1}{2}\alpha\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\{\dot{W}\}_{n+\alpha\Delta t}+\int_{\Gamma_e}\left(\frac{1}{2}\alpha\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z}\right)\hat{n}_z d\Gamma_e\{\dot{W}\}_{n+\alpha\Delta t} \qquad (4.84c)$$

One obtains from the continuity equation

$$\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}N_iN_jd\Omega_e\left\{\dot{P}\right\}_{n}=-\int_{\Omega_e}\lambda\Delta tc^2N_i\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+\alpha\Delta t}$$
$$-\int_{\Omega_e}\lambda\Delta tc^2N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+\alpha\Delta t}-\int_{\Omega_e}\lambda\Delta tc^2N_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+\alpha\Delta t}$$
$$-\int_{\Omega_e}\frac{1}{2}\alpha\lambda\Delta t^2c^2\frac{\partial N_i}{\partial x}\left(\frac{\partial N_j}{\partial x}\right)d\Omega_e\left\{\dot{P}\right\}_{n+\alpha\Delta t}+\int_{\Omega_e}\left(\frac{1}{2}\alpha\lambda\Delta t^2c^2N_i\frac{\partial N_J}{\partial x}\right)\hat{n}_x\left\{\dot{P}\right\}_{n+\alpha\Delta t}$$
$$-\int_{\Omega_e}\frac{1}{2}\alpha\lambda\Delta t^2c^2\frac{\partial N_i}{\partial y}\left(\frac{\partial N_j}{\partial y}\right)d\Omega_e\left\{\dot{P}\right\}_{n+\alpha\Delta t}+\int_{\Omega_e}\left(\frac{1}{2}\alpha\lambda\Delta t^2c^2N_i\frac{\partial N_J}{\partial y}\right)\hat{n}_y\left\{\dot{P}\right\}_{n+\alpha\Delta t} \qquad (4.85)$$
$$-\int_{\Omega_e}\frac{1}{2}\alpha\lambda\Delta t^2c^2\frac{\partial N_i}{\partial z}\left(\frac{\partial N_j}{\partial z}\right)d\Omega_e\left\{\dot{P}\right\}_{n+\alpha\Delta t}+\int_{\Omega_e}\left(\frac{1}{2}\alpha\lambda\Delta t^2c^2N_i\frac{\partial N_J}{\partial z}\right)\hat{n}_z\left\{\dot{P}\right\}_{n+\alpha\Delta t}$$

After expanding terms containing $n+\alpha\Delta t$ from equations (4.84a, b, and c) through (4.85) using the relation $A\big|_{n+\alpha\Delta t}=\alpha A\big|_{n+1}+(1-\alpha)A\big|_{n}$, one obtains

For the first component (U) of the velocity vector

$$\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{U}\right\}_n =$$

$$- \int_{\Omega_e} \alpha\Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{P}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t N_i \frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{P}\right\}_n$$

$$- \int_{\Omega_e} \alpha\Delta t 2\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t 2\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_n$$

$$+ \int_{\Gamma_e} (\alpha\Delta t 2\eta N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_n$$

$$- \int_{\Omega_e} \alpha\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{V}\right\}_n$$

$$+ \int_{\Gamma_e} (\alpha\Delta t \eta N_i \frac{\partial N_j}{\partial x})\hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x})\hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_n$$

$$- \int_{\Omega_e} \alpha\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{U}\right\}_n$$

$$+ \int_{\Gamma_e} (\alpha\Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e \left\{\dot{U}\right\}_n$$

$$- \int_{\Omega_e} \alpha\Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{U}\right\}_n$$

$$+ \int_{\Gamma_e} (\alpha\Delta t \eta N_i \frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e \left\{\dot{U}\right\}_n$$

$$- \int_{\Omega_e} \alpha\Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{W}\right\}_n$$

$$+ \int_{\Gamma_e} (\alpha\Delta t \eta N_i \frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_n$$

$$- \int_{\Omega_e} \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_n$$

$$+ \int_{\Gamma_e} (\frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} (\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_n$$

$$- \int_{\Omega_e} \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{V}\right\}_n$$

$$+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2 c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e\left\{\dot{V}\right\}_n$$
$$-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2 c^2\lambda\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2\lambda\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n$$
$$+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2 c^2 N_i\lambda\frac{\partial N_j}{\partial z})\hat{n}_x d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_x d\Gamma_e\left\{\dot{W}\right\}_n \qquad (4.86a)$$

With the same procedure the second component (V) of the velocity vector can be written as

$$\int_{\Omega_e}N_iN_j d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}N_iN_j d\Omega_e\left\{\dot{V}\right\}_n=$$
$$-\int_{\Omega_e}\alpha\Delta t N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_n$$
$$-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{V}\right\}_n$$
$$+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{V}\right\}_n$$
$$-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{U}\right\}_n$$
$$+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e\left\{\dot{U}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_x d\Gamma_e\left\{\dot{U}\right\}_n$$
$$-\int_{\Omega_e}\alpha\Delta t2\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t2\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n$$
$$+\int_{\Gamma_e}(\alpha\Delta t2\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t2\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_n$$

$$
\begin{aligned}
&-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_n\\
&-\int_{\Omega_e}\alpha\Delta t\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)\Delta t\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{V}\right\}_n\\
&+\int_{\Gamma_e}(\alpha\Delta t\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}((1-\alpha)\Delta t\eta N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x})\hat{n}_y d\Gamma_e\left\{\dot{U}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x})\hat{n}_y d\Gamma_e\left\{\dot{U}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{V}\right\}_n
\end{aligned}
$$

$$
\begin{aligned}
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_y d\Gamma_e\left\{\dot{W}\right\}_n \qquad (4.86b)
\end{aligned}
$$

For the third component (W) of the velocity vector, one obtains

$$\int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} N_i N_j d\Omega_e \left\{\dot{W}\right\}_{n} =$$

$$- \int_{\Omega_e} \alpha \Delta t N_i \frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{P}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t N_i \frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{P}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{U}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z}) \hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z}) \hat{n}_x d\Gamma_e \left\{\dot{U}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{W}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial x}) \hat{n}_x d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x}) \hat{n}_x d\Gamma_e \left\{\dot{W}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega_e \left\{\dot{W}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial y}) \hat{n}_y d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y}) \hat{n}_y d\Gamma_e \left\{\dot{W}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{V}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{V}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t \eta N_i \frac{\partial N_j}{\partial z}) \hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z}) \hat{n}_y d\Gamma_e \left\{\dot{V}\right\}_{n}$$

$$- \int_{\Omega_e} \alpha \Delta t 2\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{W}\right\}_{n+1} - \int_{\Omega_e} (1-\alpha)\Delta t 2\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} d\Omega_e \left\{\dot{W}\right\}_{n}$$

$$+ \int_{\Gamma_e} (\alpha \Delta t 2\eta N_i \frac{\partial N_j}{\partial z}) \hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_{n+1} + \int_{\Gamma_e} ((1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial z}) \hat{n}_z d\Gamma_e \left\{\dot{W}\right\}_{n}$$

$$- \int_{\Omega_e} \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n+1} - \int_{\Omega_e} \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} d\Omega_e \left\{\dot{U}\right\}_{n}$$

$$+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e\left\{\dot{U}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x})\hat{n}_z d\Gamma_e\left\{\dot{U}\right\}_n$$

$$-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n$$

$$+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_z d\Gamma_e\left\{\dot{V}\right\}_n$$

$$-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n$$

$$+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{W}\right\}_n \qquad (4.86c)$$

And finally from the continuity equation, one obtains

$$\int_{\Omega_e}N_iN_j d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}N_iN_j d\Omega_e\left\{\dot{P}\right\}_n=$$

$$-\int_{\Omega_e}\alpha c^2\lambda\Delta tN_i\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)c^2\lambda\Delta tN_i\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{U}\right\}_n$$

$$
\begin{aligned}
&-\int_{\Omega_e}\alpha c^2\lambda\Delta t N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)c^2\lambda\Delta t N_i\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{V}\right\}_n\\
&-\int_{\Omega_e}\alpha c^2\lambda\Delta t N_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_{n+1}-\int_{\Omega_e}(1-\alpha)c^2\lambda\Delta t N_i\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{W}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}d\Omega_e\left\{\dot{P}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{P}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial x})\hat{n}_x d\Gamma_e\left\{\dot{P}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}d\Omega_e\left\{\dot{P}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{P}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial y})\hat{n}_y d\Gamma_e\left\{\dot{P}\right\}_n\\
&-\int_{\Omega_e}\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{P}\right\}_{n+1}-\int_{\Omega_e}\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}d\Omega_e\left\{\dot{P}\right\}_n\\
&+\int_{\Gamma_e}(\frac{1}{2}\alpha^2\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{P}\right\}_{n+1}+\int_{\Gamma_e}(\frac{1}{2}\alpha(1-\alpha)\Delta t^2c^2\lambda N_i\frac{\partial N_j}{\partial z})\hat{n}_z d\Gamma_e\left\{\dot{P}\right\}_n \qquad (4.87)
\end{aligned}
$$

Equations (4.86a, b, and c) and (4.87) can be written in matrix form given by (4.35) previously but this time with the left hand side given by

$$M_{ij}^{11}=\iiint_{\Omega_e}\left\{N_iN_j+\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\right)+\Delta t\alpha\eta\left(2\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}+\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}+\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right)\right\}dxdydz \qquad (4.88)$$

$$M_{ij}^{12}=\iiint_{\Omega_e}\left\{\Delta t\alpha\eta\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}+\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}\right)\right\}dxdydz \qquad (4.89)$$

$$M_{ij}^{13}=\iiint_{\Omega_e}\left\{\Delta t\alpha\eta\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x}+\frac{1}{2}\alpha^2\Delta t^2c^2\lambda\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}\right)\right\}dxdydz \qquad (4.90)$$

$$M_{ij}^{14}=\iiint_{\Omega_e}\left\{\Delta t\alpha N_i\frac{\partial N_j}{\partial x}\right\}dxdydz \qquad (4.91)$$

$$M_{ij}^{21} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} \right) \right\} dxdydz \tag{4.92}$$

$$M_{ij}^{22} = \iiint_{\Omega_e} \left\{ N_i N_j + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} \right) + \Delta t \alpha \eta \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + 2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.93}$$

$$M_{ij}^{23} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.94}$$

$$M_{ij}^{24} = \iiint_{\Omega_e} \left\{ \Delta t \alpha N_i \frac{\partial N_j}{\partial y} \right\} dxdydz \tag{4.95}$$

$$M_{ij}^{31} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} \right) \right\} dxdydz \tag{4.96}$$

$$M_{ij}^{32} = \iiint_{\Omega_e} \left\{ \Delta t \alpha \eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} \right) \right\} dxdydz \tag{4.97}$$

$$M_{ij}^{33} = \iiint_{\Omega_e} \left\{ N_i N_j + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) + \Delta t \alpha \eta \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + 2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.98}$$

$$M_{ij}^{34} = \iiint_{\Omega_e} \left\{ \Delta t \alpha N_i \frac{\partial N_j}{\partial z} \right\} dxdydz \tag{4.99}$$

$$M_{ij}^{41} = \iiint_{\Omega_e} \left\{ \Delta t \alpha c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right\} dxdydz \tag{4.100}$$

$$M_{ij}^{42} = \iiint_{\Omega_e} \left\{ \Delta t \alpha c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right\} dxdydz \tag{4.101}$$

$$M_{ij}^{43} = \iiint_{\Omega_e} \left\{ \Delta t \alpha c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right\} dxdydz \tag{4.102}$$

$$M_{ij}^{44} = \iiint_{\Omega_e} \left\{ N_i N_j + \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.103}$$

$$K_{ij}^{11} = \iiint\limits_{\Omega_e} \left\{ N_i N_j - \left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda\right)\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\right) - \Delta t(1-\alpha)\eta\left(2\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right)\right\} dxdydz \quad (4.104)$$

$$K_{ij}^{12} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y}\right)\right\} dxdydz \quad (4.105)$$

$$K_{ij}^{13} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z}\right)\right\} dxdydz \quad (4.106)$$

$$K_{ij}^{14} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha) N_i \frac{\partial N_j}{\partial x}\right\} dxdydz \quad (4.107)$$

$$K_{ij}^{21} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial y} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial x}\right)\right\} dxdydz \quad (4.108)$$

$$K_{ij}^{22} = \iiint\limits_{\Omega_e} \left\{ N_i N_j - \left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda\right)\left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\right) - \Delta t(1-\alpha)\eta\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + 2\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right)\right\} dxdydz \quad (4.109)$$

$$K_{ij}^{23} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left(\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z}\right)\right\} dxdydz \quad (4.110)$$

$$K_{ij}^{24} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha) N_i \frac{\partial N_j}{\partial y}\right\} dxdydz \quad (4.111)$$

$$K_{ij}^{31} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial z} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left(\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial x}\right)\right\} dxdydz \quad (4.112)$$

$$K_{ij}^{32} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha)\eta \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial z} - \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left(\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial y}\right)\right\} dxdydz \quad (4.113)$$

$$K_{ij}^{33} = \iiint\limits_{\Omega_e} \left\{ N_i N_j - \left(\frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda\right)\left(\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right) - \Delta t(1-\alpha)\eta\left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + 2\frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}\right)\right\} dxdydz \quad (4.114)$$

$$K_{ij}^{34} = \iiint\limits_{\Omega_e} \left\{ -\Delta t(1-\alpha) N_i \frac{\partial N_j}{\partial z}\right\} dxdydz \quad (4.115)$$

$$K_{ij}^{41} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)c^2\lambda N_i \frac{\partial N_j}{\partial x} \right\} dxdydz \tag{4.116}$$

$$K_{ij}^{42} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)c^2\lambda N_i \frac{\partial N_j}{\partial y} \right\} dxdydz \tag{4.117}$$

$$K_{ij}^{43} = \iiint_{\Omega_e} \left\{ -\Delta t(1-\alpha)c^2\lambda N_i \frac{\partial N_j}{\partial z} \right\} dxdydz \tag{4.118}$$

$$K_{ij}^{44} = \iiint_{\Omega_e} \left\{ N_i N_j \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda \left( \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right) \right\} dxdydz \tag{4.119}$$

$$\begin{aligned} C_j^1 = \int_{\Gamma_e} \Bigg\{ & \left( \alpha\Delta t 2\eta N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_x + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_y + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_z \\ & + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_x + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_x + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_x \\ & + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_y + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.120}$$

$$\begin{aligned} C_j^2 = \int_{\Gamma_e} \Bigg\{ & \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_y + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_x + \left( \alpha\Delta t 2\eta N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_y \\ & + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_x + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_z + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_y \\ & + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_y + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.121}$$

$$\begin{aligned} C_j^3 = \int_{\Gamma_e} \Bigg\{ & \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_z + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_z + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial x} \right)\hat{n}_x \\ & + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_x + \left( \alpha\Delta t\eta N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_y + \left( \alpha\Delta t 2\eta N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_z \\ & + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right)\hat{n}_y + \left( \frac{1}{2}\alpha^2\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right)\hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.122}$$

$$C_j^4 = \int_{\Gamma_e} \left\{ \left( \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \frac{1}{2}\alpha^2 \Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \right\} d\Gamma_e \tag{4.123}$$

$$\begin{aligned} D_j^1 = \int_{\Gamma_e} \Bigg\{ & \left( (1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \\ & + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_x + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_x \\ & + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_y + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.124}$$

$$\begin{aligned} D_j^2 = \int_{\Gamma_e} \Bigg\{ & \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_y + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( (1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y \\ & + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_x + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_y \\ & + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.125}$$

$$\begin{aligned} D_j^3 = \int_{\Gamma_e} \Bigg\{ & \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_z + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x \\ & + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_x + \left( (1-\alpha)\Delta t \eta N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( (1-\alpha)\Delta t 2\eta N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \\ & + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_y + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_z \Bigg\} d\Gamma_e \end{aligned} \tag{4.126}$$

$$D_j^4 = \int_{\Gamma_e} \left\{ \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial x} \right) \hat{n}_x + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial y} \right) \hat{n}_y + \left( \frac{1}{2}\alpha(1-\alpha)\Delta t^2 c^2 \lambda N_i \frac{\partial N_j}{\partial z} \right) \hat{n}_z \right\} d\Gamma_e \tag{4.127}$$

## 4.3 Solution procedure

The discretization of the governing equations by the two developed schemes yields systems of algebraic equations which need to be solved; this can be achieved by using direct methods or iterative methods. Thus it is informative to briefly examine the difference, advantages and disadvantages of these two methods and then select an appropriate method to adopt in this project.

Direct methods regroup techniques such as Cramer's rule, Gaussian elimination, Cholesky method, Thomas algorithm (TDMA), Runge-Kutta method, among others. These groups of solution techniques are mainly suitable for linear system of equations, have the disadvantages of being time consuming and are susceptible to round-off errors, which in case of large system of equations can lead to unacceptable results. Interested reader about these techniques may refer to Butcher (1993), Duff (1986), Shampine (1994), Sewell (1988), Lapidus (1971), Greenspan (1960), and Curtis and Patrick (1994), among others.

The iterative methods on the other hand have the unique advantage, that round-off errors in each step are corrected in the subsequent step, they can be used to solved both linear and nonlinear systems of equations, and when the coefficient matrix is sparse, they are non-time consuming and economical in term of computer storage. Many iterative techniques have been developed and among them are the Jacobi method, Gauss-Seidel method, Alternating direction implicit (ADI) method, conjugate gradient methods (CGM), domain decomposition methods (DDM) and the generalized minimal residual (GMRES) methods. Further information about these techniques can be found from authors like Greenbaun (1997), Axelsson (1994), Hageman (1981), Traub (1964), Varga (1962), Wachspress (1966), Dahlquist (1974), Saad (1996), Hestenes and Stiefel (1952), Concus *et al.* (1976), Kershaw (1978), Press *et al.* (1992), Glowinski and Wheeler (1987), Lions (1988), and Scharz (1869).

The solution technique adopted in the present project is the frontal method which is a modification of the Gaussian elimination. The technique was developed to tackle the problem of total

assembly of elemental stiffness equations experienced with direct methods. The frontal method readily avoid such problem by stepwise reduction of the total matrix (non-zero band) in a Gaussian elimination procedure, further information about this technique can be obtained from Iron (1970), Platonov and Trivailo (1979), Light and Luxmoore (1977), Beer and Haas (1982), Postnov *et al.* (1979), Forsaith and Moler (1969), Duff and Reid (1983), Duff *et al.* (1986), and Hood (1976).

## 4.4 Convergence of the solution

In the present study, the convergence of the solutions is checked using the calculated ratio of the difference of the Euclidian norm (Lapidus and Pinder, 1982) between successive iterations to the norm of the solution. This is given by

$$\sqrt{\frac{\sum_{i=1}^{N}\left|X_i^{r+1}-X_i^{r}\right|^2}{\sum\left|X_i^{r+1}\right|^2}} \leq \varepsilon \qquad (4.128)$$

Where r denotes the number of the iteration cycle

N is the total number of degrees of freedom

X are the field unknowns

And $\varepsilon$ denotes the convergence tolerance value.

The criterion given by (4.128) is used for both pressure and velocity components in separated calculations and converged solution is obtained when both sets of results satisfy this criterion.

## 4.5 Mesh refinements

The aim of solving real world problems using the computational fluid dynamics techniques is to obtain desired solutions as accurately as possible while maintaining cost as efficient as possible. But achieving both cost efficiency and accuracy is often no trivial matter, especially when one is constrained to use a fixed computational method and limited computer resources like is the case experienced during the course of this study, in which the researcher is confronted to use a fixed computational scheme (the UVWP method) and a limited computer resources (a Pentium (R) IV 3.00GHz). Given this circumstance, the best strategy to adopt to obtain stable and accurate results at low cost is the refinement of the computational grid known as mesh refinements. Mesh refinements also can be used for testing the convergence in the solution of non-linear problems through the comparison of the results obtained on successively refined meshes. Mesh refinements are part of adaptive methods which are designated to achieve both accuracy and efficiency and in which mesh refinements are applied only where needed. The adaptive methods generally provide mesh refinements for efficiency as dictated by predetermined criteria, the criteria are determined by some error indicators which are usually represented by gradients of a suitable variable and the larger the gradient, the finer the mesh required.

There are two types of adaptive methods namely the structured adaptive methods (Dwyer et al., 1982, Gnoffo, 1980, Nakamura 1982, Eiseman 1985, and Brackbill and Saltzmann 1982) developed for finite difference method and the unstructured adaptive methods (Oden *et al.* 1986, and Babuska *et al.* 1986) developed for finite element method formulation. The latter can be formulated using mesh refinement methods (h-methods), mesh movement methods (r-methods), mesh enrichment methods (p-methods), combined mesh refinements and movements (hr-methods), and combined mesh refinements and enrichments (hp-methods). The mesh refinement methods (h-methods) are adopted in the present work. The basic idea of the h-methods consist of keeping unchanged the element selected for the domain discretization while the number and size of the elements vary with each level of mesh refinement. Further information about the structured adaptive methods and the unstructured adaptive methods can be found from the following authors Bathe (1996), Zienkiewick and Taylor (1994), Babuska and Suri (1990), Oden *et al*. (1989, 1995), Chung (2002), Oden (1988), Peraire *et al*. (1987), Probert *et al*. (1991), Ghia *et al*. (1989), and Altas and Stephenson (1991).

## 4.6 Schematic diagram of the developed schemes

The solution algorithms for the two developed schemes described above can be summarised by the flow chart given by figure 4.1.

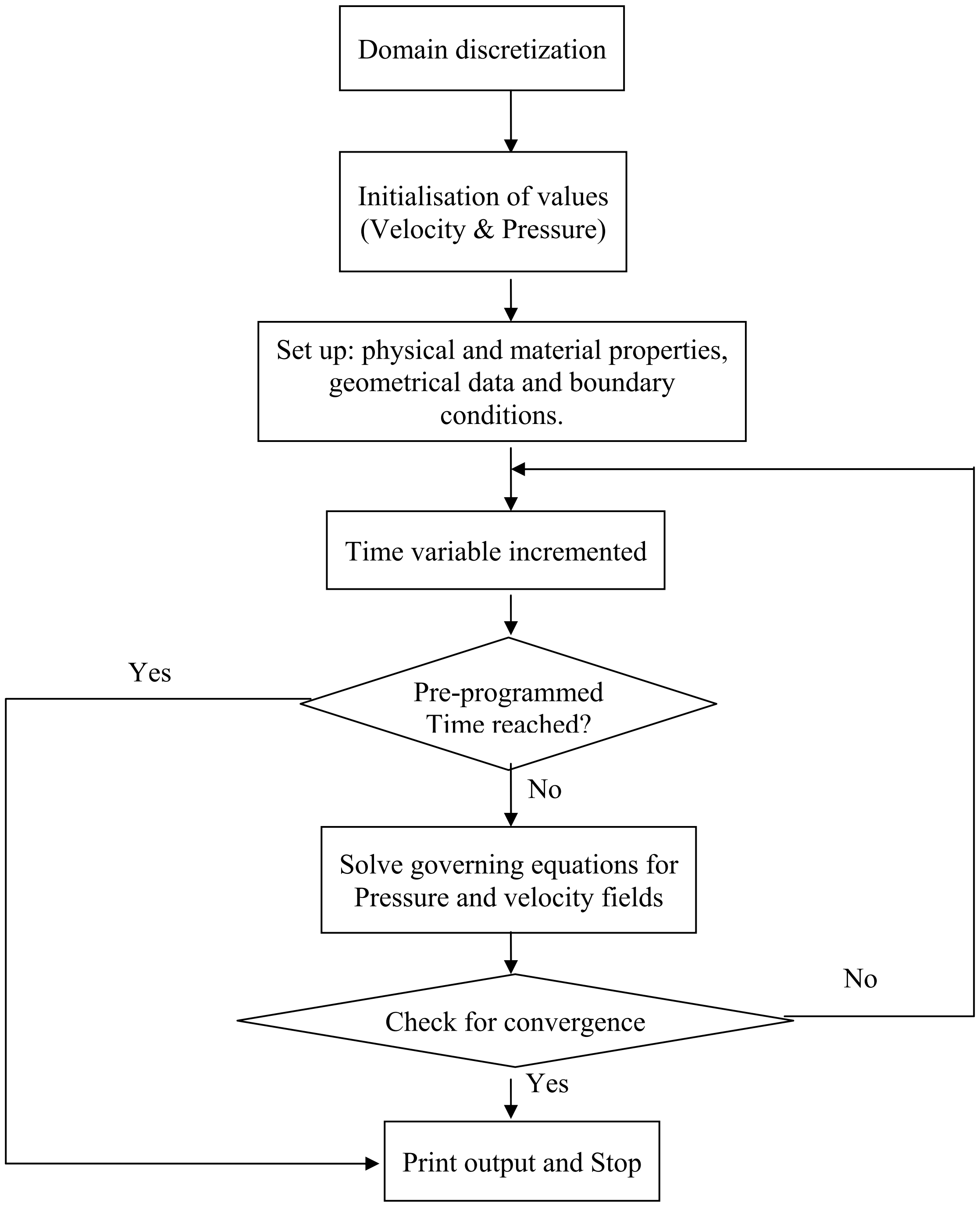

Domain discretization
Initialisation of values (Velocity & Pressure)
Set up: physical and material properties, geometrical data and boundary conditions.
Time variable incremented
Pre-programmed Time reached?
Yes
No
Solve governing equations for Pressure and velocity fields
Check for convergence
No
Yes
Print output and Stop

**Figure 4.1:** Flow chart for both the UVWP and the modified UVWP schemes.

# CHAPTER 5

# RESULTS AND DISCUSSIONS

In this chapter simulations results obtained in the present study are presented and discussed. These results are generated using the following procedure. The two schemes developed in chapter 4 were coded and compiled using FORTRAN via Microsoft Visual Studio 2005, to serve as the solver routines in the present simulation studies. The pre-processing (mesh generation) part of the simulation was done using Cosmos GeoStar software and the post-processing (visualization of results) was carried out using Tecplot software.

The software used for pre-processing, number crunching and post-processing, ( here referred to as Cosmos GeoStar, FORTRAN and Tecplot) are linked via an in-house developed utility programme (FEUT) which reads the output files from the pre-processor (Cosmos GeoStar) converting them into input files in a format readable by the solver (FORTRAN). Once the solution process completed, the solver returns the solutions back to the utility FEUT programme which rewrites the solutions in a format readable by the post-processor (Tecplot) in order to proceed with the analysis of results.

All these processes (geometry definition, mesh generation and visualization) are carried out using an Intel Pentium ® IV 3.00 GHz personal computer.

In all of the simulations presented in this chapter the fluid rheology is based on assuming a generalized Newtonian behaviour. Typical set of physical properties of such a fluid are as given in the following table.

| **Physical properties** | **Values** |
|---|---|
| $\rho$ (density) | $980 \text{ kg m}^{-3}$ |
| n (power law index) | 0.87-1.23 |
| $\eta_0$ (consistency index) | $80 \text{ kg m}^{-1} \text{s}^{-1}$ |
| c (speed of sound) | $1500 \text{ m s}^{-1}$ |

**Table 5.1:** Physical properties of the generalized Newtonian fluid used.

The simulations are carried out using a time level ( $\alpha$ ) of 0.95 and a time increment ( $\Delta t$ ) of 0.001 s.

Although the results obtained using the two developed schemes are mainly similar, the modified UVWP scheme reaches convergence quicker than the traditional UVWP scheme. Therefore, for each simulation only one set of converged results is shown, however, the important difference in the time of taken to obtain such a result using different schemes is noted.

Starting with three benchmark problems in which the simulated fluid is considered to be purely Newtonian (i.e. power law index is set to be 1 in the power law model), the developed codes are used for the numerical analysis of complex problems for generalized Newtonian fluids (for both shear thinning and shear thickening cases). Dimensions of the computational domains used in the complex problems are the same as the benchmark problems but with the difference that in complex problems various types of internal obstructions are introduced within the flow domain.

In the benchmark cases domains consist of rectangular ducts, in which the fluid enters the domain at one end and exits at a specified outlet situated far from the inlet to make the imposition of simple exit boundary conditions acceptable. Three different outlets are of interest in this work, the first one is placed in a position normal to the direction of the fluid flow, the second one is situated at the end of top solid wall while the last outlet is placed at the end of the bottom solid wall. These domains are shown in figures 5.1.1, 5.2.1, and 5.2.3, respectively. The cross section areas of the inlets and outlets for the different geometries in this work are all the same and have a dimension of 0.01 $m^2$ .

As there are no experimental data to validate the results obtained in this project, the validation processes is based on the examination of mass balance across the entire computational domain as well as evaluation of the logical consistency of the computational results.

The drawback of the program is that it cannot be used to simulate fluid flow within a short domain. For such simulation, the imposition of stress free exit conditions may not be realistic, and thus this method may not generate very accurate results.

## 5.1 Benchmark problem 1

### 5.1.1 Computational domain and boundary conditions

In the first benchmark problem, the domain consists of a simple rectangular box of 1m length, 0.1m width and 0.1m high and there is no obstruction to flow as shown in figure 6-1-1. The computational domain is discretized using 8-noded hexahedral isoparametric elements into a mesh of 8550 nodes, and 7252 elements (see figure 5-1-2) and the prescribed boundary conditions correspond to the fluid entering the domain with a velocity of $0.1\,ms^{-1}$ perpendicular to the inlet; the other components of the velocity (v, and w) are zero. The only prescribed boundary condition at the outlet is a zero datum pressure, and the no-slip conditions are applied to the remaining sides of the rectangular box (see figure 5-1-3). Although there is no apparent imposition of exit conditions (as no velocity value is given at this point) stress free conditions at this boundary have been imposed (i.e. the gradient terms appearing after the application of the Green's theorem to the second order derivatives in the equation of motion are set to be zero).

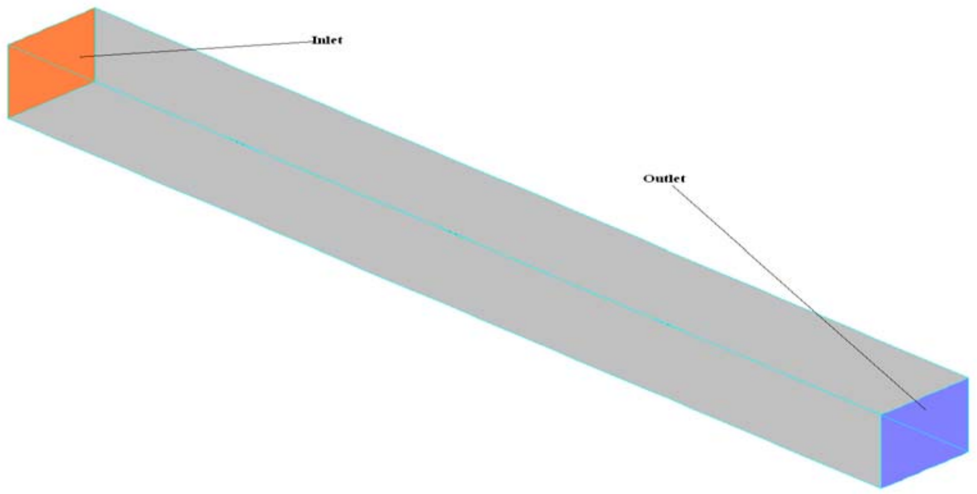


**Figure 5.1.1:** Geometry of benchmark problem 1.

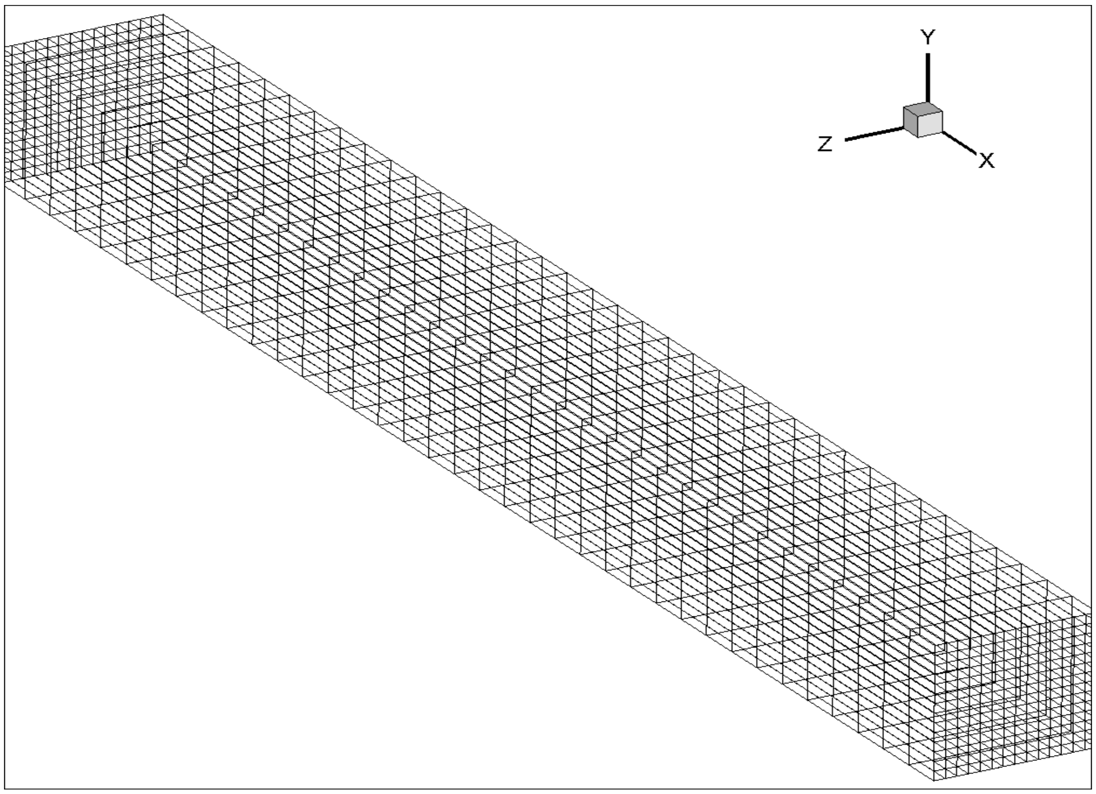


**Figure 5.1.2:** Finite element mesh of benchmark problem 1

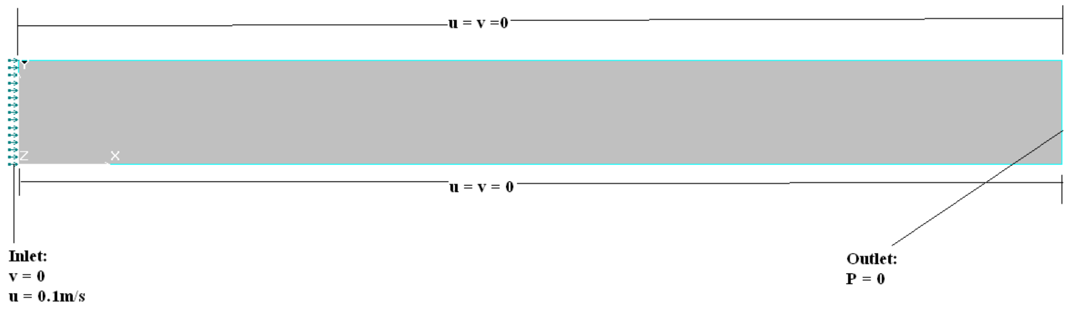


**Figure 5.1.3a:** 2-D schematic representation of the boundary condition in the xy plane (benchmark problem 1).

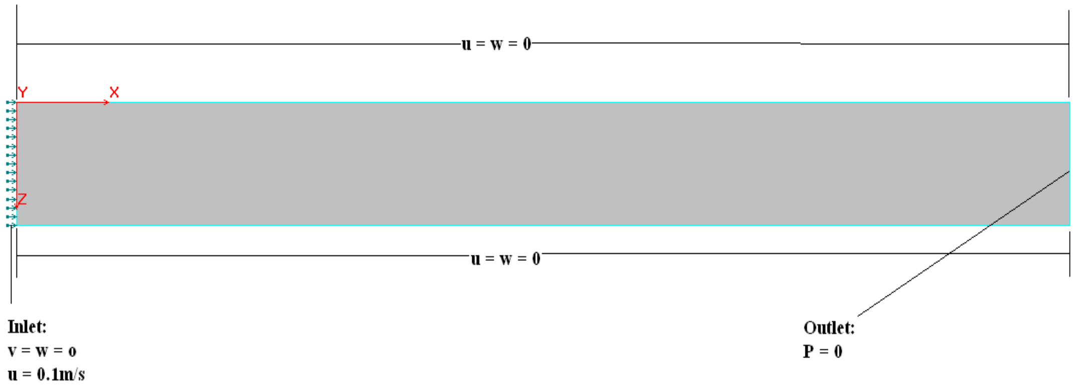


**Figure 5.1.3b:** 2-D schematic representation of the boundary condition in the xz plane (benchmark problem 1).

### 5.1.2 Results

The results obtained after running the simulation for this first benchmark case are given by figure 5.1.4 through 5.1.14, and as it can be seen from figure 5.1.4, pressure decrease in the direction of the flow with the highest pressure (about 10K Pa) found in the vicinity of the inlet and as the fluid moves across the domain there is loss of energy which explain the decrease in the pressure values which reached zero around the outlet as given by figure 5.1.5.

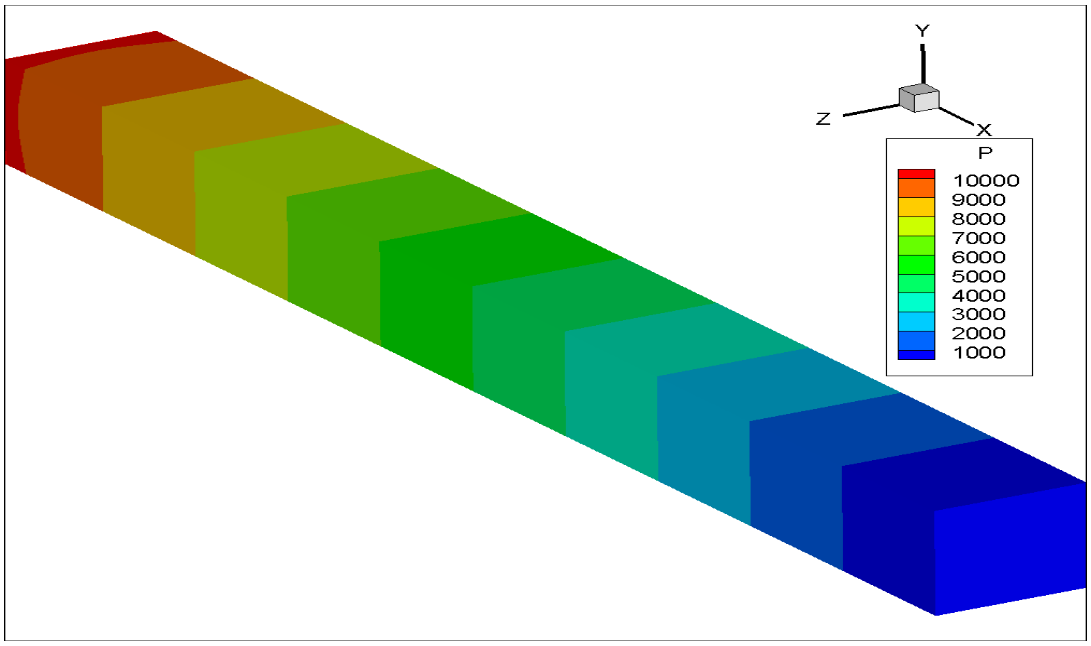


**Figure 5.1.4:** Pressure distribution (benchmark problem 1).

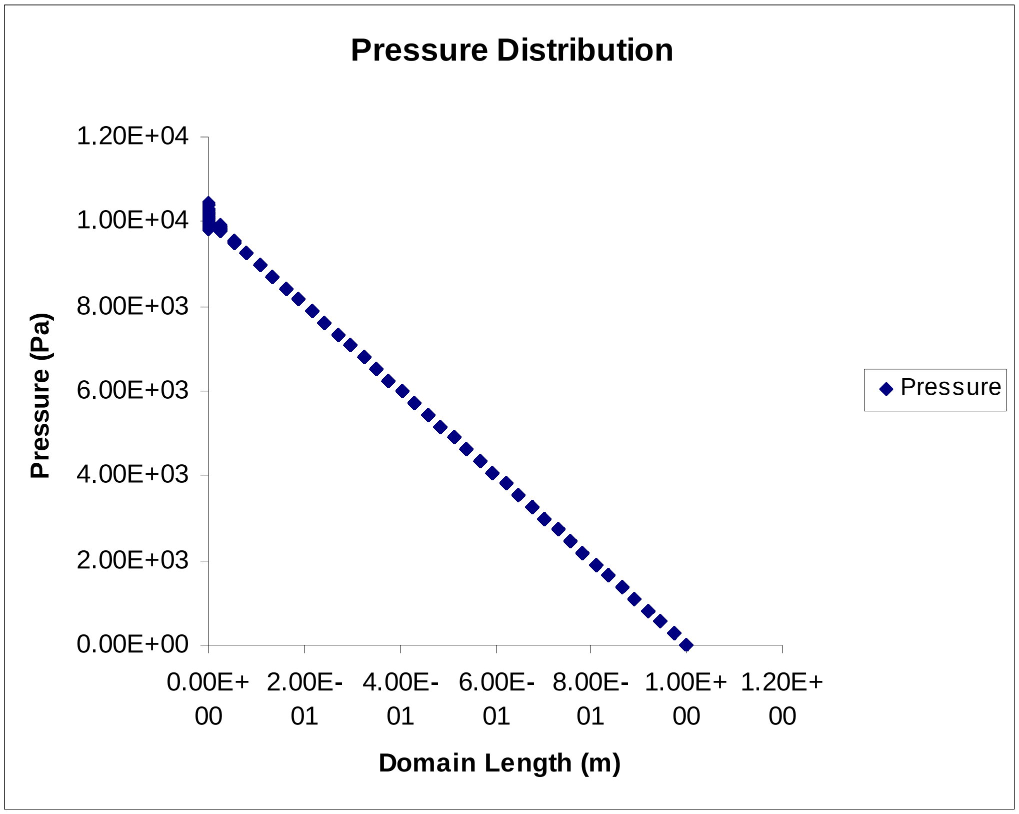


**Figure 5.1.5:** pressure distribution across the domain (benchmark problem 1)

Figure 5.1.6 shows the velocity vector profile in the x-y plane taken at position z = 0.05m. The fluid enters the domain through the inlet with an average velocity of 0.1 $m.s^{-1}$ giving mass inflow rate of $1\,kg.s^{-1}$ and exit the domain with an average velocity of $0.0988\,m.s^{-1}$ giving a mass outflow rate of $0.988\,kg.s^{-1}$. These figures prove that there is conservation of mass since the error between the mass inflow and mass outflow is only 1.2% which falls in the acceptable range. The maximum velocity of about $0.12\,m.s^{-1}$ is found to be located between the range starting from x = 0.1m to x = 1m, y = 0.02m to y = 0.07m, and z = 0.02m to z = 0.07m across the domain length high, width respectively.

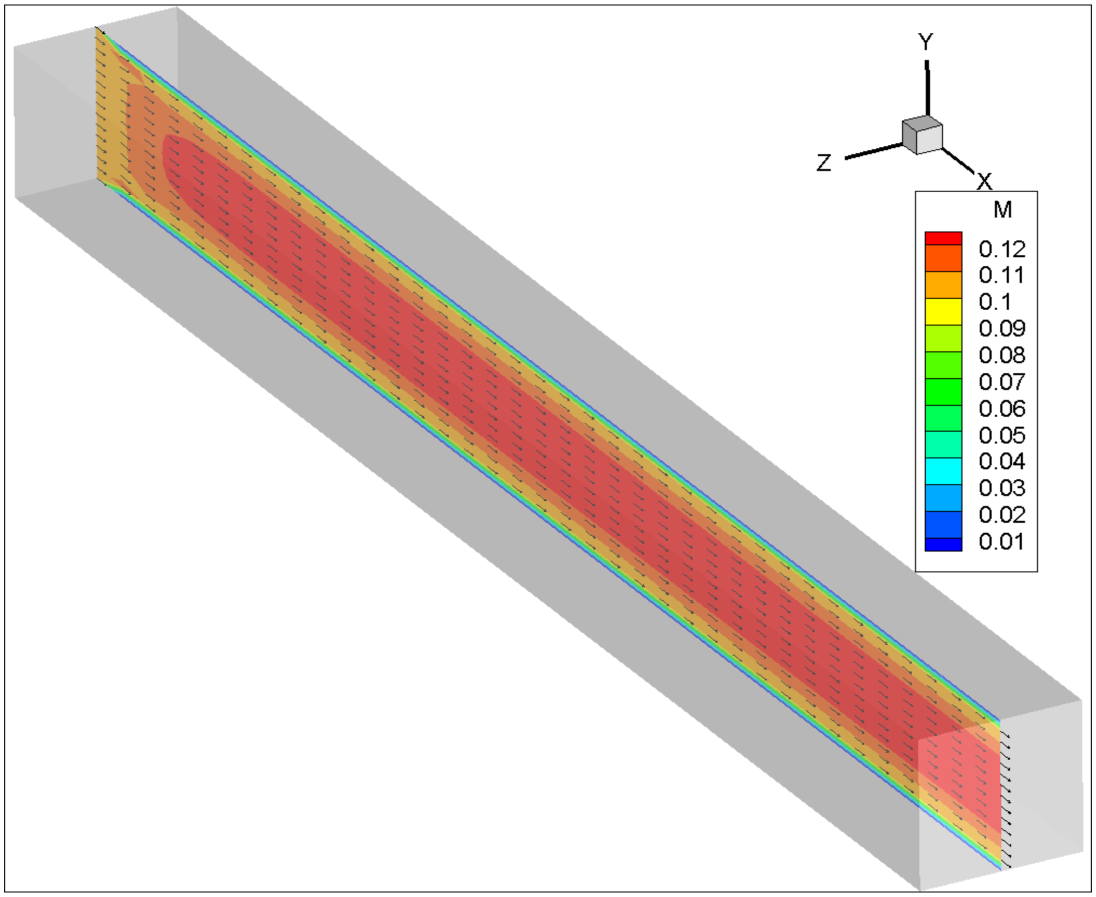


**Figure 5.1.6:** Vector plot profile coloured by the velocity magnitude contour (benchmark problem 1).

As there are always viscous momentum boundary layers at solid surfaces, different section contour plots of the velocity in the vicinity of the solid surfaces are presented for each of the simulations done in this project. For the first test case these contours plot are represented by figures 5.1.7 through 5.1.10. Figure 5.1.8 illustrates the velocity section contour plotted at position y equal 0.002m, that is just 0.002m above the bottom solid wall, while figure 5.1.9 illustrates the velocity section contour plotted at position y equal 0.098min that is just 0.002m below the top solid wall, and figure 5.1.10 represent a plot combined the section plot from figure 5.1.9, figure 5.1.10 plus an additional section plotted at position y equal 0.05m which represents half of the domain in the y direction and the region where the fluid moves with maximum velocity. It can be noted that in the vicinity of the solid walls, the fluid flow experiences a velocity change and this change is due to the presence of boundary layers. And as proved by many researchers, the velocity profiles are less developed in planes closer to solid walls because of the boundary layers effect.

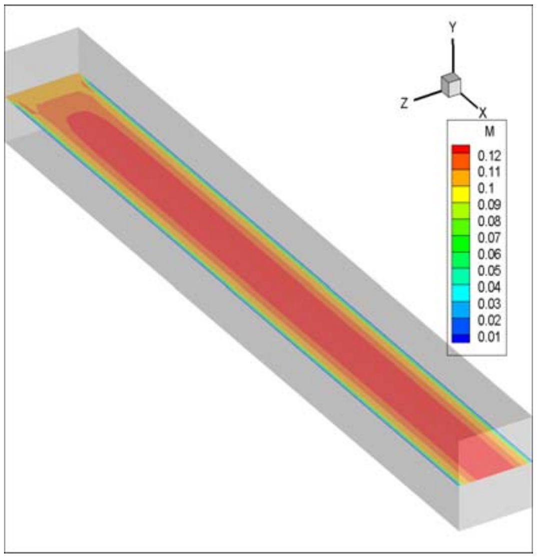


**Figure 5.1.7:** Profile of the contour of the velocity magnitude (y = 0.05m plane) benchmark problem 1.

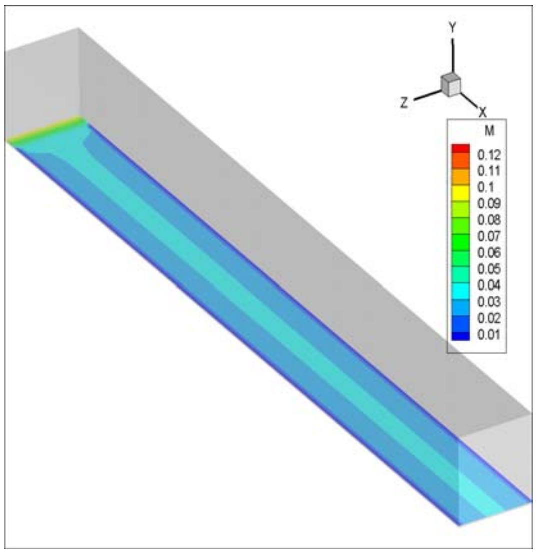


**Figure 5.1.8:** Profile of the contour of the velocity magnitude(y = 0.02m plane). benchmark problem 1.

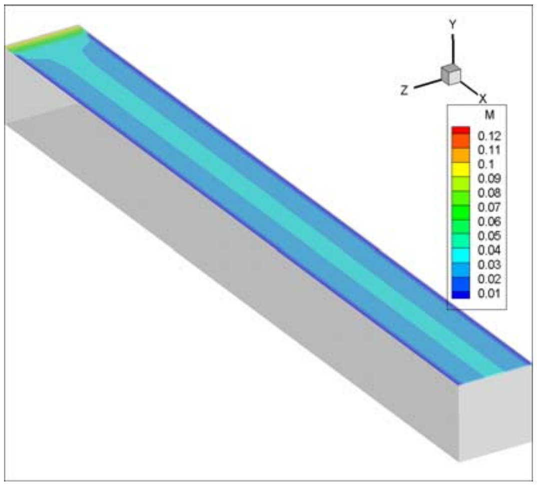


**Figure 5.1.9:** Profile of the contour of the velocity magnitude(y = 0.098m plane) benchmark problem 1.

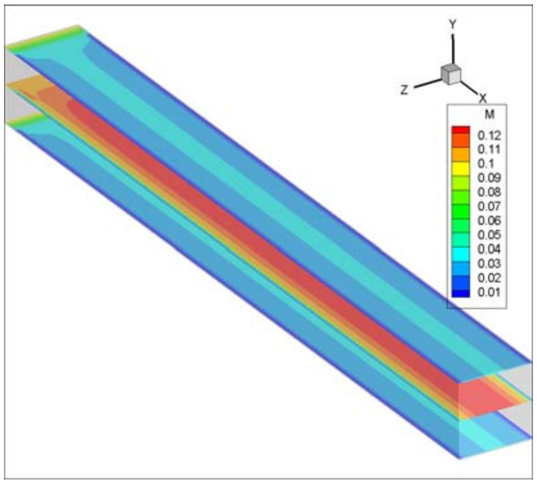


**Figure 5.1.10:** Combined profiles of the the velocity magnitude (y = 0.02m, y = 0.05m, and y = 0.098m planes). benchmark problem 1.

The results presented in figures 5.1.11 through 5.1.14 represent section plot of the velocity contours in the x-y plane. The contours are plotted at location z equal 0.5m (middle of the domain), z equal 0.02m (close to the right solid wall), and at z equal 0.098m (close to the left solid wall). One can note that even with these plots in the x-y plane the results obtained are similar to obtained those obtained previously in x-z plan (figure 5.1.7 through 5.1.10).

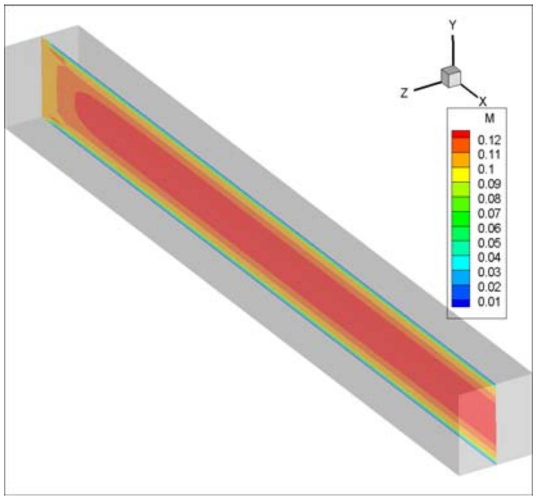


**Figure 5.1.11:** Profile of the contour of the velocity magnitude (z = 0.05m plane) benchmark problem 1.

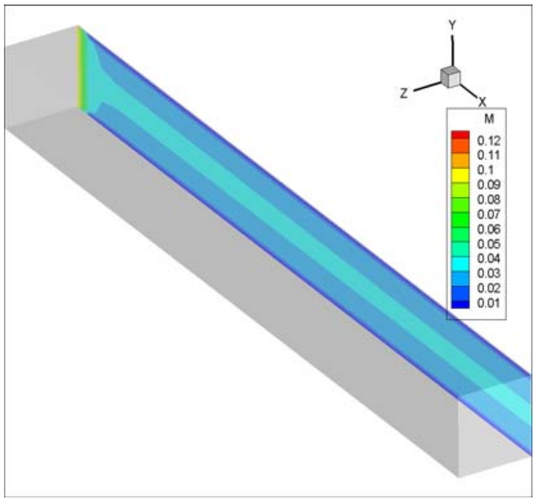


**Figure 5.1.12:** Profile of the contour of the velocity magnitude(z = 0.02m plane) benchmark problem 1.

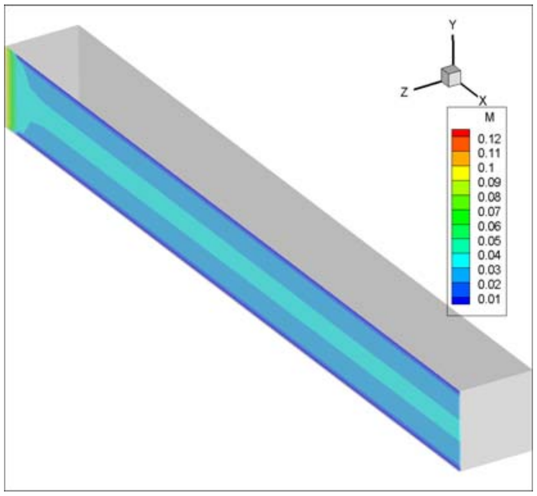


**Figure 5.1.13:** Profile of the contour of the velocity magnitude (z = 0.098m plane) benchmark problem 1.

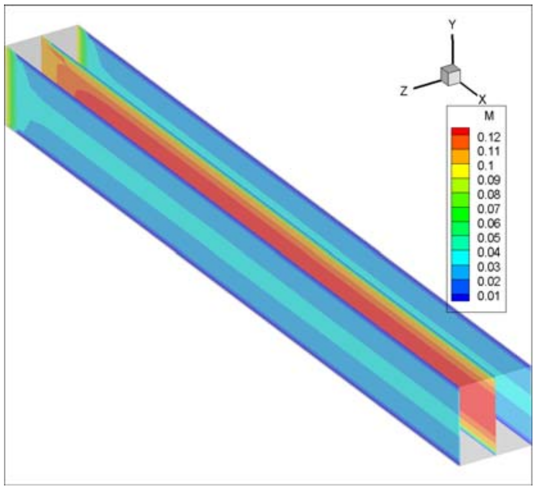


**Figure 5.1.14:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) benchmark problem 1.

## 5.2 Benchmark problem 2

### 5.2.1 Computational domain and boundary conditions

The geometry dimension, finite element mesh sizes are the same as for the one for bench mark problem 1, the outlet for this case is positioned as given in figure 5.2.1, the finite element mesh size, and the boundary conditions are given by figure 5.2.2 and 5.2.3, respectively.

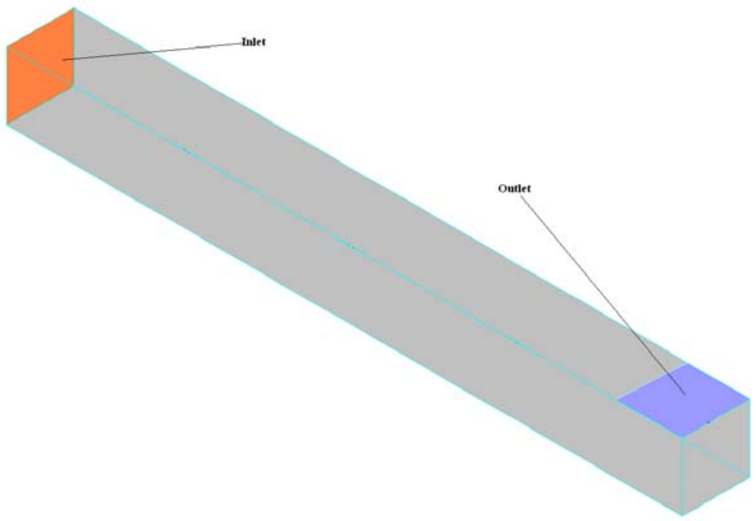


**Figure 5.2.1:** Geometry of benchmark problem 2**.**

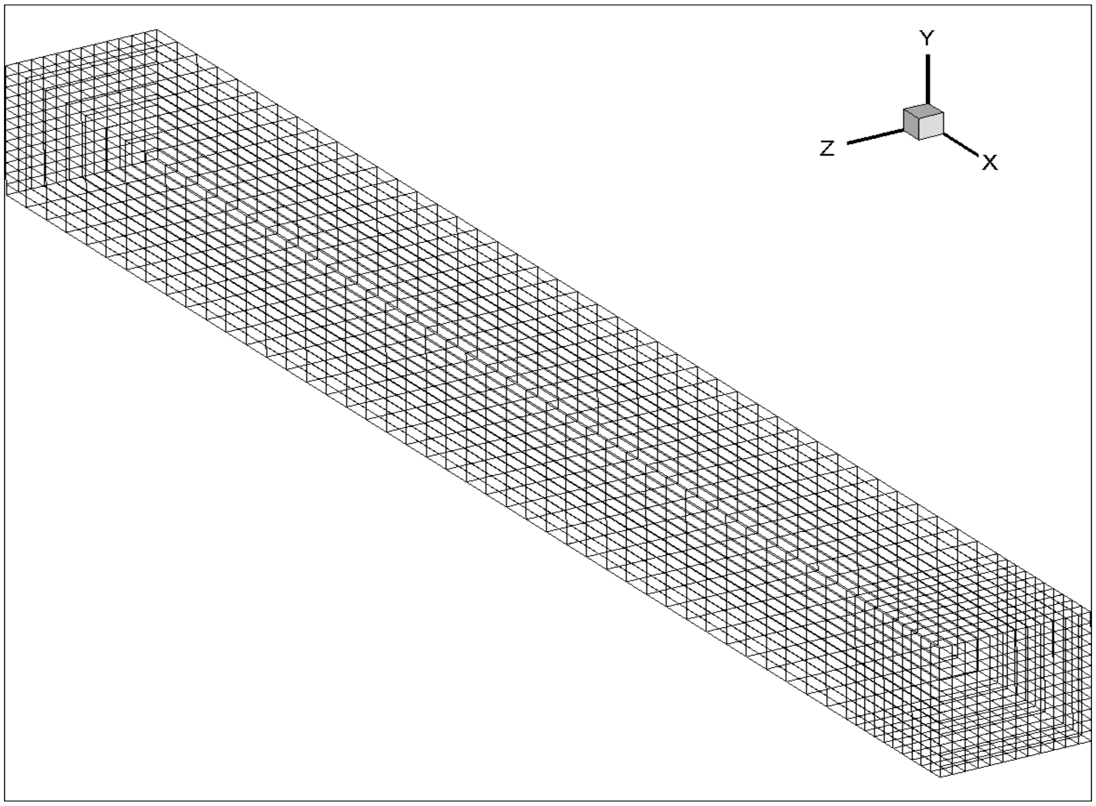


**Figure 5.2.2:** Finite element mesh for benchmark problem 2.

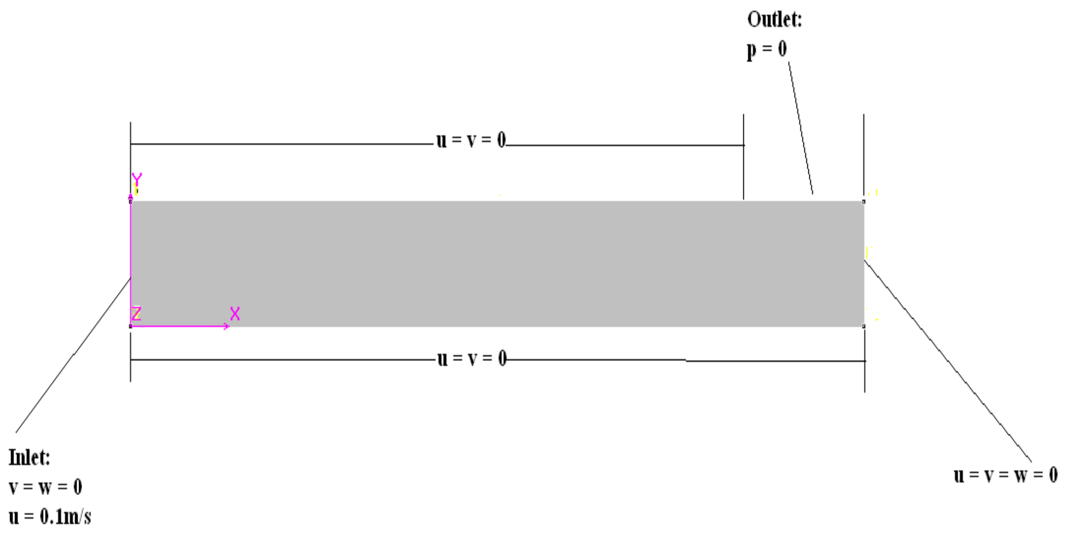


**Figure 5.2.3a:** 2-D schematic representation of the boundary condition in the xy plane (benchmark problem 2).

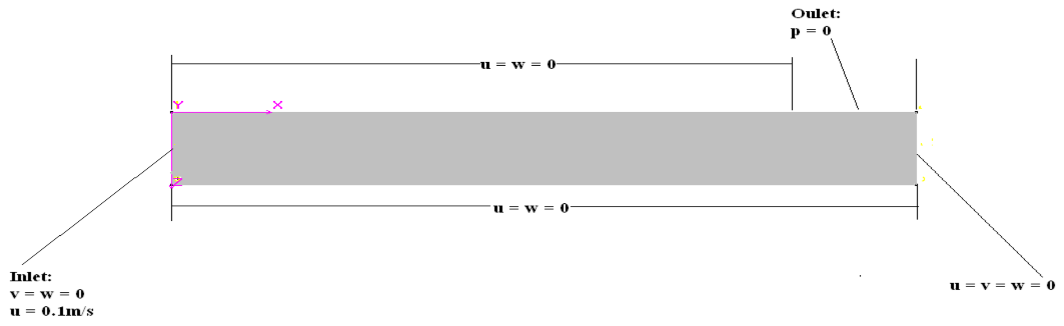


**Figure 5.2.3b:** 2-D schematic representation of the boundary condition in the xz plane (benchmark problem 2).

## 5.2.2 Results

The results for the benchmark problem 2 are given in figures 5.2.4 through 5.2.14. One can note that when the outlet is placed at the top end of the top solid wall, there is an increase in both pressure and velocity values with the pressure values vary between zero (the initial value) to a maximum of 20KPa (figures 5.2.4 – 5.2.5), and the velocity magnitudes vary from $0.01\,ms^{-1}$ to a maximum of $0.14\,ms^{-1}$ (figure 5.2.6) compared to the previous benchmark problem where the outlet was situated at a position normal to the fluid flow. Once again, it can be noted from figure 5.2.4 that pressure decreases in the direction of the flow. But although the fluid moves slightly fast in this second benchmark case, the mass flow from the outlet is $0.983\,kg.s^{-1}$ which is lower than $0.988\,kg.s^{-1}$ obtained for benchmark problem 1. The difference between the mass flow in and the mass flow out is about 1.7% which represent 0.7% increase compared to the 1.2% error obtained for the first benchmark case, the lower value of the mass flow from the outlet in this benchmark case can be explained by the fact that the region where the fluid moves faster is found to be far from the outlet (figure 5.2.6), but in the vicinity of the outlet the velocity of the fluid is low. The pressure distribution and graph given by figures 5.2.4 and 5.2.5 show that the highest values of the pressure (18KPa-20KPa) are distributed over a length of 0.211m which is longer than the length of 0.105m on which the highest values of pressure (9KPa to 10KPa) were distributed for benchmark problem 1.

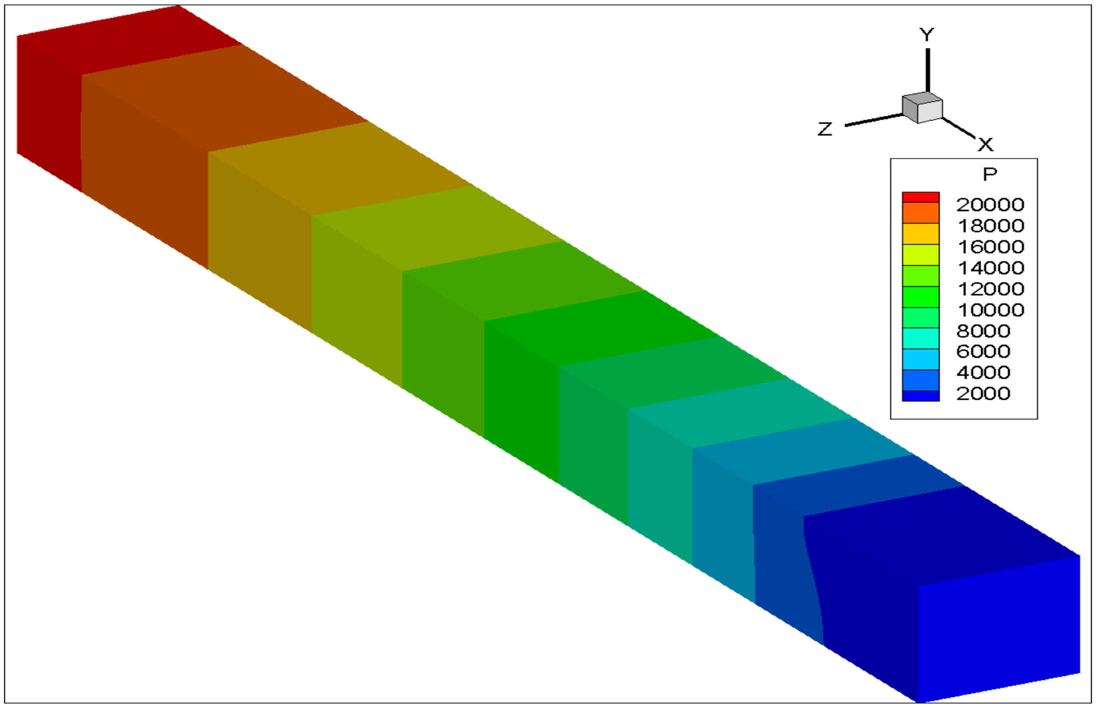


**Figure 5.2.4:** Pressure distribution (benchmark problem 2).

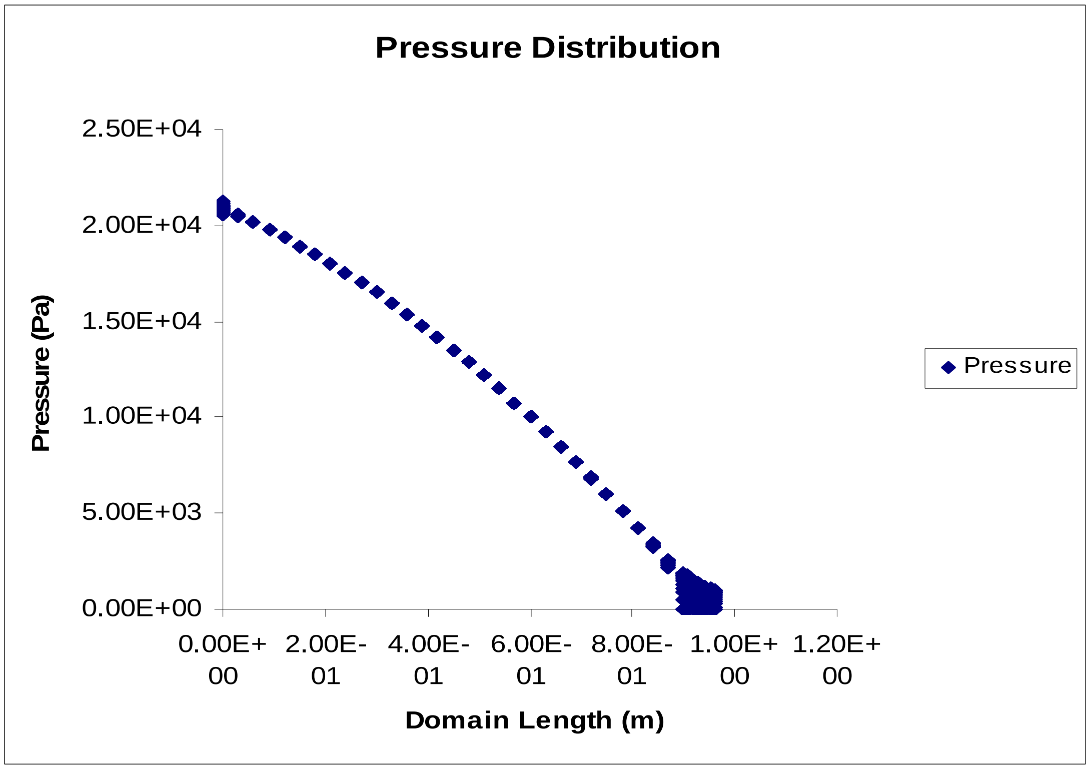


**Figure 5.2.5:** pressure distribution across the domain (benchmark problem 2).

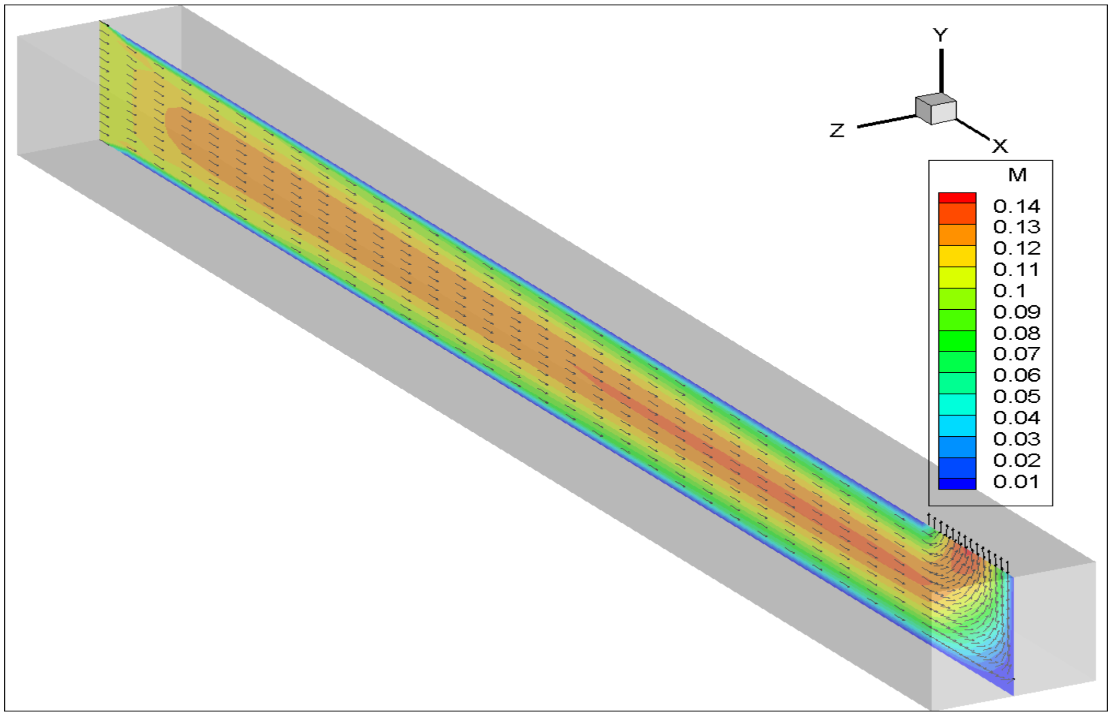


**Figure 5.2.6:** Vector plot profile coloured by the velocity magnitude contour. (benchmark problem 2).

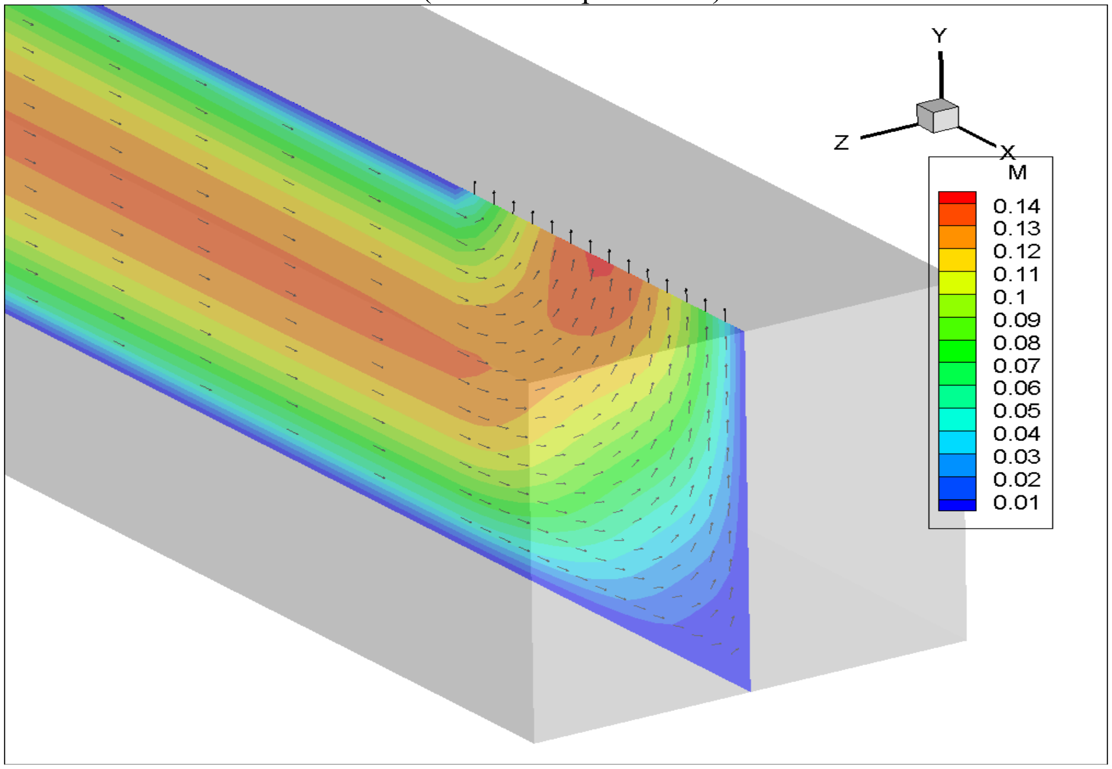


**Figure 5.2.7:** Vector plot profile coloured by the velocity magnitude contour zoomed around the outlet (benchmark problem 2).

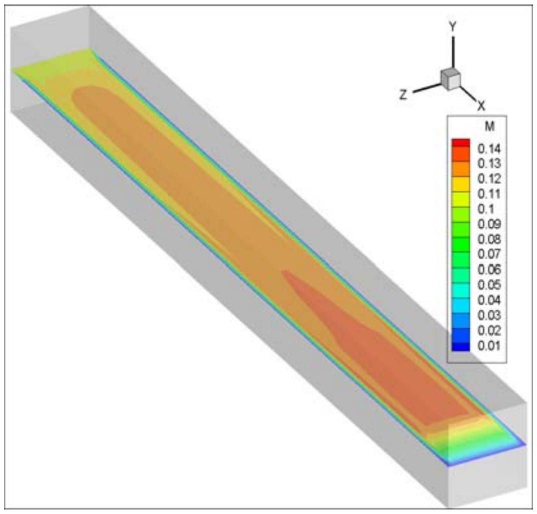


**Figure 5.2.8:** Profile of the contour of the velocity magnitude (y = 0.05m plane) (benchmark problem 2).

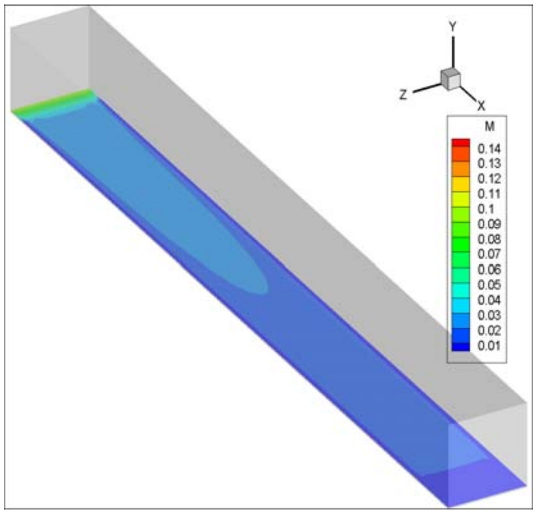


**Figure 5.2.9:** Profile of the contour of the velocity magnitude(y = 0.02m plane) (benchmark problem 2).

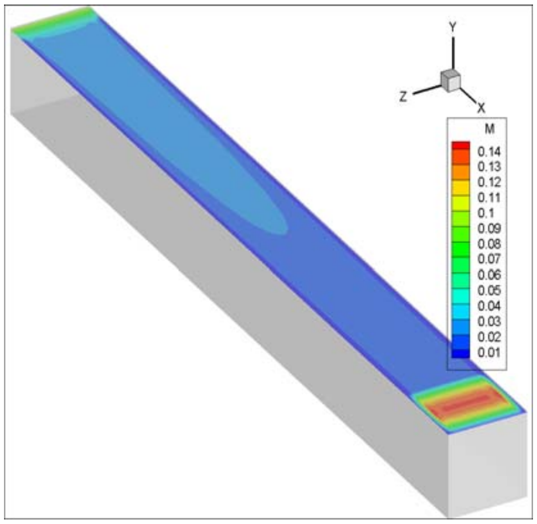


**Figure 5.2.10:** Profile of the contour of the velocity magnitude(y = 0.098m plane) (benchmark problem 2).

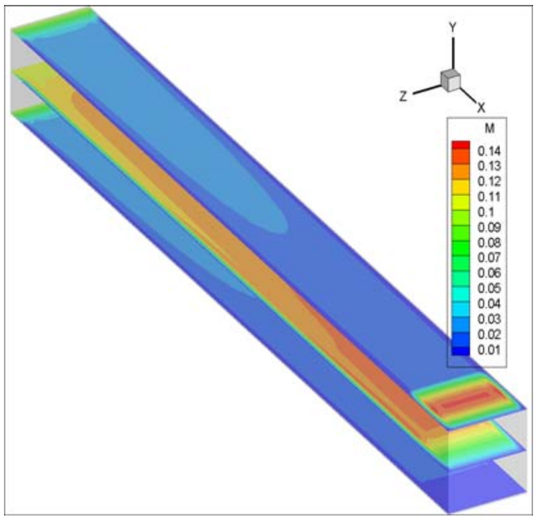


**Figure 5.2.11:** Combined profiles of the the velocity magnitude (y = 0.02m, y = 0.05m, and y = 0.098m planes) (benchmark problem 2).

Figures 5.2.7 through 5.2.8 above represent the contours plots of the velocity plotted in different positions (x-z plane) in the vicinity of the bottom solid wall (figure 5.2.8), and of the top solid wall (figure 5.2.9). Note that in the second benchmark case, the boundary layers have a parabolic shape with a higher width but cover over a distance of about half of the entire domain whereas in the previous test case, the boundary layers are thinner but cover the entire length of the domain (figure 5.1.8 and 5.1.9).

Figures 5.2.12 and 5.2.13 represent the contours plots of the velocity in different positions (x-y plane) in the vicinity of the right solid wall (figure 5.2.8), and of the left solid wall (figure 5.2.9). It can be seen that the shape and size of these boundary layers are now different compared to the ones obtained in the x-z plane, the length of the boundary layers are now of about 0.70m but their starting points are located about 0.3m ahead of the inlet. Another difference to note is that for the benchmark case 1, the shape and size of the boundary layers were the same regardless of whether the contour plots were taken in the x-y or x-z plane.

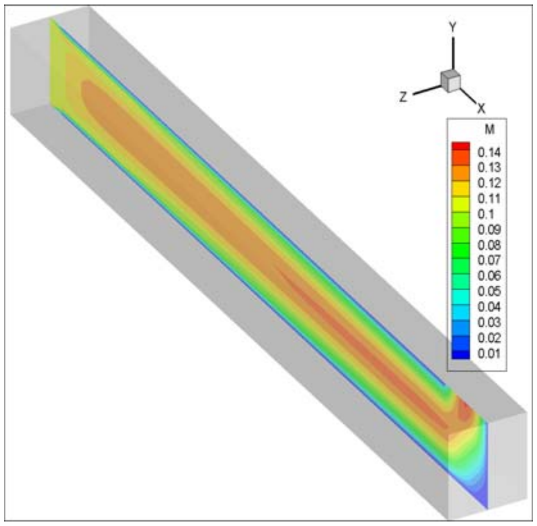


**Figure 5.2.12:** Profile of the contour of the velocity magnitude (z = 0.5m plane) (benchmark problem 2).

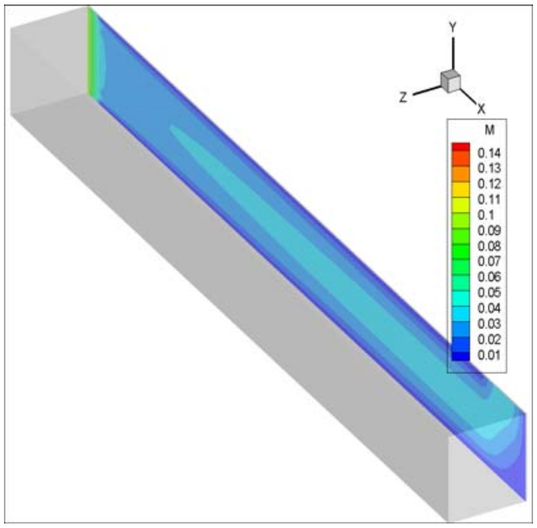
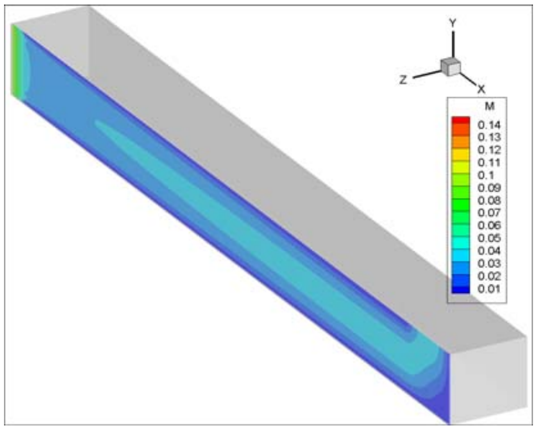


**Figure 5.2.13:** Profile of the contour of the velocity magnitude (z = 0.02m plane) (benchmark problem 2).



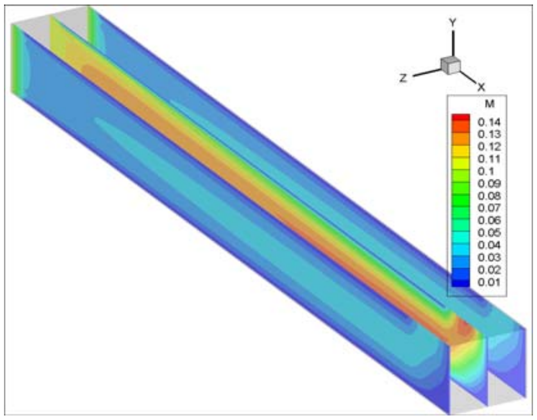


**Figure 5.2.14:** Profile of the contour of the velocity magnitude (z = 0.098m plane) (benchmark problem 2).
.

**Figure 5.2.15:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) (benchmark problem 2).

## 5.3 Benchmark problem 3

### 5.3.1 Computational domain and boundary conditions

The geometry dimension, finite element mesh sizes are the same as for the two previous cases, the outlet for this case is positioned as given in figure 5.3.1, the finite element mesh size, and the boundary conditions are given by figure 5.3.2 and 5.3.3 respectively.

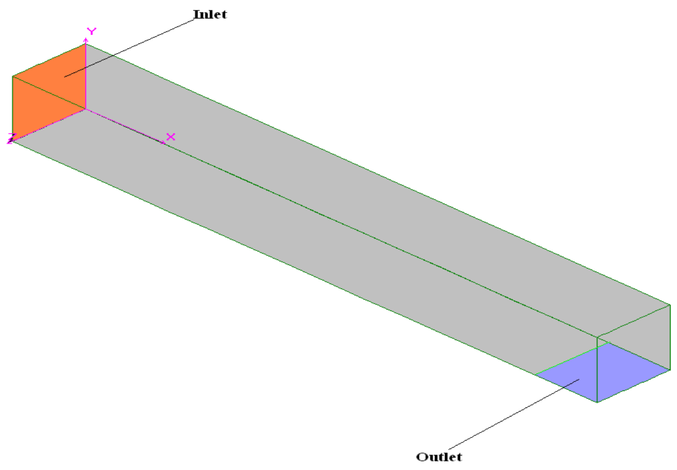


**Figure 5.3.1:** Geometry of benchmark problem 3**.**

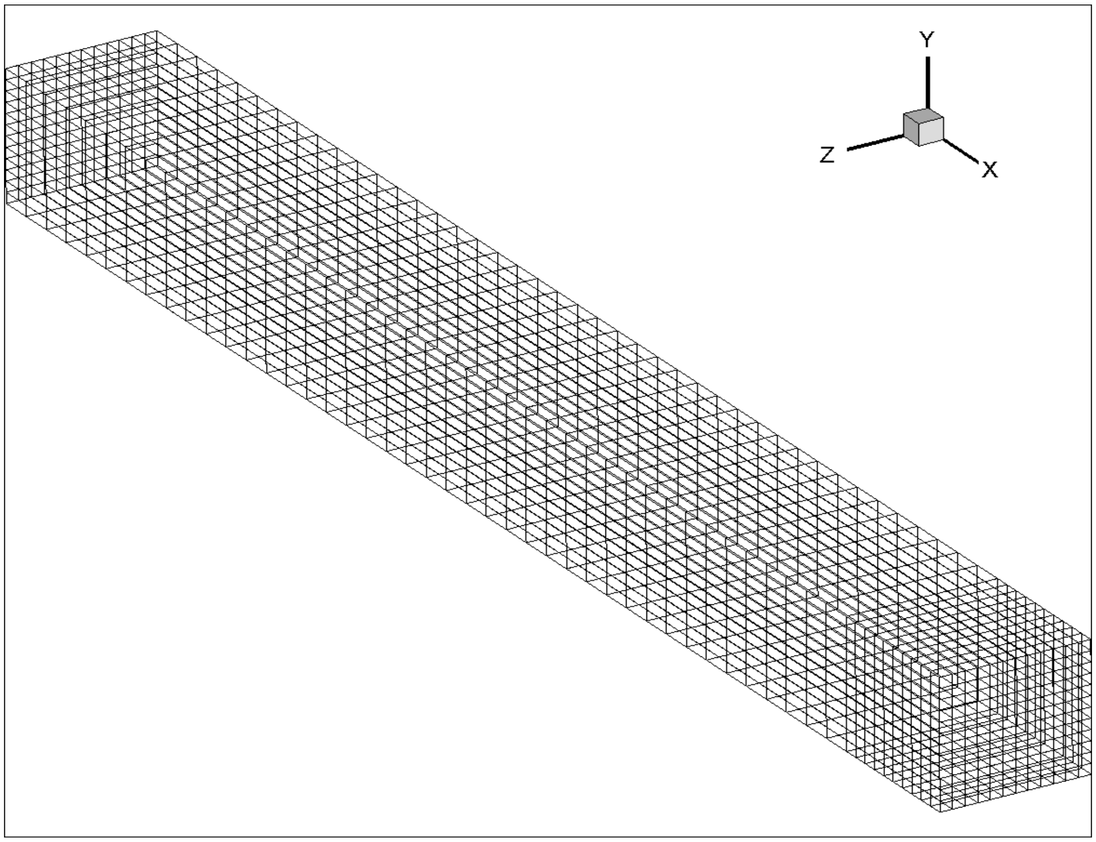


**Figure 5.3.2:** Finite element mesh for benchmark problem 3

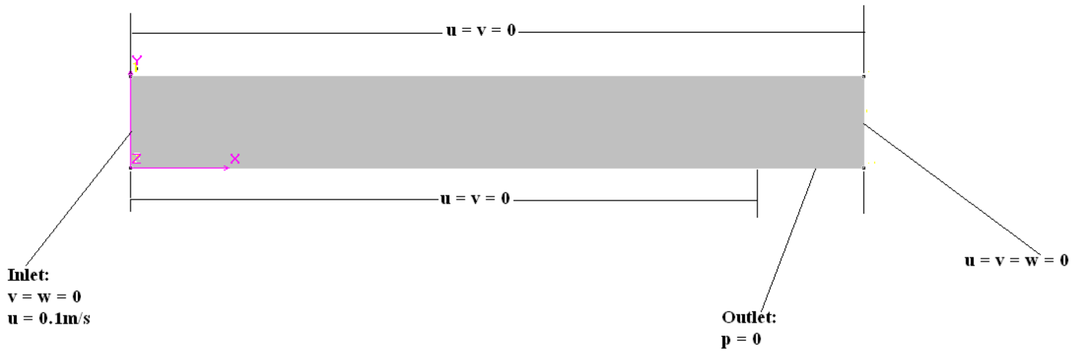


**Figure 5.3.3a:** 2-D schematic representation of the boundary condition in the xy plane (benchmark problem 3).

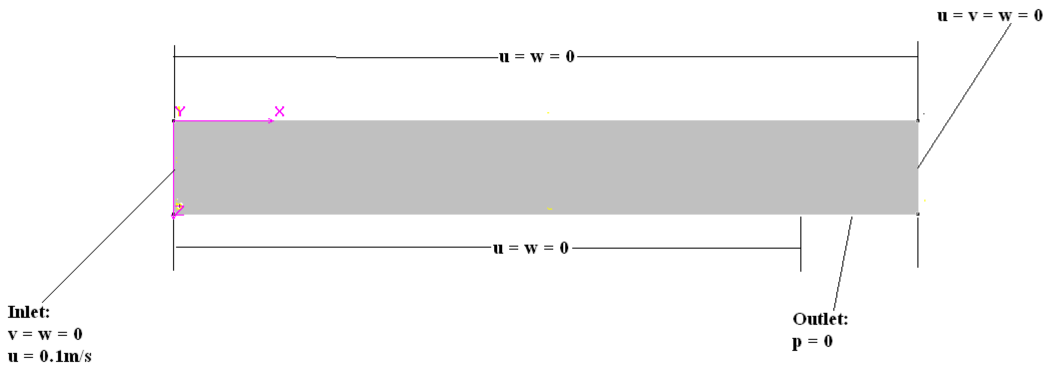


**Figure 5.3.3b:** 2-D schematic representation of the boundary condition in the xz plane (benchmark problem 3).

## 5.3.2 Results

The results obtained when the outlet is located at the bottom end of the solid wall show an increase in pressure values compared to the two previous cases. The values of the pressure for in this case vary from 0 initially set at the outlet to 26K Pa (figure 5.3.4 and 5.3.5) compared to the maximal value of 10K Pa for benchmark problem 1 (outlet located at the end of the geometry) and 20K Pa for benchmark problem 2 (outlet located at the end of the top solid wall). But the maximum magnitude of the velocity remains the same as it was for benchmark case 2; the values vary from 0.01 $m s^{-1}$ to 0.14 $m s^{-1}$ whereas the same figure was from 0.01 $m s^{-1}$ to 0.12 $m s^{-1}$ for benchmark case 1 and from 0.01 $m s^{-1}$ to 0.14 $m s^{-1}$ for benchmark case 2. The pressure decrease in the direction of the flow (figure 5.3.4) just as obtained for the two previous benchmark problems. But although the magnitude of the velocity is high for this third benchmark case, the computed mass flow from the outlet 0.980 $kg\, s^{-1}$ is lower than the ones obtained for benchmark problem 1 and 2 and this low value of outlet mass flow yield a 2% error in difference between the mass in and the mass out.

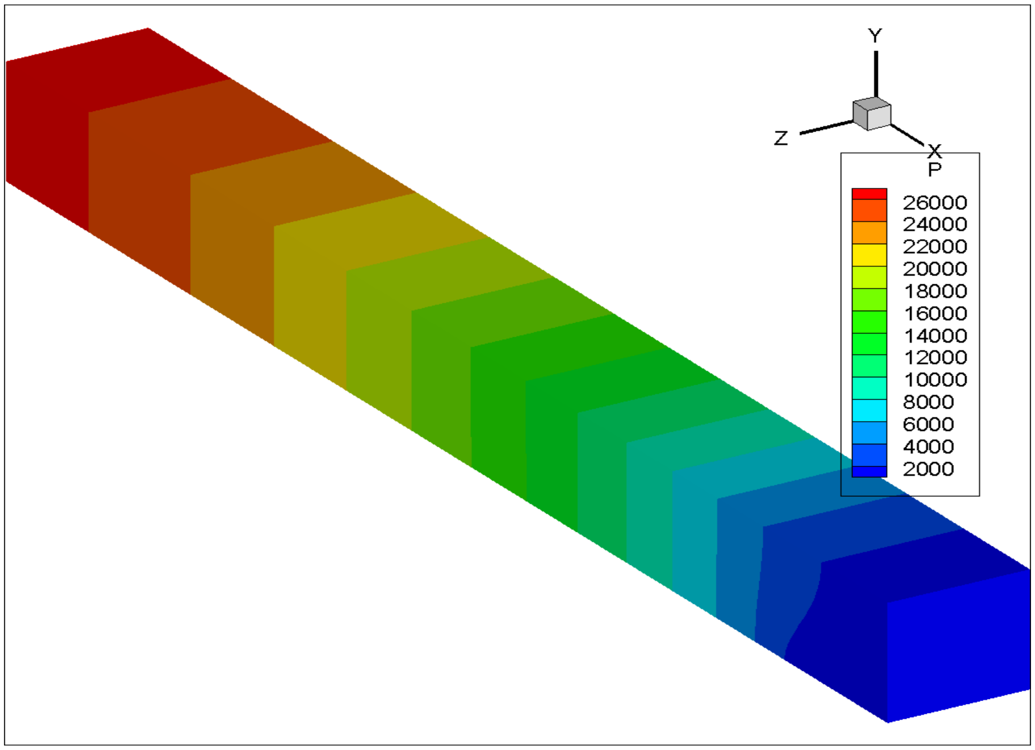


**Figure 5.3.4:** Pressure distribution (benchmark problem 3).

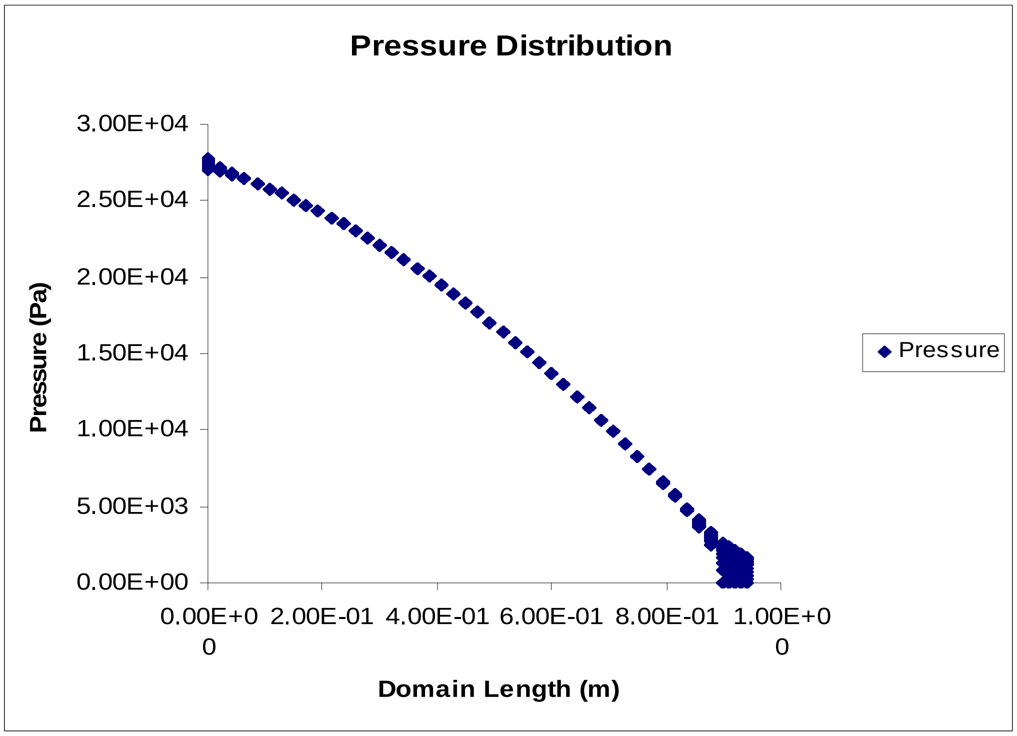


**Figure 5.3.5:** Graph of pressure distribution across the domain (benchmark problem 3).

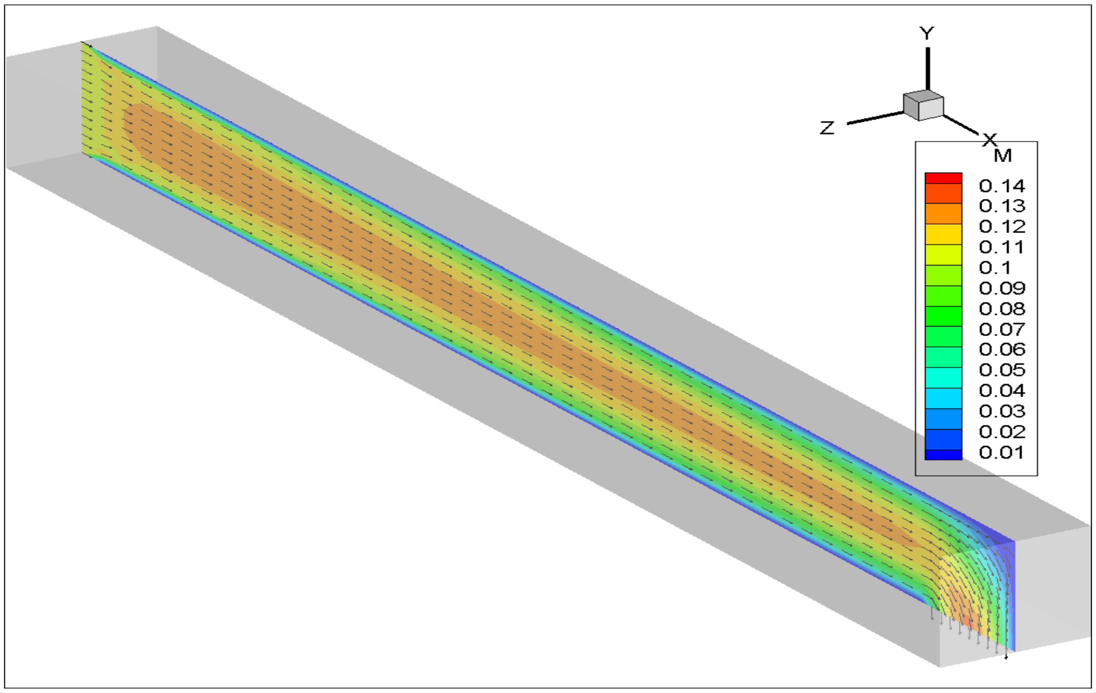


**Figure 5.3.6:** Vector plot profile coloured by the velocity magnitude contour (benchmark problem 3).

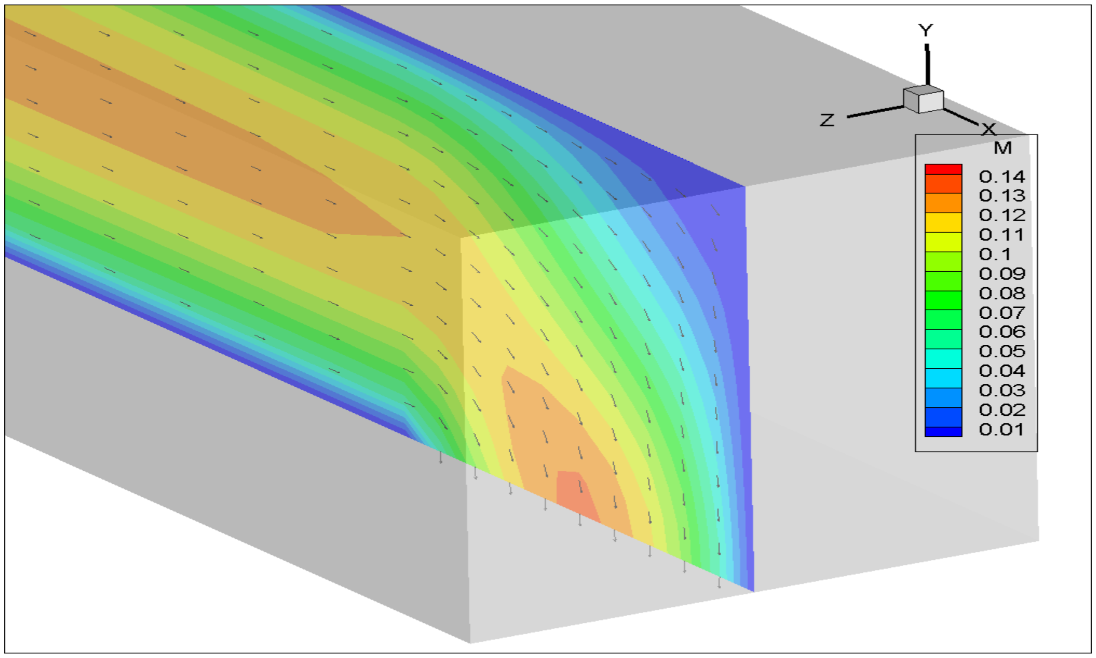


**Figure 5.3.7:** Vector plot profile coloured by the velocity magnitude contour zoomed around the outlet (benchmark problem 3).

Figures 5.3.7 through 5.3.10 represent the contour plots of the velocity in the x-z plan and are similar to the ones obtained for benchmark problem 2 (figures 5.2.7 through 5.2.10), which means when the outlet is located at the end of the bottom solid wall, the fluid flow experienced the same effect in the vicinity of the solid walls than when the outlet is placed at the end of the top solid wall. But when the contour plots are taken in the x-z plan (figures 5.3.11 through 5.3.14), the shape and size of the boundary layers are different from those obtained in benchmark problem 1 (figures 5.1.10 through 5.1.14) and benchmark problem 2 (figures 5.2.10 through 5.3.14). One can note that the length of these boundary layers is short and that their thicknesses are smaller compared to those obtained in the two previous benchmark cases.

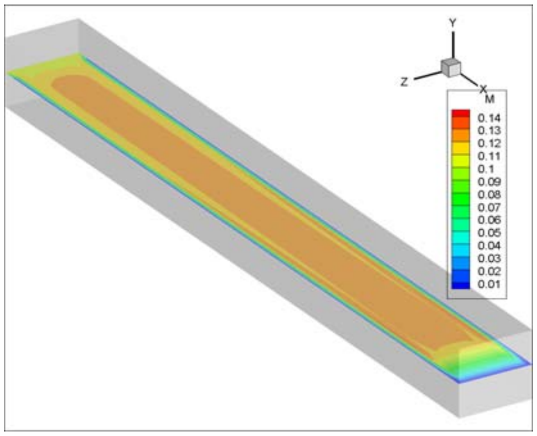


**Figure 5.3.7:** Profile of the contour of the velocity magnitude (y = 0.5m plane) benchmark problem 3.

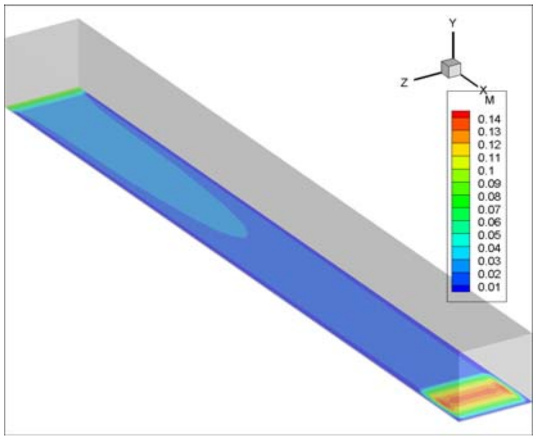


**Figure 5.3.8:** Profile of the contour of the velocity magnitude(y = 0.02m plane) benchmark problem 3.

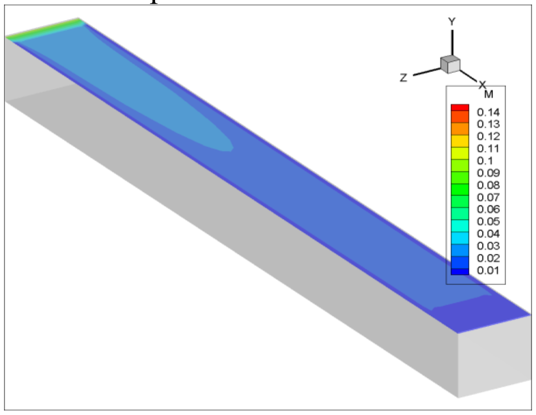


**Figure 5.3.9:** Profile of the contour of the velocity magnitude(y = 0.098m plane) benchmark problem 3.

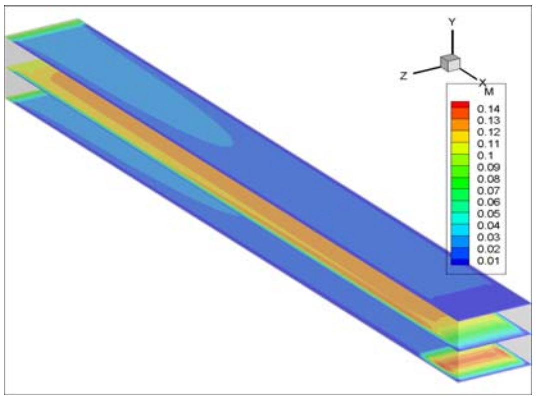


**Figure 5.3.10:** Combined profiles of the the velocity magnitude (y = 0.02m, y = 0.05m, and y = 0.098m planes). benchmark problem 3.

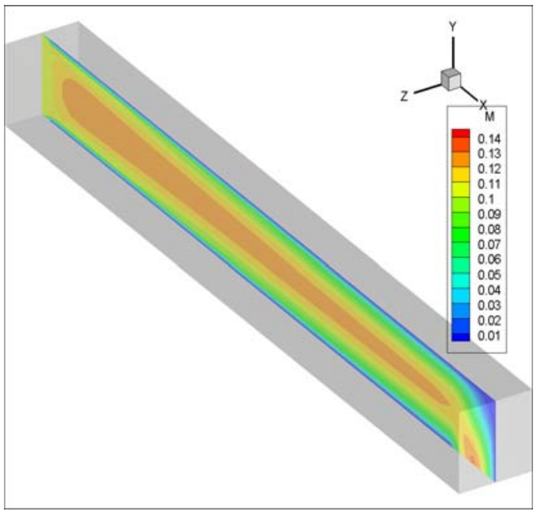


**Figure 5.3.11:** Profile of the contour of the velocity magnitude (z = 0.5m plane) benchmark problem 3.

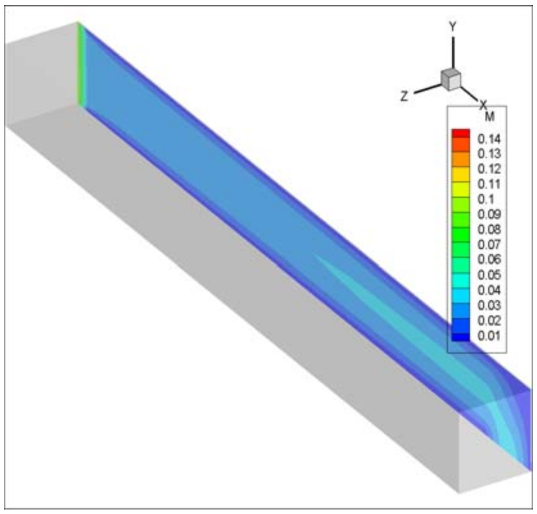


**Figure 5.3.12:** Profile of the contour of the velocity magnitude (z = 0.02m plane) benchmark problem 3.

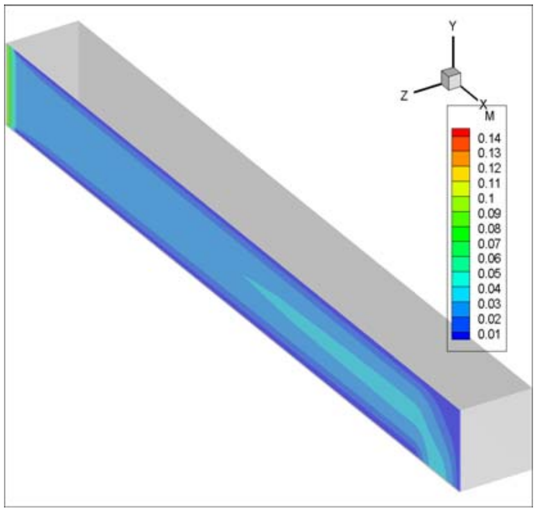


**Figure 5.3.13:** Profile of the contour of the velocity magnitude (z = 0.098m plane) benchmark problem 3.

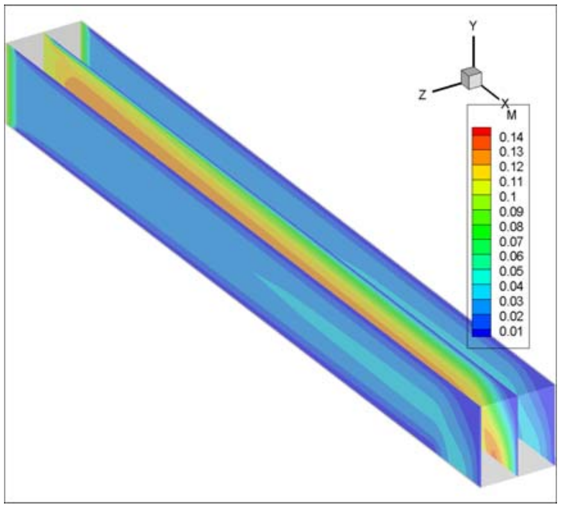


**Figure 5.3.14:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) benchmark problem 3.

The results obtained for the three benchmark problems were stable and accurate these showed that the conservation of mass and momentum across the entire domain were not violated. The worst case of difference between the mass in and the mass out was found to be 2% (benchmark problem 3) which still in the acceptable range. The velocity profiles and pressure contours were all as expected for the three cases. Therefore the numerical schemes developed in chapter four can be used with confidence to proceed with the simulations of the complex test cases planned for this thesis. For the simulations that will follow, an obstruction or obstructions to the fluid flow will be placed somewhere inside the geometry but far away from the outlet so that the flow could reached a fully developed state before exiting the domain. The reason to introduce an obstruction or obstructions is to investigate how it or they will affect the velocity profiles and the pressure distributions across the domain. The obstruction will consist of a square block, or a cylindrical block, or a combination of them. The computational domain in each case will have the same size than the ones used for the benchmark problems 1, 2, and 3 and the emplacement of the outlet will chosen to be one of the three used previously for each case.

The geometries selected for the simulations are used to generate maximum possible contrast between simple flow fields and more complex flow fields including obstructions. The various cases simulated in this work are all typical situations often encountered during process such as polymer moulding, coating, extrusion, and mixing among others. Although the seven test cases are simulated using rectangular geometrical domains, the program can be used for the simulation of fluid flow within others domains of different geometrical shapes. And as mentioned earlier in this chapter, only results obtained using the modified UVWP scheme are presented. Both schemes presents same results when a converged solution is reached, the main difference between the two schemes is that the modified scheme converges faster than the standard UVWP scheme. For instance, with a value of $\Delta t = 0.001s$ used for the test cases simulations, the modified scheme reached convergence just after 3 iterations

While the standard scheme needed 5 iterations for the solutions to converge. It took about 45 minutes to the modified scheme to run one iteration of a problem with 7344 elements and 8882 nodes, whereas with the same data, the standard scheme took about 65 minutes to run a simulation.

The only drawback of the program is that it cannot be used to simulate fluid flow within a short domain. For such simulation, the imposition of stress free exit conditions may not be realistic, and thus this method may not generate very accurate results.

## 5.4 Test case 1: Flow in a duct past a big square obstacle (0.05 × 0.05 × 0.05m)

### 5.4.1 Computational domain and boundary conditions

For first test case problem, a square blockage of 0.05m of length, width, and high is placed inside the rectangular domain given in the benchmark problem 1 (figure 5.2.1), the domain is discretized using 8-noded hexahedral isoparametric elements giving a finite element mesh of 8912 nodes, and 7392 elements (figure 5.4.2). The boundary conditions given by figures 5.4.3a and 5.4.3b are as the ones specified for the benchmark problem 1 but with additional no slip boundary conditions (u, v, and w set to zero) around the six faces of the square blockage.

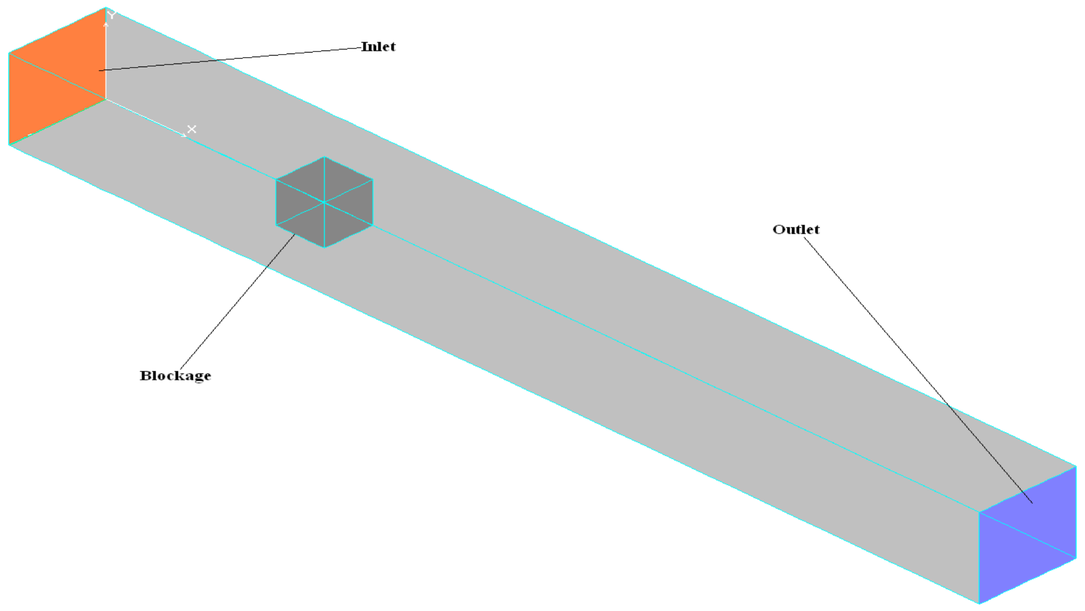


**Figure 5.4.1:** Computational domain for the test case 1 with a big square blockage.

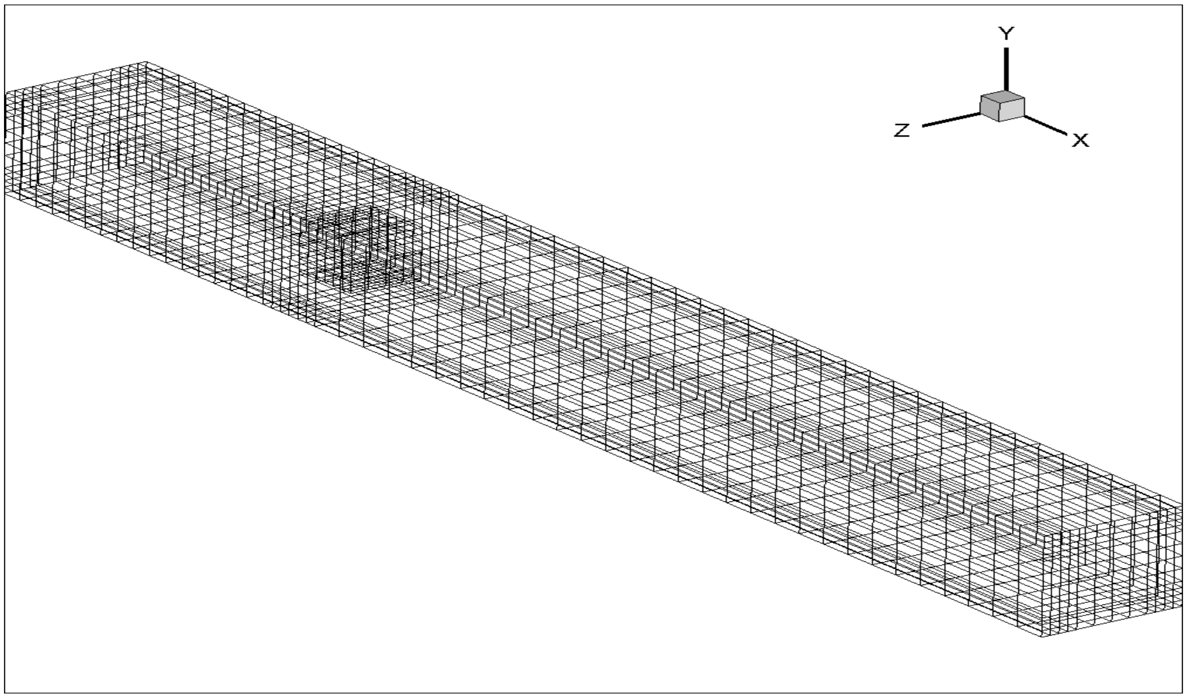


**Figure 5.4.2:** Finite element mesh for test case 1.

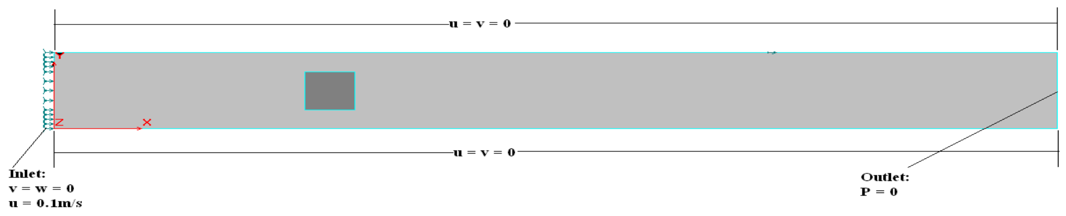


**Figure 5.4.3a:** 2-D schematic representation of the boundary condition in the xy plane (test case 1).

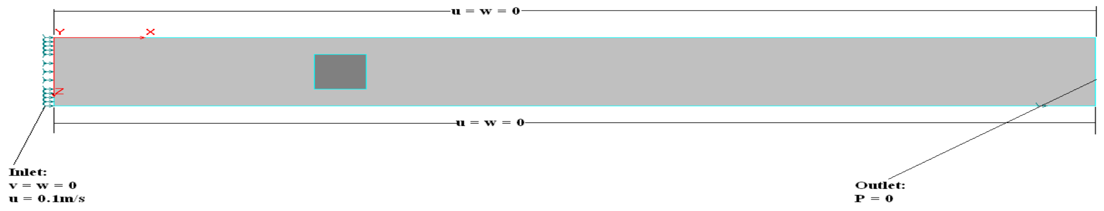


**Figure 5.4.3b:** 2-D schematic representation of the boundary condition in the xy plane (test case 1).

### 5.4.2 Results

Results obtained are given by figures 5.4.9 through 5.4.16, and in contrast to the benchmark problem 1 where there was no obstruction, one notes an increase for the pressure values for this case, pressure decrease in the direction of the flow (figure 5.4.4) from a maximum values of 22000Pa around the inlet to 0Pa initially set at the outlet (figure 5.4.5). Figures 5.4.6 through 5.4.8 also show an increase for the velocity with values varying from 0.02 $ms^{-1}$ to 0.2 $ms^{-1}$ compared to 0.01 $ms^{-1}$ to 0.12 $ms^{-1}$ previously obtained for benchmark problem 1. But despite this increase in velocity values, it can be seen from figures 5.4.6 through 5.4.8 that the region where fluid moves with high velocity is narrow and situated just at the top, bottom, left, and bottom sides of the obstacle. There is no fluid or fluid flowing only with a very low velocity at about 0.02 $ms^{-1}$ in most of region behind the obstacle. The average velocity of the outlet is 0.097 $ms^{-1}$ giving an outflow mass rate of 0.97 $kg.s^{-1}$ and hence a 3% error difference between the mass in and mass out of the computational domain.

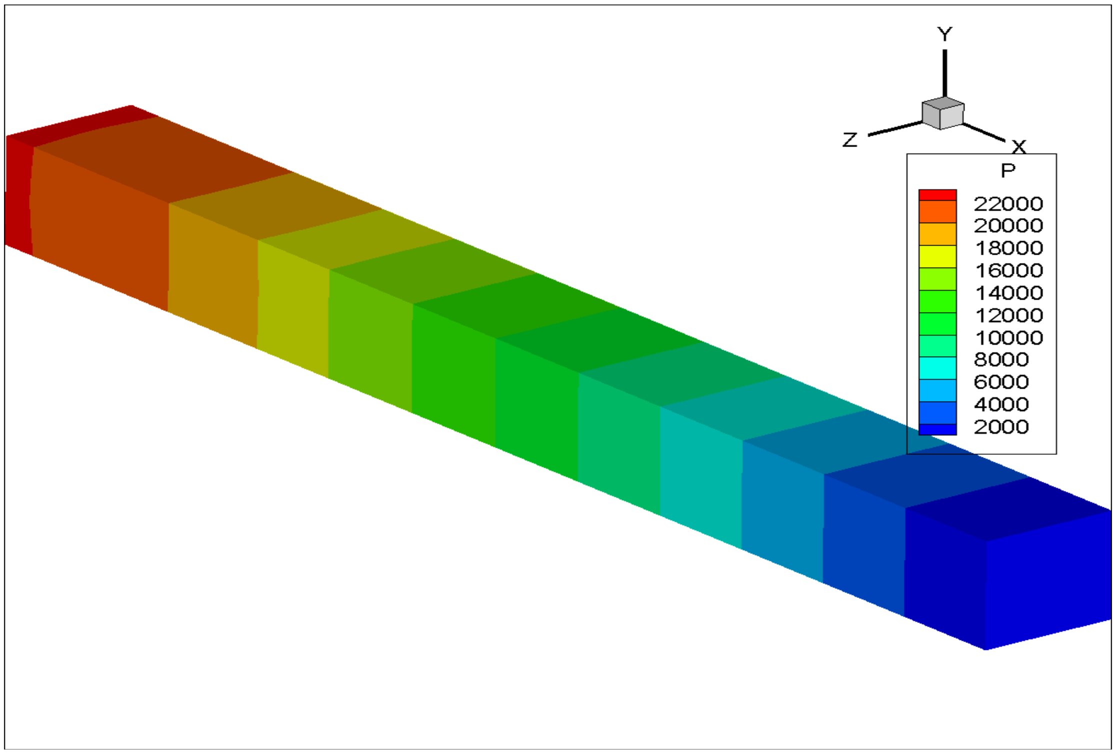


**Figure 5.4.4:** Pressure distribution (test case 1).

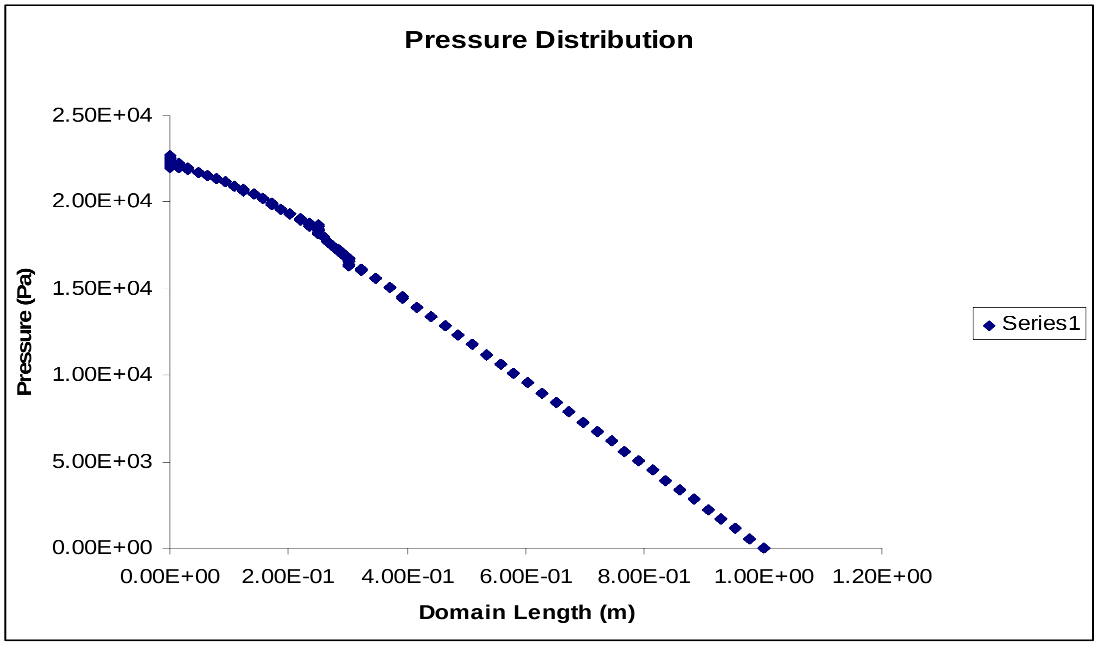


**Figure 5.4.5:** Graph of pressure distribution across the domain (test case 1).

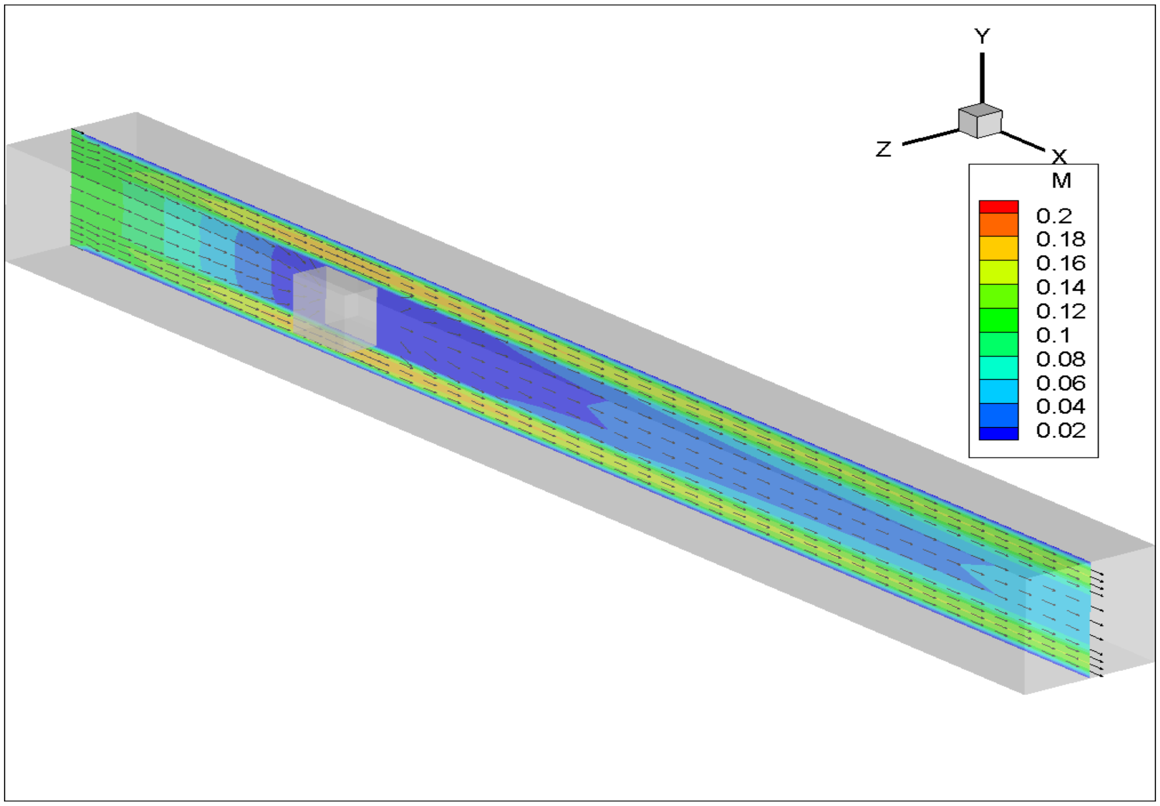


**Figure 5.4.6:** Vector plot profile coloured by the velocity magnitude contour (test case 1).

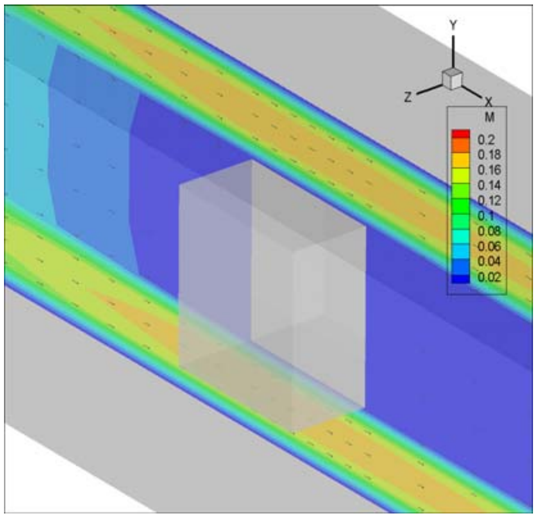


**Figure 5.4.7:** Vector plot profile coloured by the velocity magnitude contour zoomed around the obstacle in the x-y plane (test case 1).

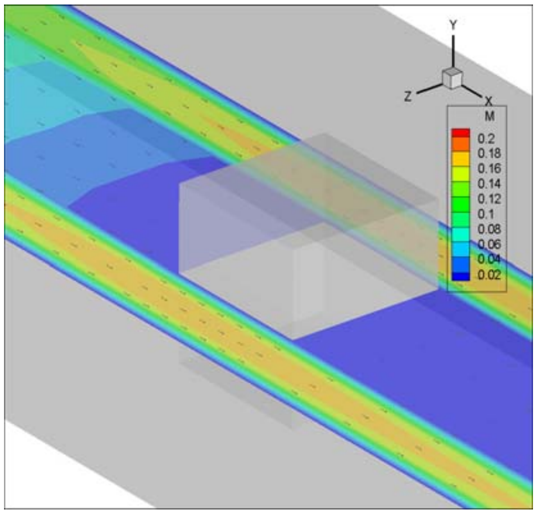


**Figure 5.4.8:** Vector plot profile coloured by the velocity magnitude contour zoo med around the obstacle in the x-z plane (test case 1).

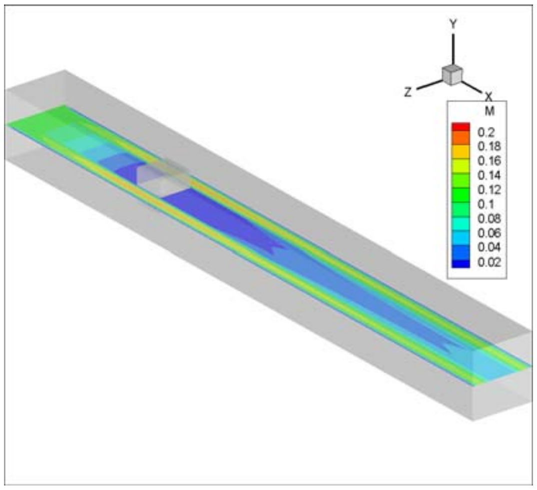


**Figure 5.4.9:** Profile of the contour of the velocity magnitude (y = 0.05m plane) (test case 1).

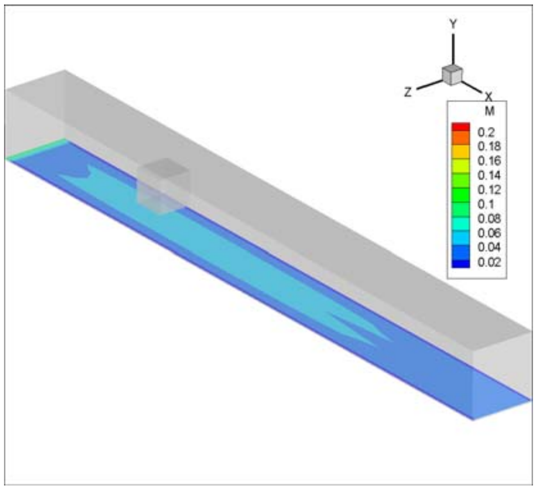


**Figure 5.4.10:** Profile of the contour of the velocity magnitude(y = 0.02m plane) (test case 1).

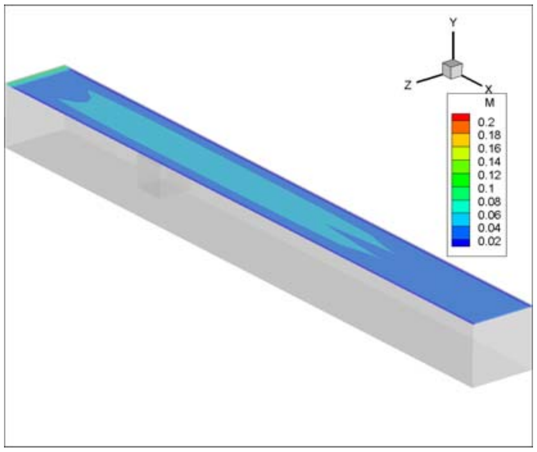


**Figure 5.4.11:** Profile of the contour of the velocity magnitude (y = 0.098m plane) (test case 1).

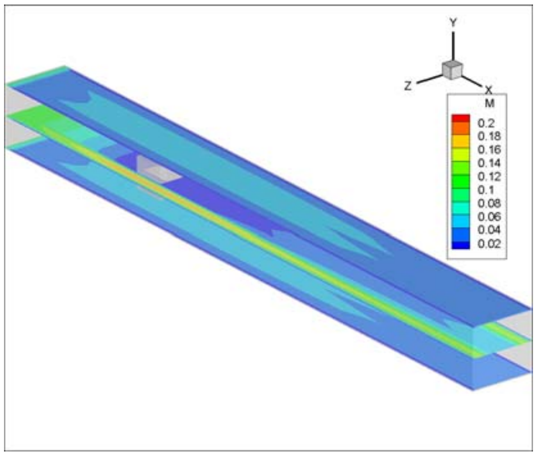


**Figure 5.4.12:** Profile of the contour of combined the velocity magnitudes (position y = 0.02, 0.05 and 0.098m) (test case 1).

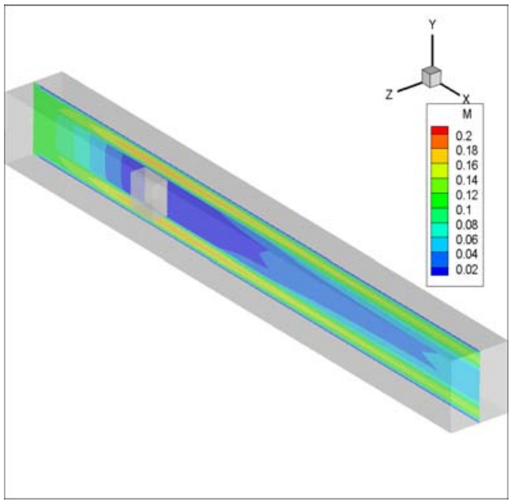


**Figure 5.4.13:** Profile of the contour of the velocity magnitude (z = 0.05m plane) (test case 1).

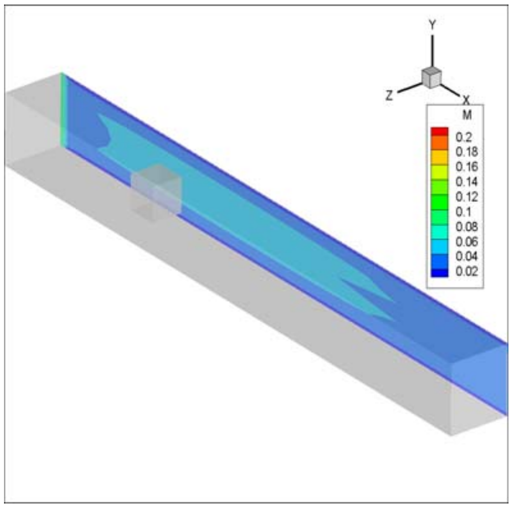


**Figure 5.4.14:** Profile of the contour of the velocity magnitude (z = 0.02m plane). (test case 1).

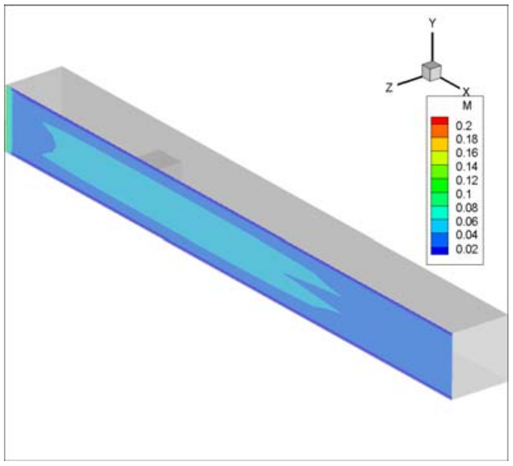


**Figure 5.4.15:** Profile of the contour of the velocity magnitude (z = 0.098m plane) (test case 1).

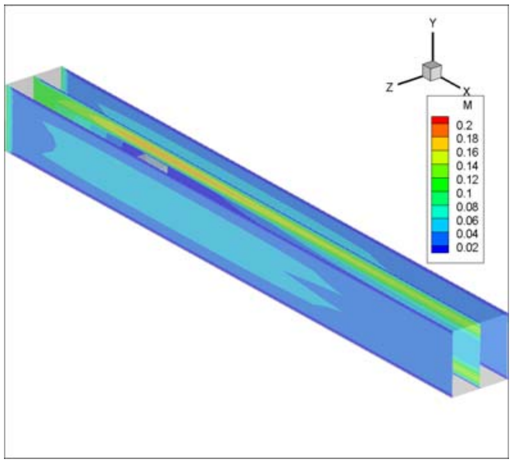


**Figure 5.4.16:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) (test case 1).

Figures 5.4.9 through 5.4.16 above represent different section contour plots of the velocity taken at the same locations as for figures 5.1.7 through 5.1.14 (benchmark problem 1). It is interesting to investigate again the profile of the velocity in the vicinity of the solid walls for this case with a square blockage and compared with the same figures obtained for benchmark problem 1 where there was no obstruction inside the computational domain.

As it can be noted from these plots, the boundary layers for this test case are wider, higher, but shorter in length compared to those obtained for the benchmark problem 1. This change of shape is certainly due to the presence of the blockage that disrupts the fluid movement.

## 5.5 Test case 2: Flow in a duct past a small square obstacle (0.025 × 0.025 × 0.025m)

### 5.5.1 Computational domain and boundary conditions

In this second test case, the size of the blockage is reduced by half (0.025m × 0.025m × 0.025m) in order to investigate whether the obstruction size has an effect on the flow. The geometry (figure 5.4.17), the finite element mesh (figure 5.4.18), and the imposed boundary conditions (figure 5.4.19a, and figure 5.4.20b) for this case are similar to those for the benchmark problem 1.

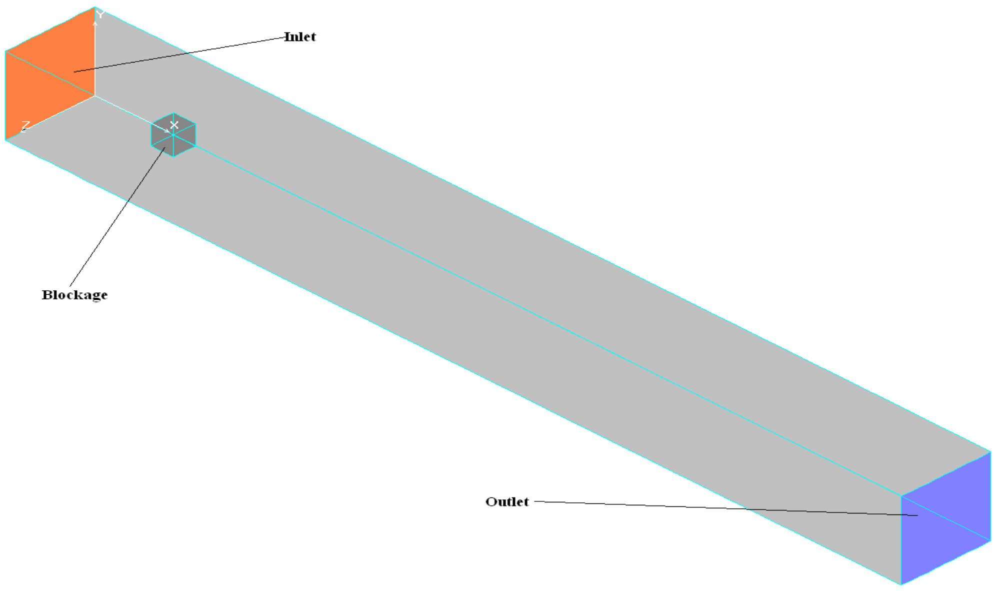


**Figure 5.5.1:** Computational domain for test case 2.

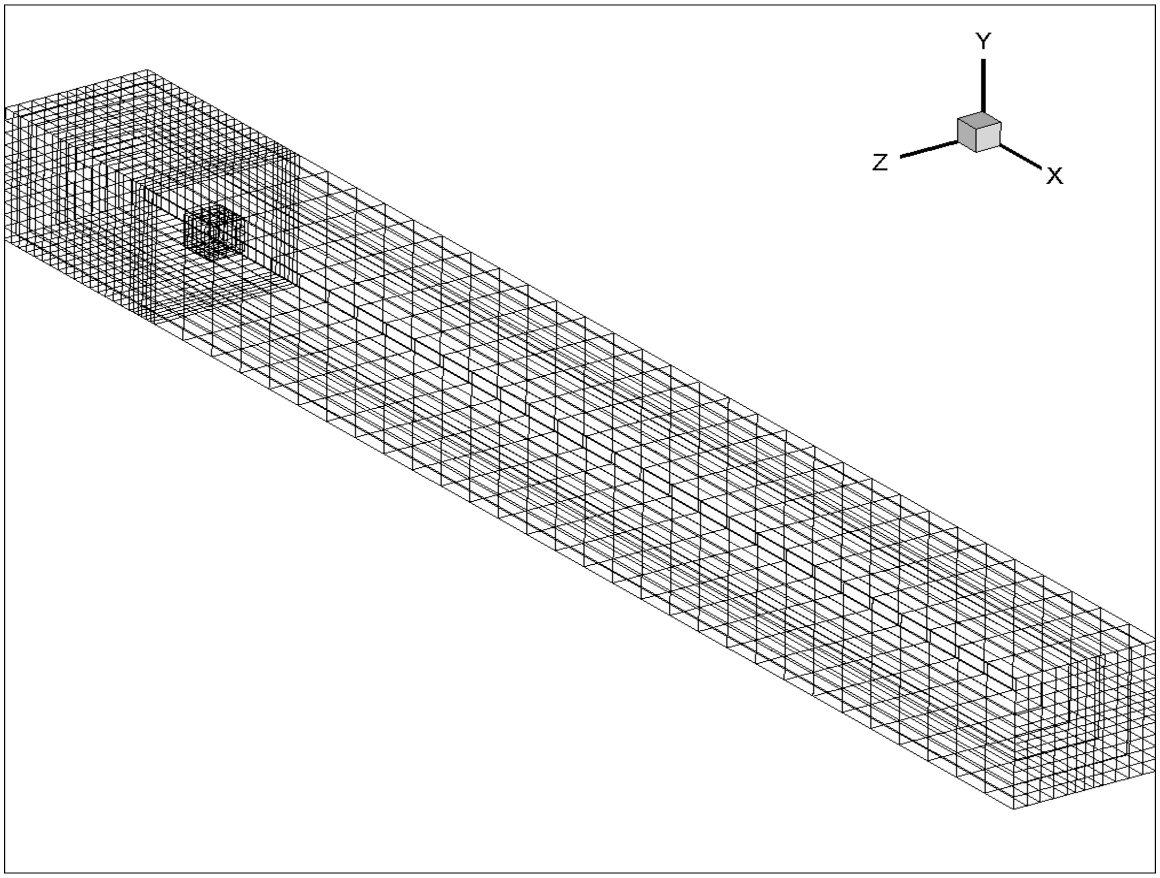


**Figure 5.5.2:** Finite element mesh for test case 2.

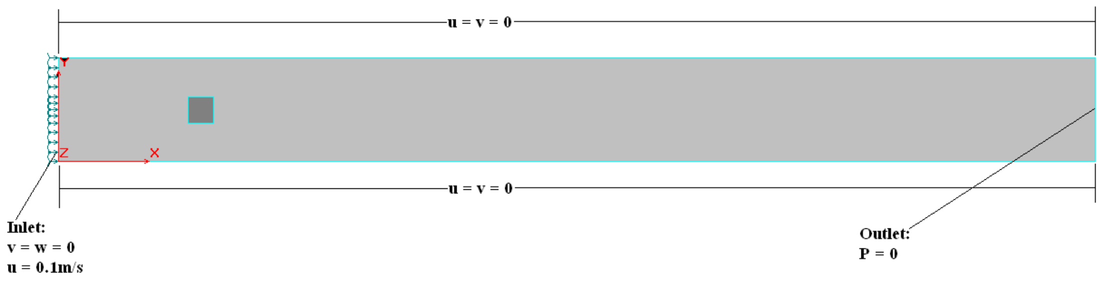


**Figure 5.5.3a:** 2-D schematic representation of the boundary condition in the xy plane (test case 2).

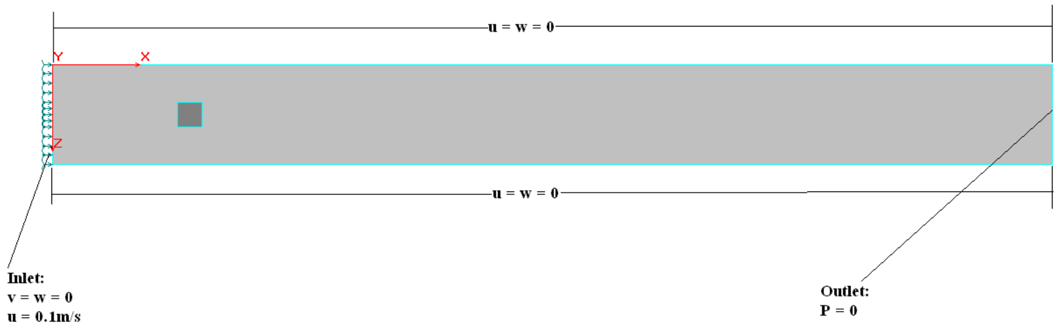


**Figure 5.5.3b:** 2-D schematic representation of the boundary condition in the xz Plane (test case 2).

### 5.4.2 Results

The results obtained for this test case with the square blockage size halved showed that pressure decreased in the direction of the flow as obtained for the previous case and with an increase in pressure values starting for 0Pa to 55000Pa (figures 5.4.20 and 5.4.21) compared to the same figure obtained for test case 1; which was from 0Pa to 22000Pa (figures 5.4.4 and 5.4.5), this pressure rise reduced the magnitude of velocity which vary in this case from the initial values of 0.01 $ms^{-1}$ to 0.16 $ms^{-1}$ (figure 5.4.22). Given that the obstacle is small, the fluid entering the domain have more free space around it to move freely and this can be seen from figures 5.4.23 and 5.4.24 (orange coloured region representing the region of the domain where the flow occurs with high velocity and at about 0.14 $ms^{-1}$ ). The fluid exits the domain with an average velocity of 0.0975 $ms^{-1}$ giving a mass outflow rate of 0.975 $kg.s^{-1}$, and hence an error in mass balance of 2.5%. Thus reducing the size of the obstacle by half improves the mass balance by 0.5%.

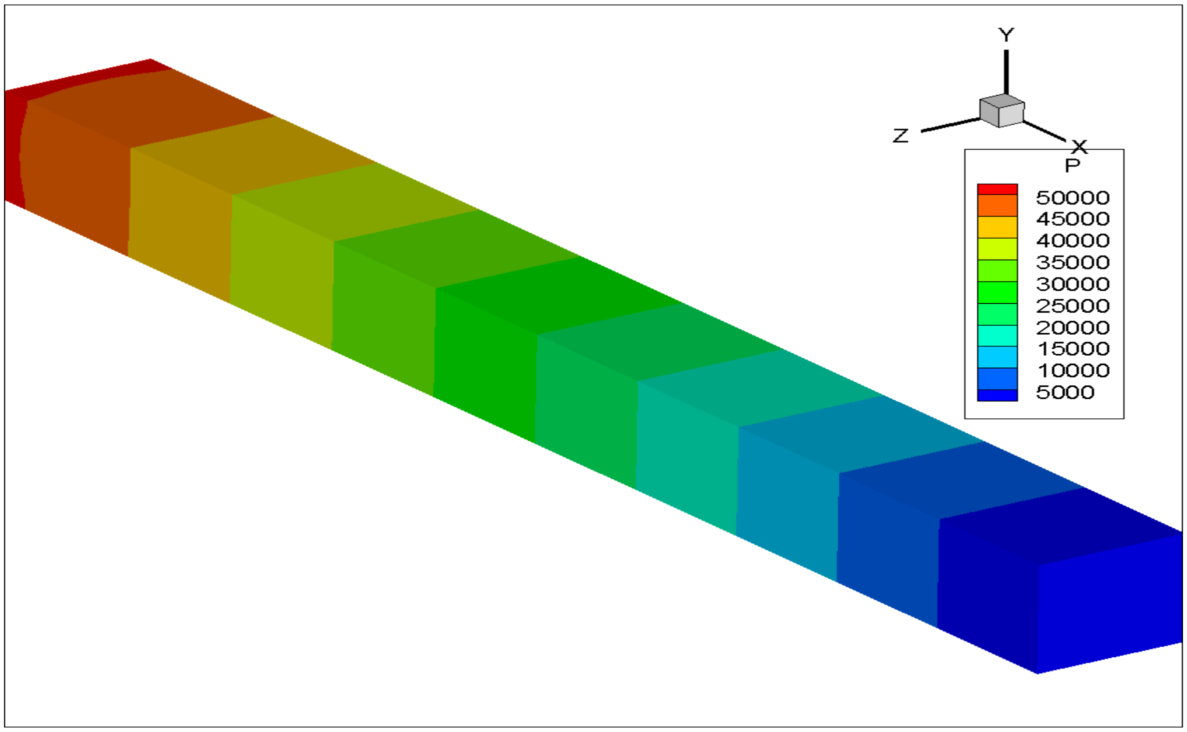


**Figure 5.5.4.:** Pressure distribution (test case 2).

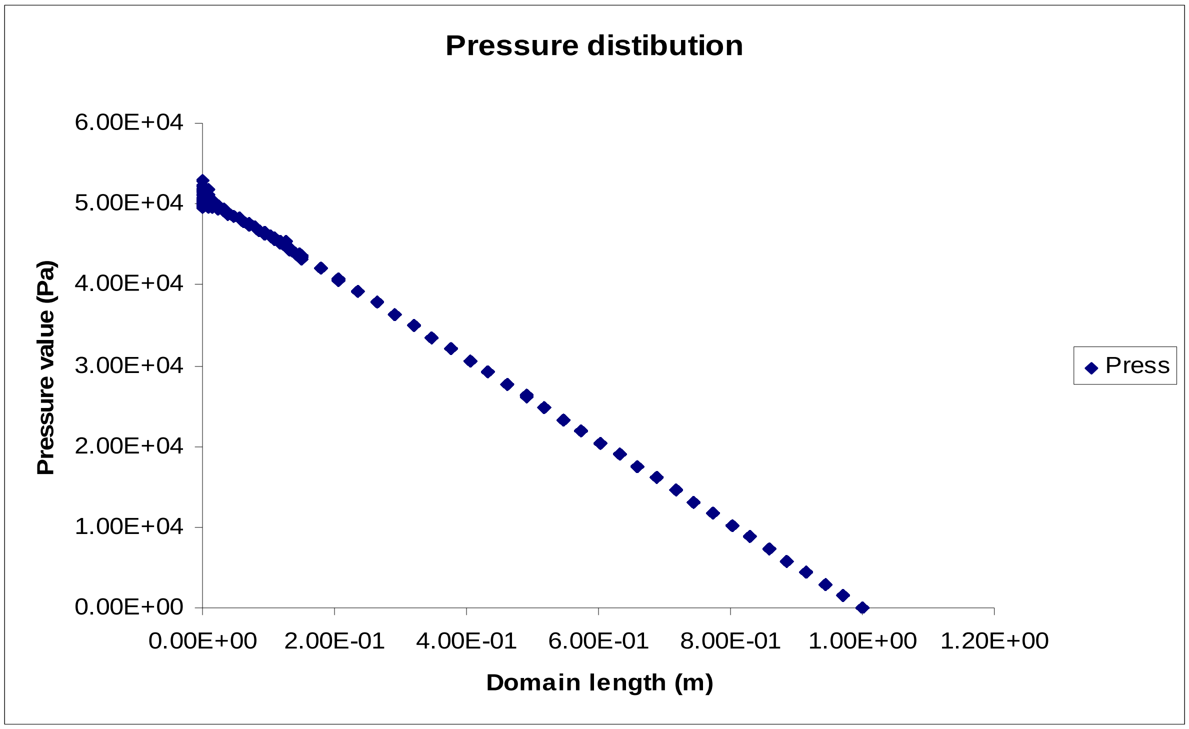


**Figure 5.5.5:** Graph of pressure distribution across the domain (test case 2).

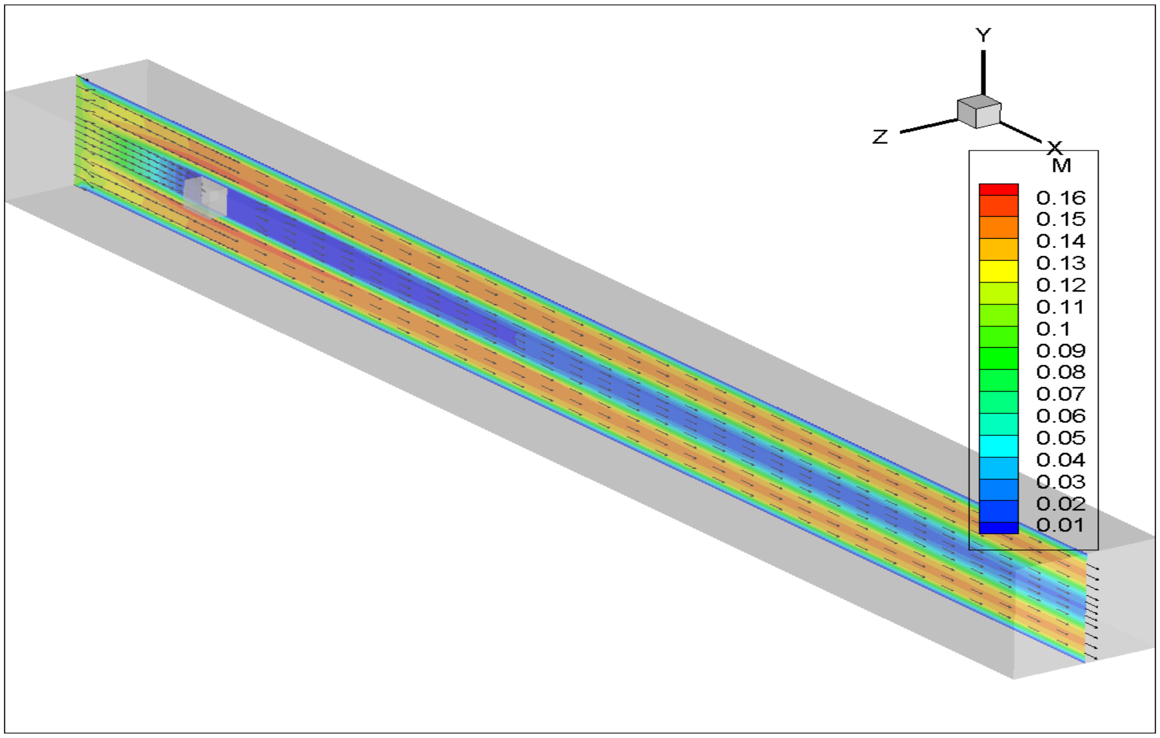


**Figure 5.5.6:** Vector plot profile coloured by the velocity magnitude contour (test case 2. ).

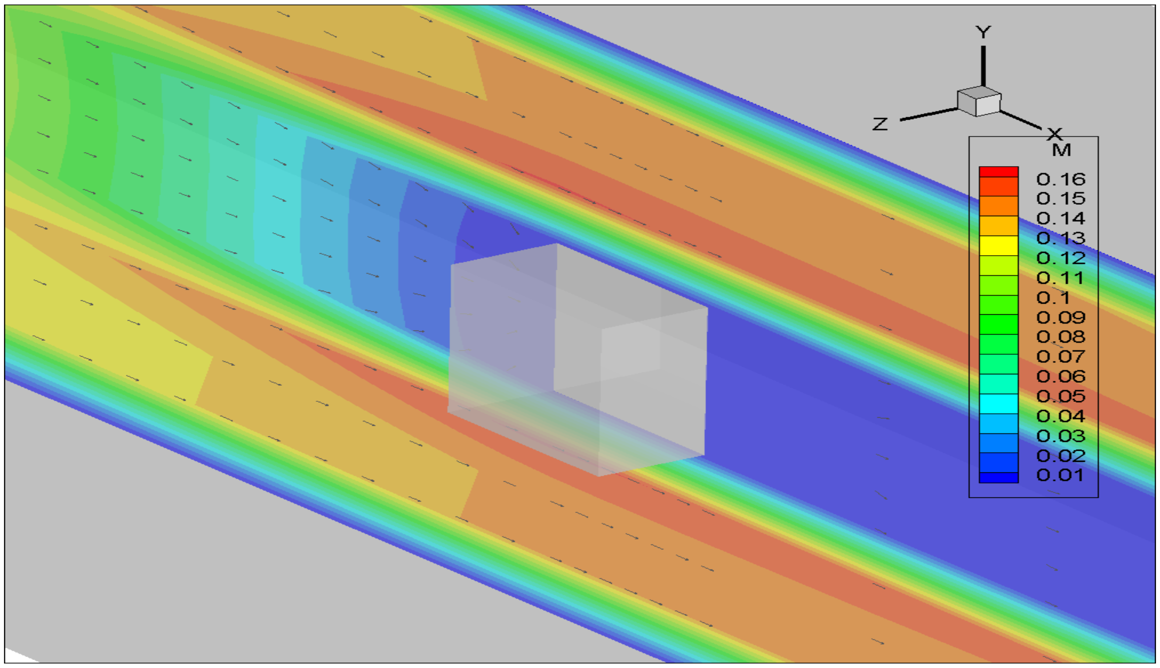


**Figure 5.5.7:** Vector plot profile coloured by the velocity magnitude contour zoomed around the obstacle in the x-y plane (test case 2).

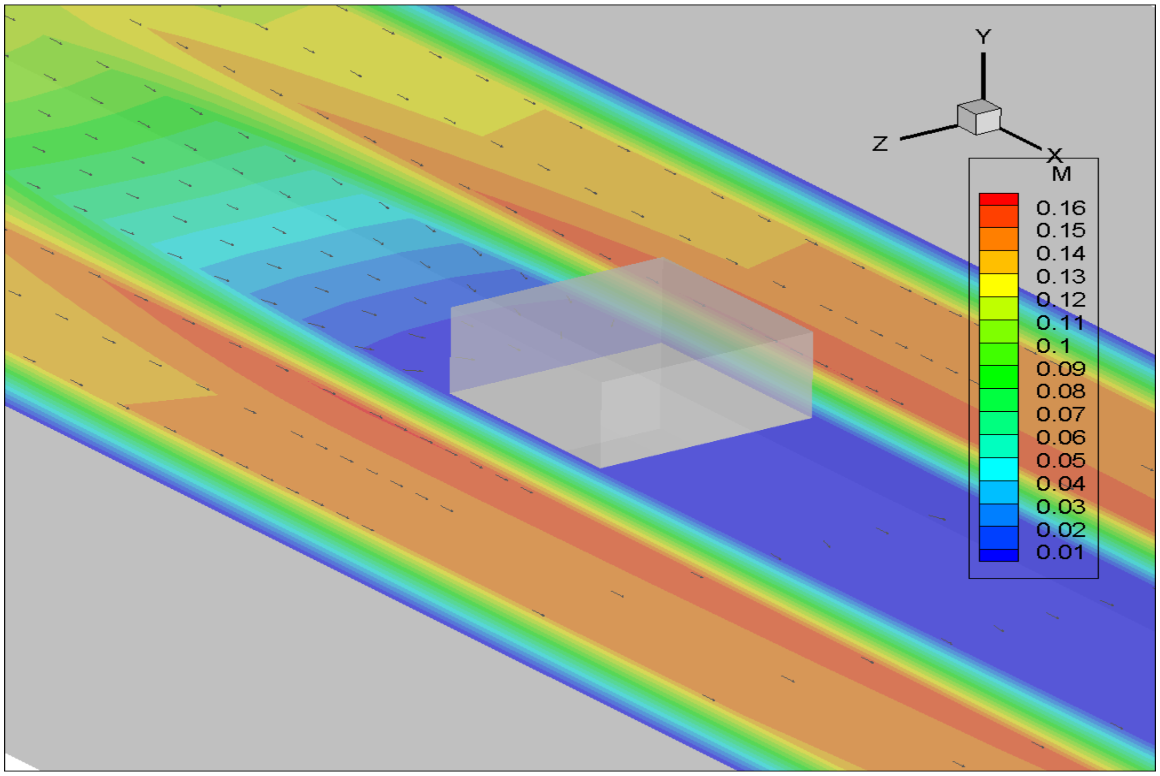
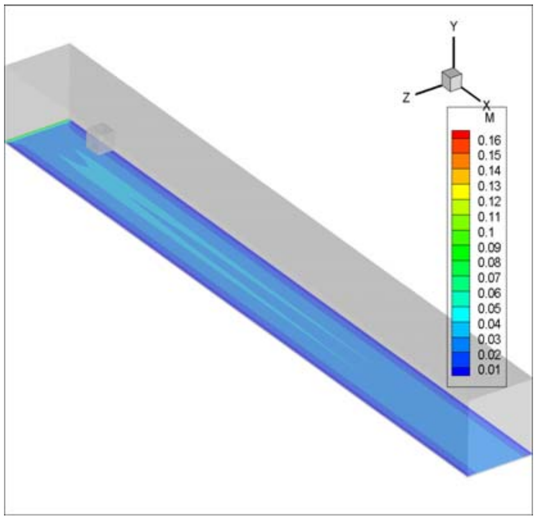


**Figure 5.5.8:** Vector plot profile coloured by the velocity magnitude contour zoomed around the obstacle in the x-z plane (test case 2).



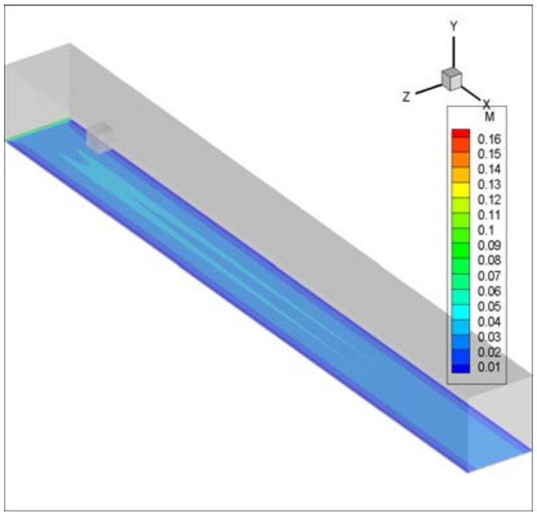


**Figure 5.5.9:** Profile of the contour of the velocity magnitude (y = 0.5m plane) test case 2.

**Figure 5.5.10:** Profile of the contour of the velocity magnitude(y = 0.02m plane) test case 2.

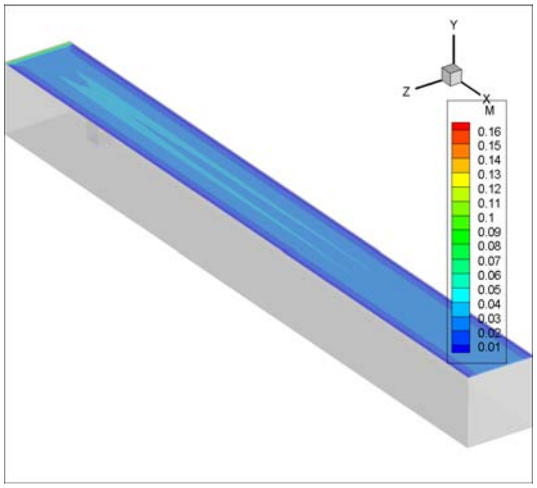


**Figure5.5.11:** Profile of the contour of thevelocity magnitude(y = 0.098m plane) test case 2.

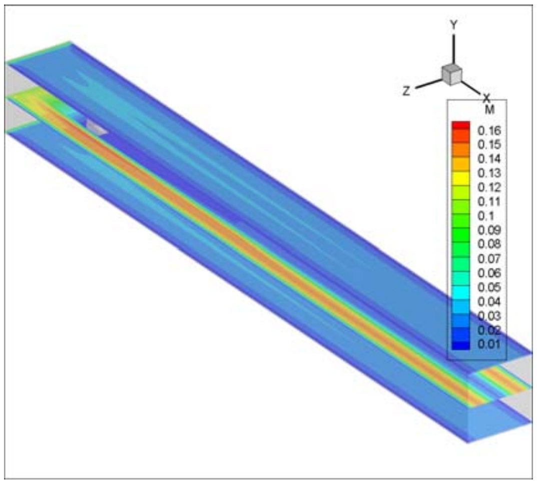


**Figure 5.5.12:** Combined profiles of the velocity magnitude (y = 0.02m, y = 0.05m, and y = 0.098m planes). test case 2.

Figures5.4.25 through 5.4.32 of this section represent different section contour plots of the velocity taken at the same locations as for the benchmark problem 1. These plots show that in the vicinity of the solid walls, the shapes of the boundary layers are similar to those obtained for test case 1 problem but the only difference is that there are smaller.

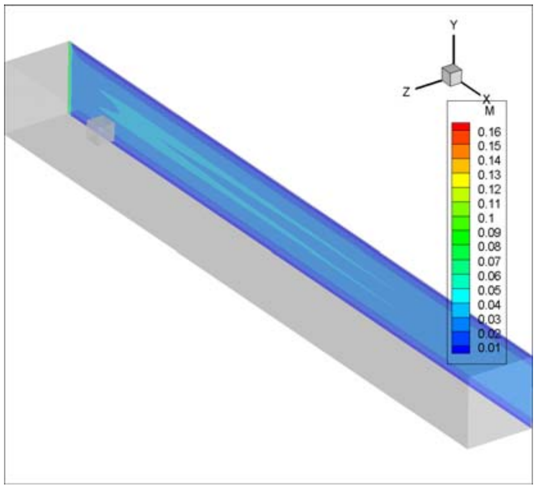


**Figure 5.5.13:** Profile of the contour of the velocity magnitude (z = 0.5m plane) test case 2.

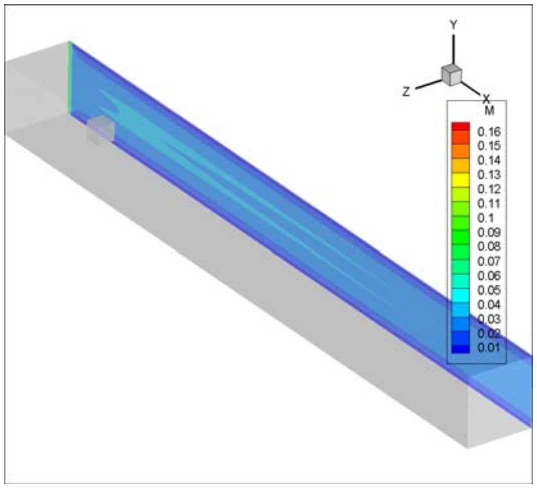


**Figure 5.5.14:** Profile of the contour of the velocity magnitude(z = 0.02m plane) test case 2.

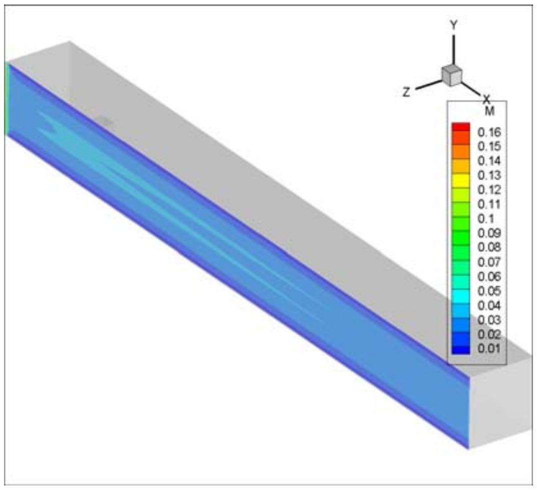


**Figure 5.5.15:** Profile of the contour of the velocity magnitude (z = 0.098m plane) test case 2.

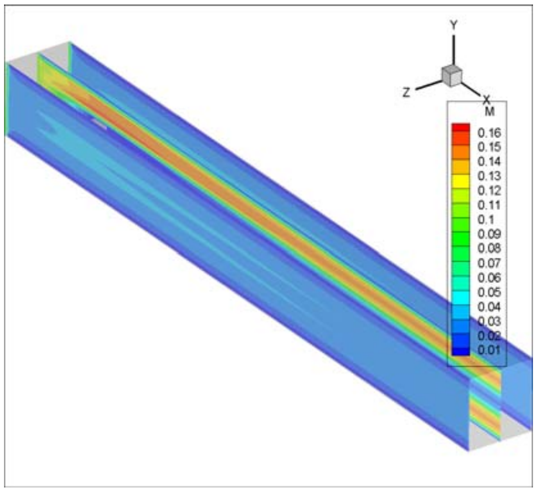


**Figure 5.5.16:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) test case 2.

## 5.6 Test case 3: Flow in a duct past a big square obstacle (0.05 × 0.075 × 0.05m) with an outlet positioned normal to the direction of the flow.

### 5.6.1 Computational domain and boundary conditions

Seeing that the results obtained for test cases 1 and 2 showed that the case with a bigger obstruction (test case 1) had a bad percentage (3%) error in mass balance, the next simulation (test case 3) that follows will be carry out over a domain including a slightly bigger obstruction than the one for test case 1. The size of the obstacle in this case is 0.05m, 0.70m, and 0.05m for length, height, and wide respectively, the computational domain size remain the same (see figure 5.6.1). The entire domain is discretized using 8-noded hexahedral isoparametric elements giving a finite element mesh of 8882 nodes and 7344 elements (figure 5.6.2). The three different emplacement of the outlets used for benchmark problems 1, 2, and 3 will be use for this case. The aim of the simulation in this case is to investigate whether bigger obstacles and the emplacement of the outlet will cause the violation of the mass balance and hence the reliability of the developed codes.

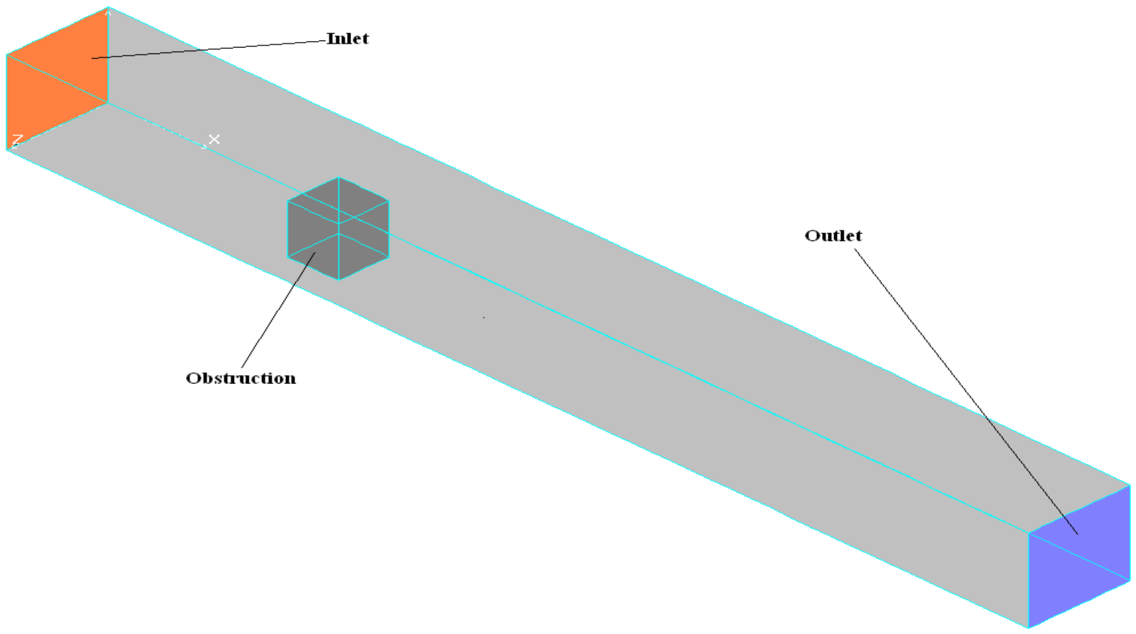


**Figure 5.6.1:** Computational domain for test case 3.

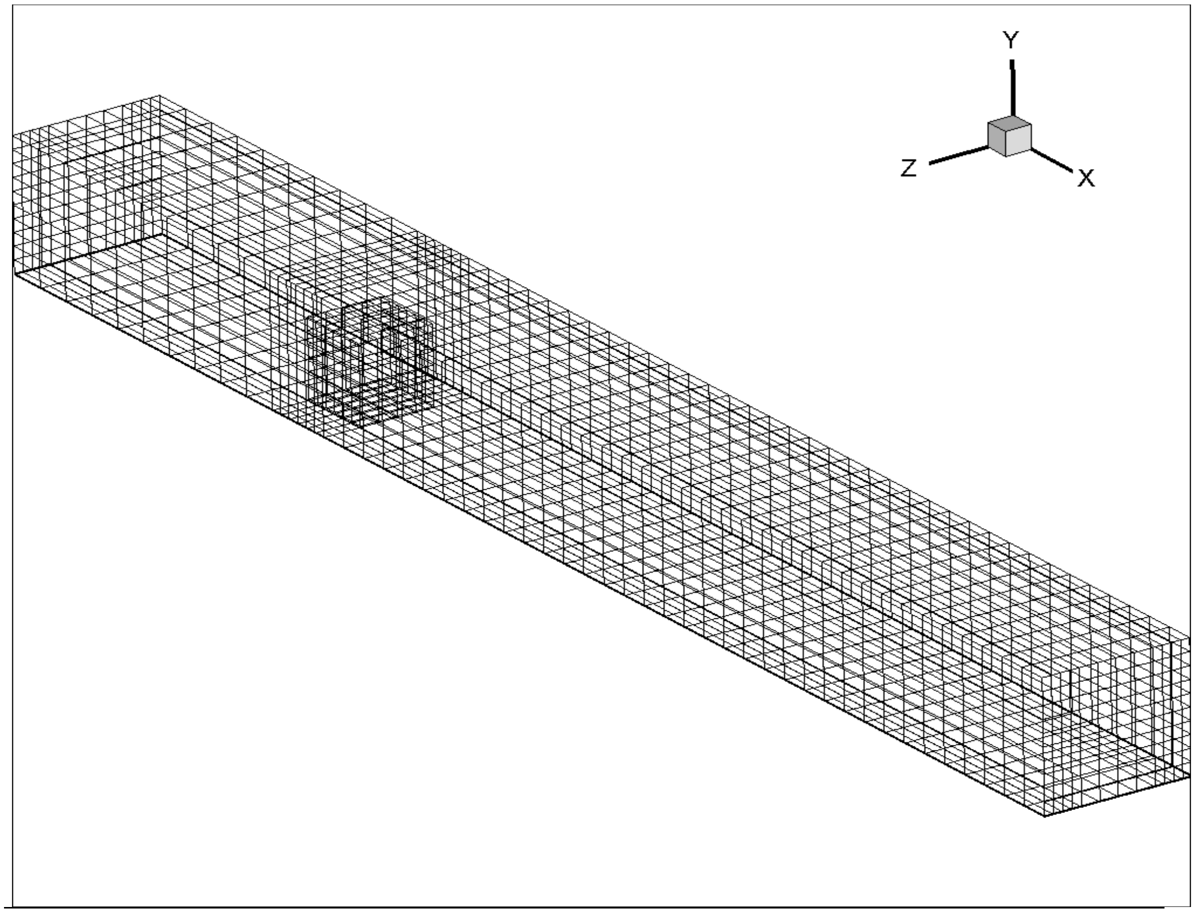


**Figure 5.6.2:** Finite element mesh for test case 3.

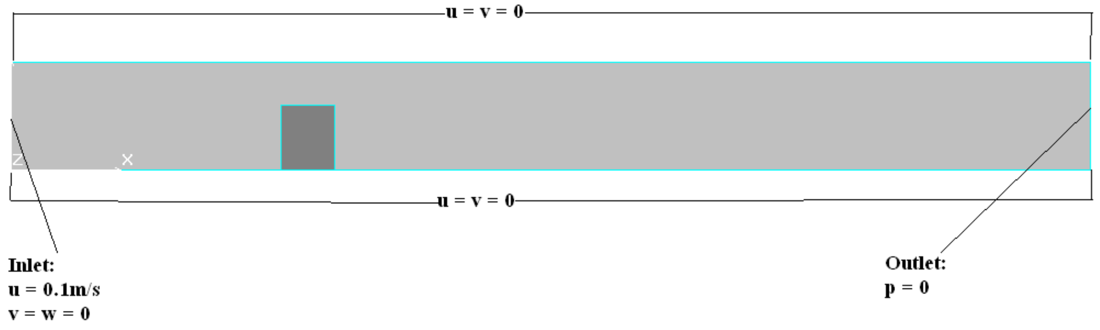


**Figure 5.6.3a:** 2-D schematic representation of the boundary condition in the xy plane (test case 3).

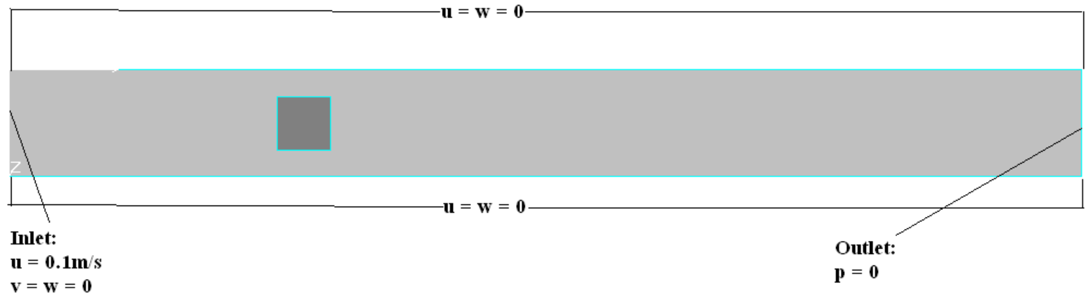


**Figure 5.6.3b:** 2-D schematic representation of the boundary condition in the xz Plane (test case 3).

## 5.6.2 Results

Increasing the obstacle size has no effect on the pressure distribution as it can seen from figures 5.6.4 and 5.6.5, the values of computed pressure obtained vary from 0Pa to 22KPa and are similar to the computed pressure obtained in test case 1. The velocity profiles on the other hand experience an increase in magnitude values, the values here vary from 0.02 $ms^{-1}$ to 0.26 $ms^{-1}$ whereas the same figure was 0.02 $ms^{-1}$ to 0.2 $ms^{-1}$ for test case 1. But as it can be seen from figures 5.6.6 and 5.6.7, the regions where the fluid moves with high velocity are very smaller. The computed average exit velocity for this case is 0.0963 $ms^{-1}$ giving a mass outflow of 0.963 $kg.s^{-1}$ and an error mass balance difference of 3.7% which is bigger than the 3% obtained in test case 1 but still in within the tolerated limit. Hence, even with a big obstacle as used in this case, the mass balance criterion is not violated, that is the developed computational scheme still valid regardless of the size of the obstruction.

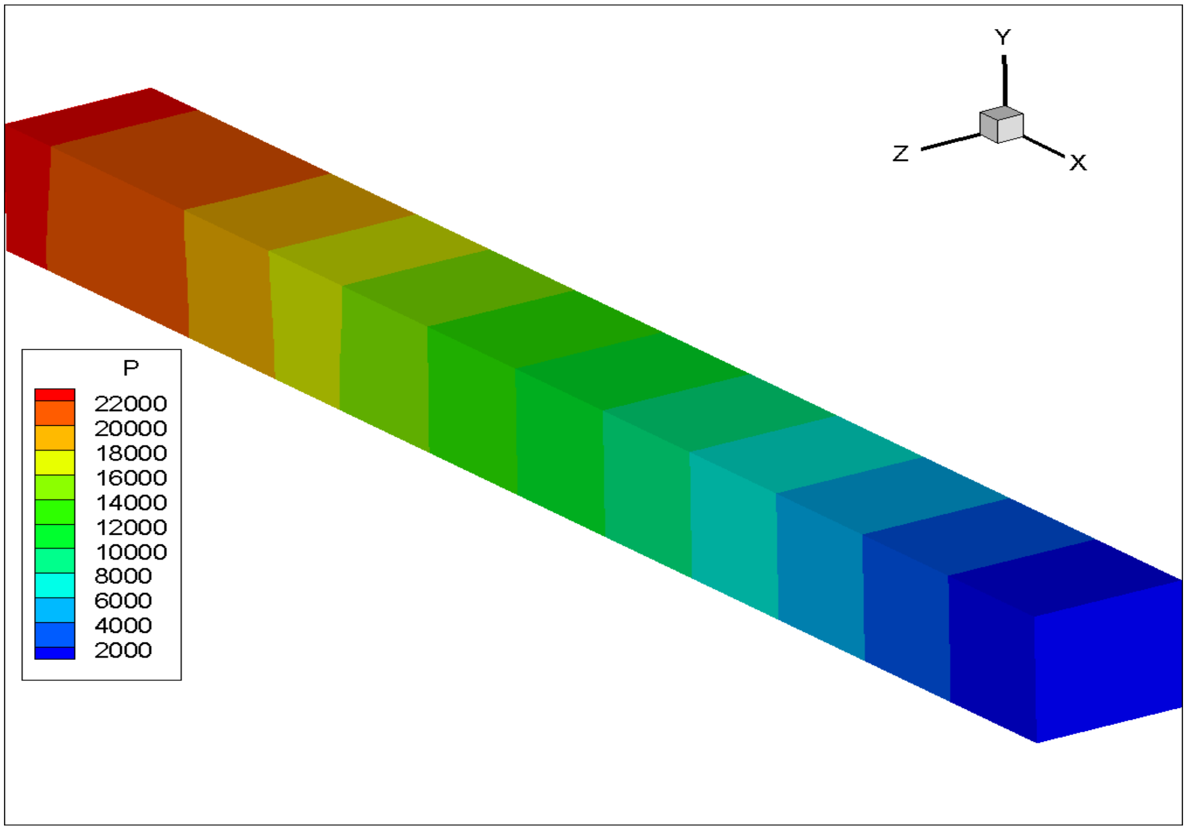


**Figure 5.6.4:** Pressure distribution. for test case 3.

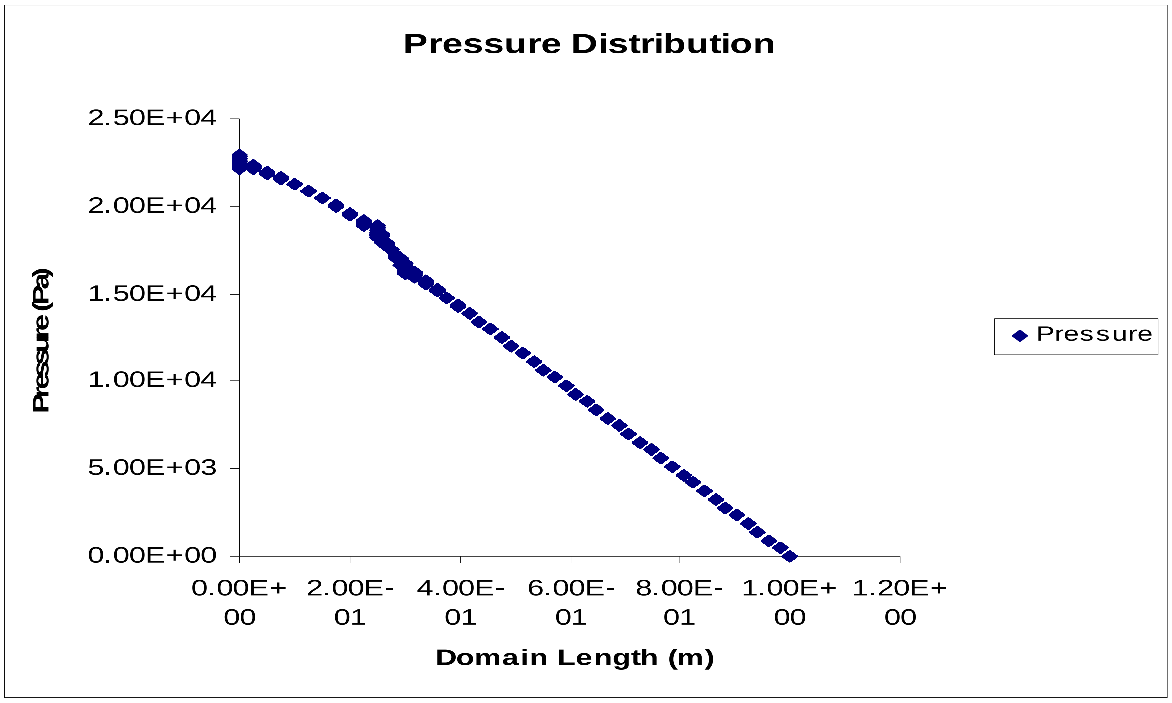


**Figure 5.6.5:** Graph of pressure distribution across the domain (test case 3).

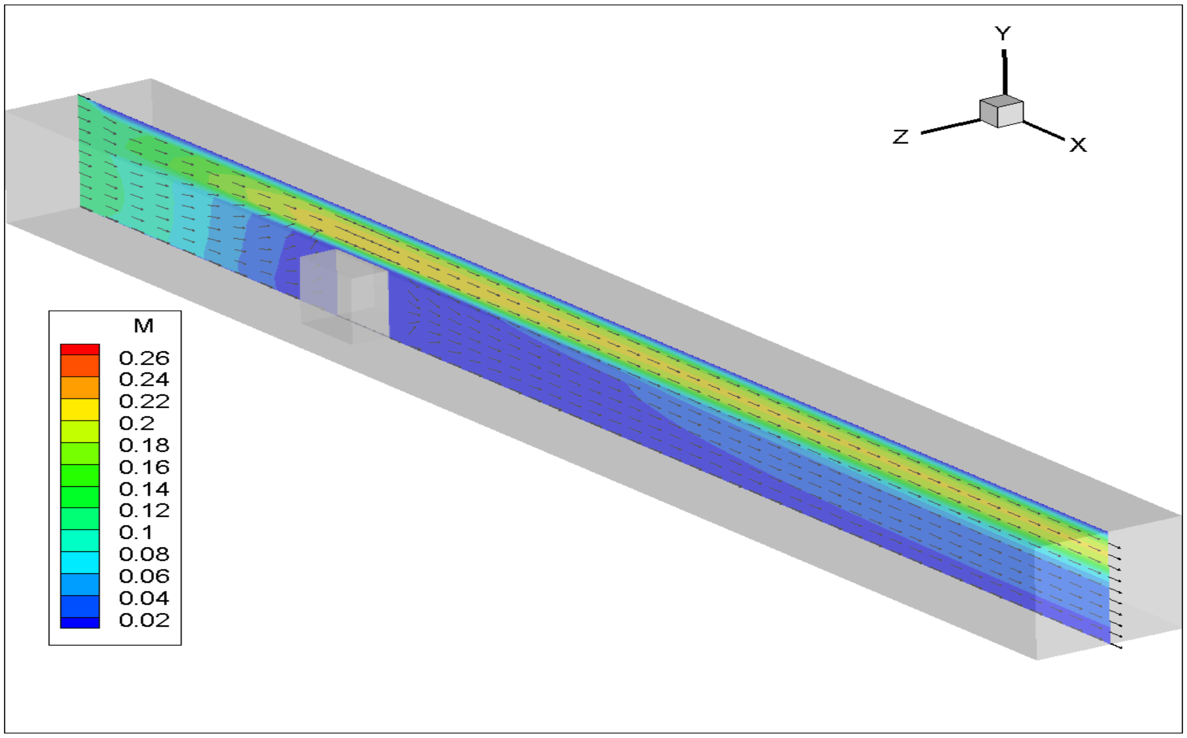


**Figure 5.6.6:** Vector plot profile coloured by the velocity magnitude contour in the z = 0.5m plane (test case 3).

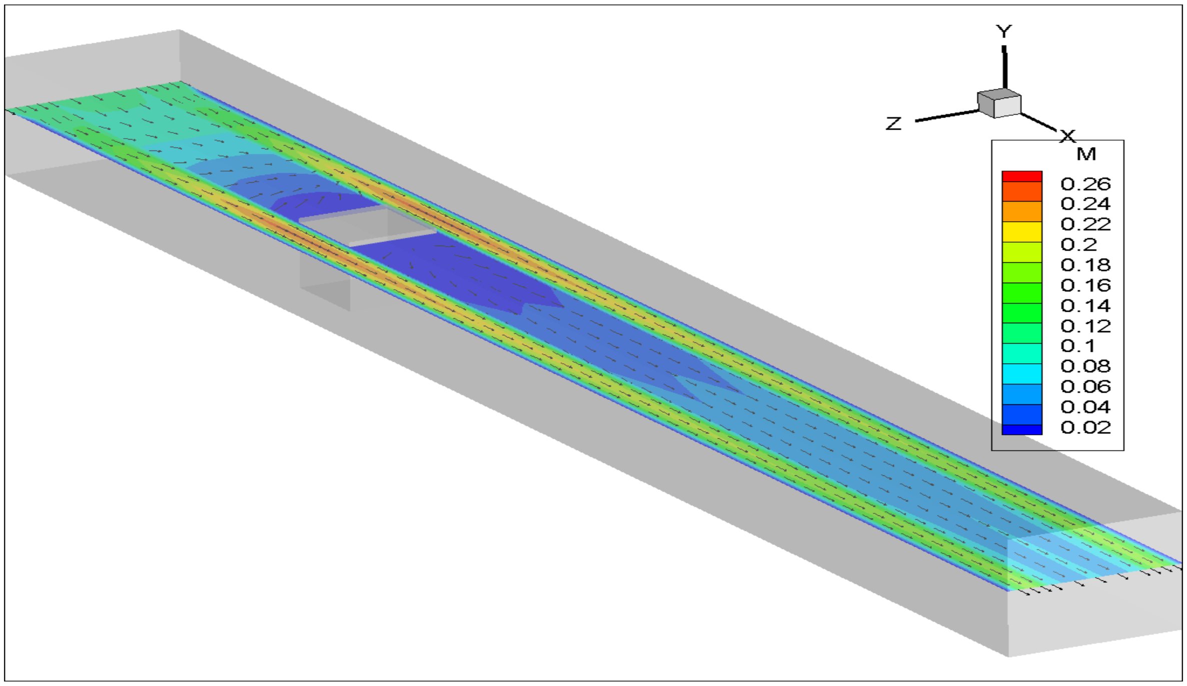


**Figure 5.6.7:** Vector plot profile coloured by the velocity magnitude contour in the y = 0.5m plane (test case 3).

Figure 5.6.8 (in the z = 0.05m plane) shows the velocity vector profile around the obstruction, the contour pressure is chosen in this case as background for the velocity vectors because it provides better view than the velocity magnitude contour. The profile is as expected, one can see that in the vicinity of the blockage, the fluid deviates and move toward the top of the computational domain where it can moves freely, then once passing the obstruction, some the fluid move back down and continues to flow toward the exit.

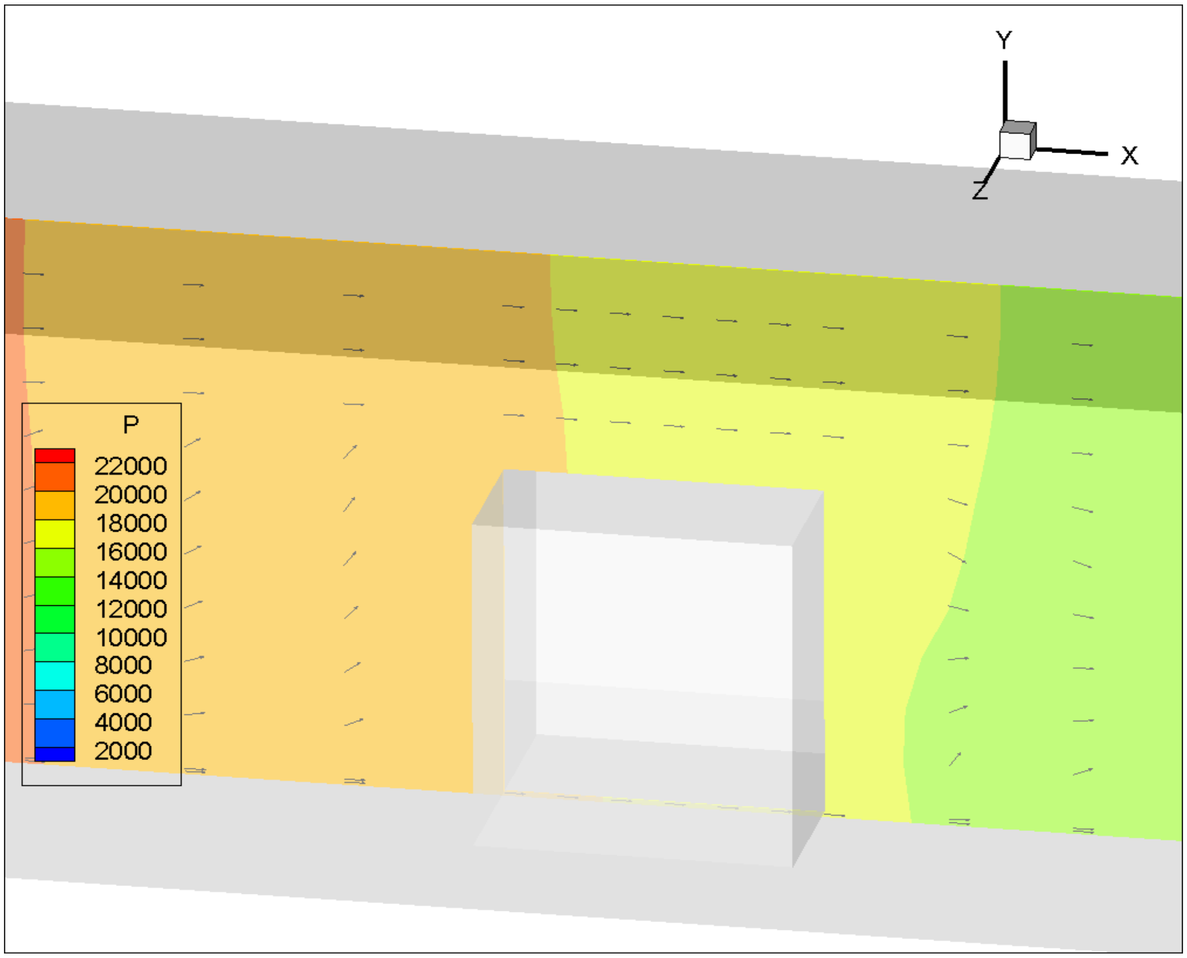


**Figure 5.6.8:** Vector plot profile coloured by the pressure contour zoomed around the obstruction (test case 3).

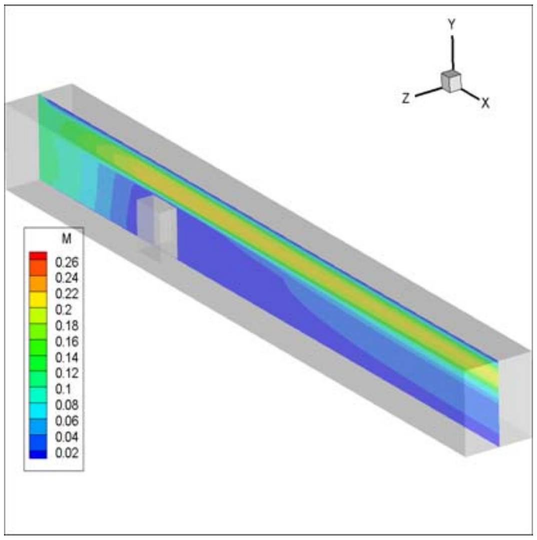


**Figure 5.6.9:** Profile of the contour of the velocity magnitude (z = 0.5m plane) test case 3.

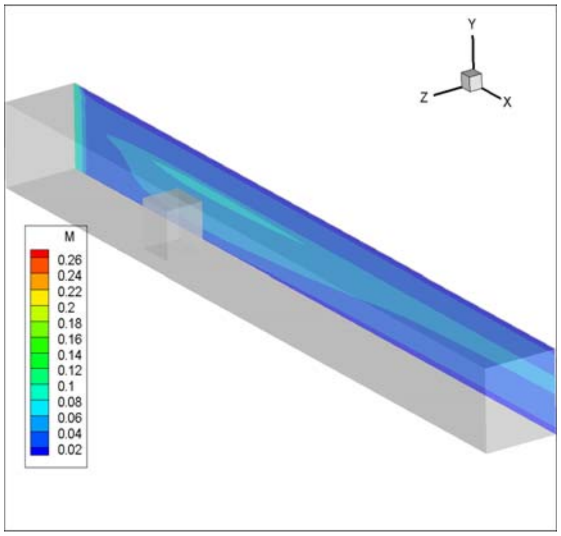


**Figure 5.6.10:** Profile of the contour of the velocity magnitude (z = 0.02m plane). test case 3.

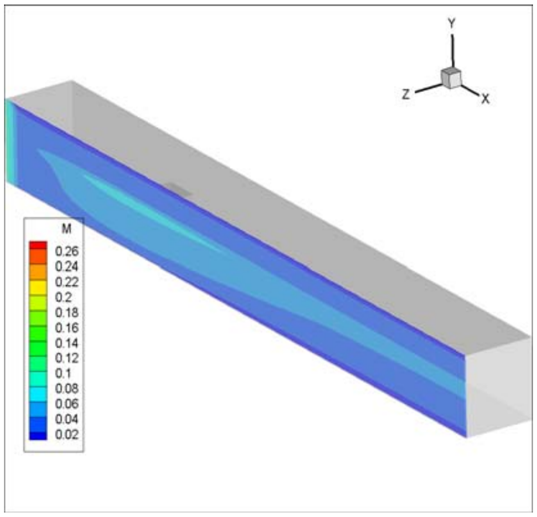


**Figure 5.6.11:** Profile of the contour of the velocity magnitude (z = 0.098m plane) test case 3.

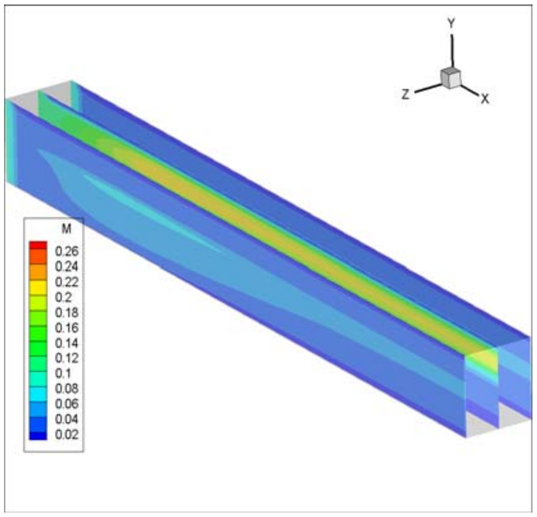


**Figure 5.6.12:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) test case 3.

Figures 5.6.9 through 5.6.12 show the section plots of the contours of the magnitude of the velocity plotted in the same position than those plotted for test case 1 problem (figures 5.4.9 through 5.4.16). It can be seen that when the size of the obstruction is increased and that it is placed at the bottom of the domain, the boundary layers obtained in the z = 0.02m and z = 0.098m planes are similar, bigger and start from a region close to the inlet to end at the outlet whereas for the test case 1 problem (figures 5.4.9-5.4.12) where the obstruction was smaller and placed far from the bottom solid wall, the boundary layers obtained at the same positions where smaller and shorter in length. Another difference is that in the planes y = 0.02m and y = 0.098m, for the present case, the shape of the boundary layer at position y = 0.02m (figure 5.6.14) differs from the one at position y = 0.098m whereas the same figure obtained for test case 1 problem (figures 5.4.13-5.4.16) showed that they were similar.

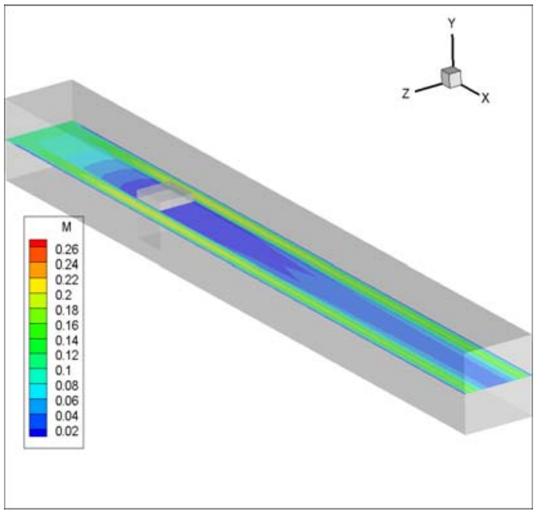


**Figure 5.6.13:** Profile of the contour of the velocity magnitude (y = 0.5m plane) test case 3.

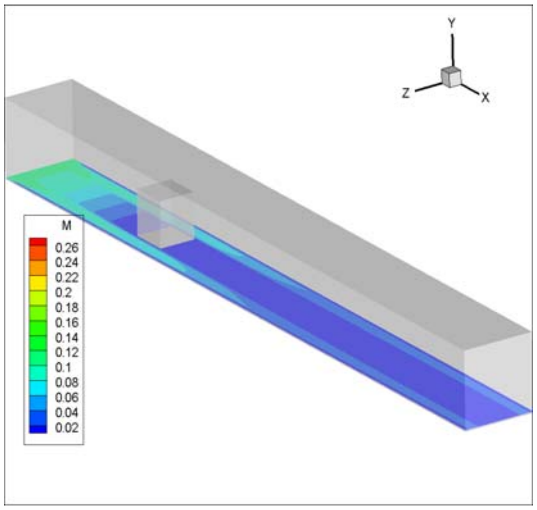


**Figure 5.6.14:** Profile of the contour of the velocity magnitude(y = 0.02m plane) test case 3.

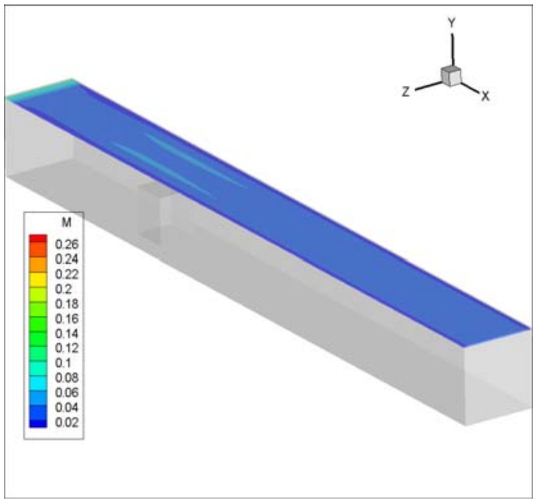


**Figure 5.6.15:** Profile of the contour of the velocity magnitude (y = 0.098m plane) test case 3.

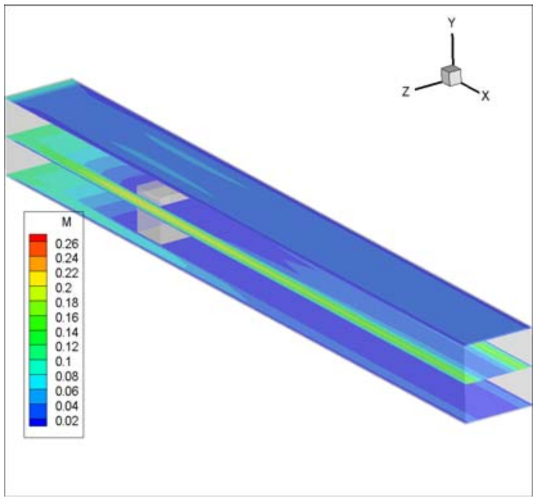


**Figure 5.6.16:** Profile of the contour of combined the velocity magnitudes (position y = 0.02, 0.05 and 0.098m) test case 3.

## 5.7 Test case 4: Flow in a duct past a big square obstacle (0.05 × 0.075 × 0.05m) with an outlet placed at the top end of the domain.

### 5.7.1 Computational domain and boundary conditions

The dimension of the domain (figure 5.4.49), finite element mesh (figure 5.4.50), and boundary conditions for this test case are similar to those given for test case 3, except the fact that here the outlet is placed at the top end of the domain.

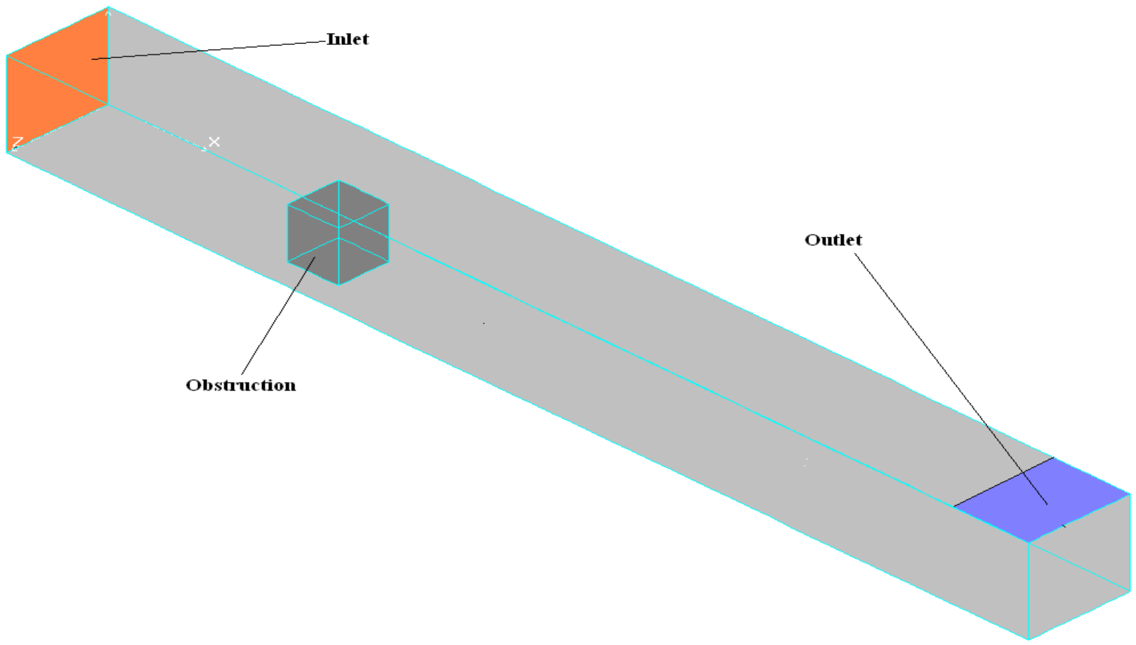


**Figure 5.7.1:** Computational domain for test case 4.

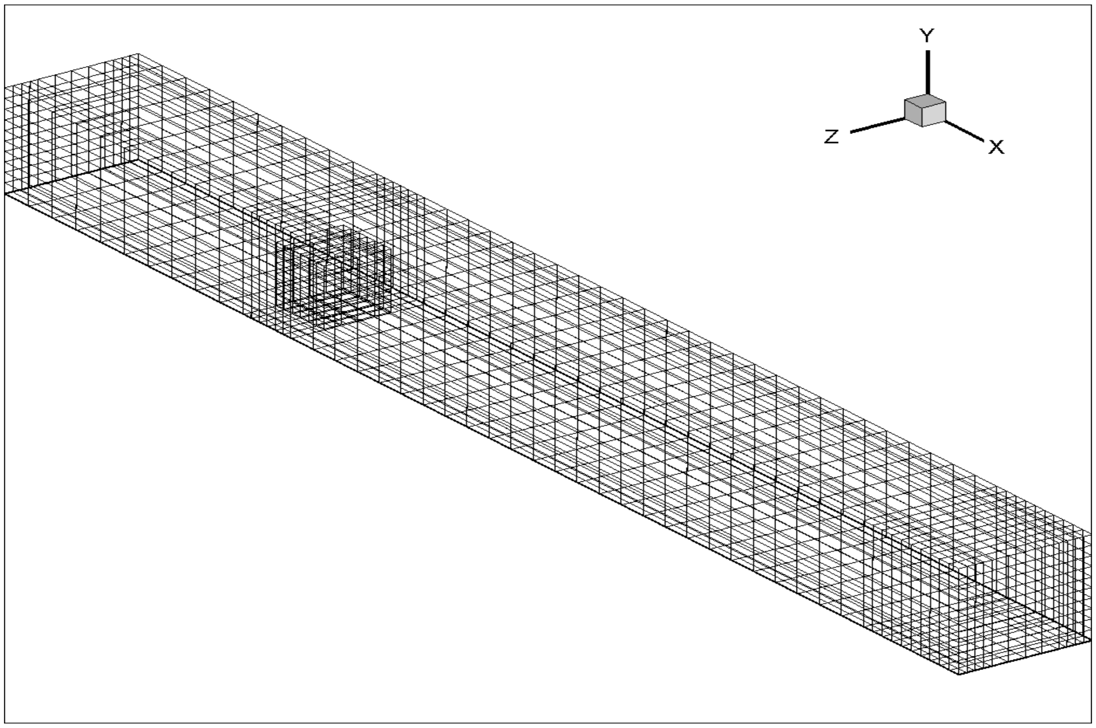


**Figure 5.7.2:** Finite element mesh for test case 4.

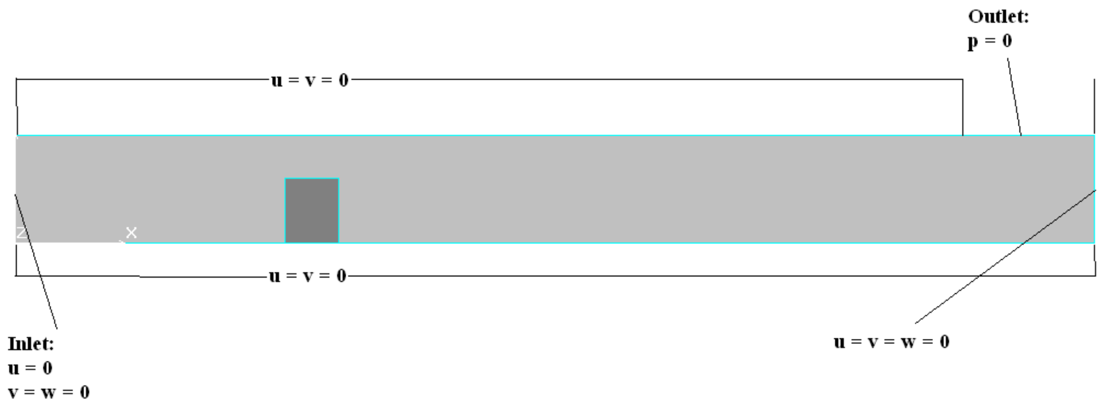


**Figure 5.7.3a:** 2-D schematic representation of the boundary condition in the xy plane (test case 4).

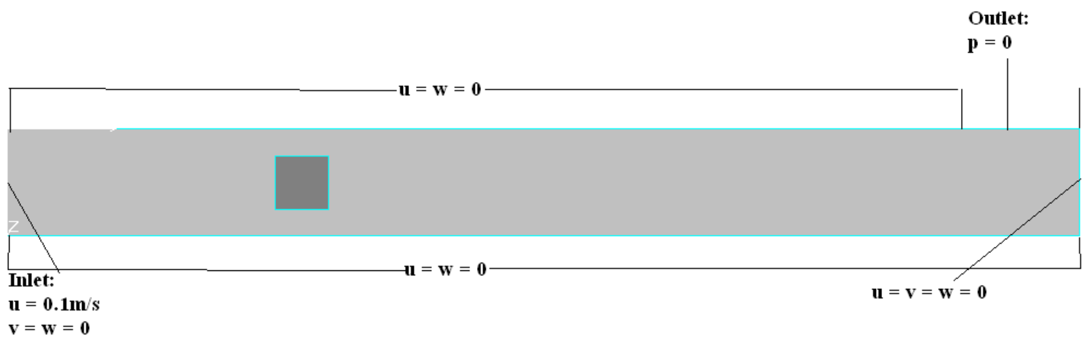


**Figure 5.7.3b:** 2-D schematic representation of the boundary condition in the xz Plane (test case 4).

## 5.7.2 Results

When the outlet is placed at the top end of the geometry, the pressure gradient obtained vary from 0 imposed at the outlet to 28KPa (figures 5.7.4 and 5.7.5) with the pressure gradient decreasing in the direction of the flow. The developed pressure profile can be justified with the accompanying flow field expressed by the velocity vectors in figures 5.7.6 through 5.7.8 from which it can be noted that the fluid moves with a minimum velocity of 0.02 $ms^{-1}$ and a maximum velocity of 0.26$ms^{-1}$. The computed mass balance across the computational domain shows that 96.05% of the fluid entering the domain exit, that is an error of 3.95% between the mass inflow and mass outflow is recorded.

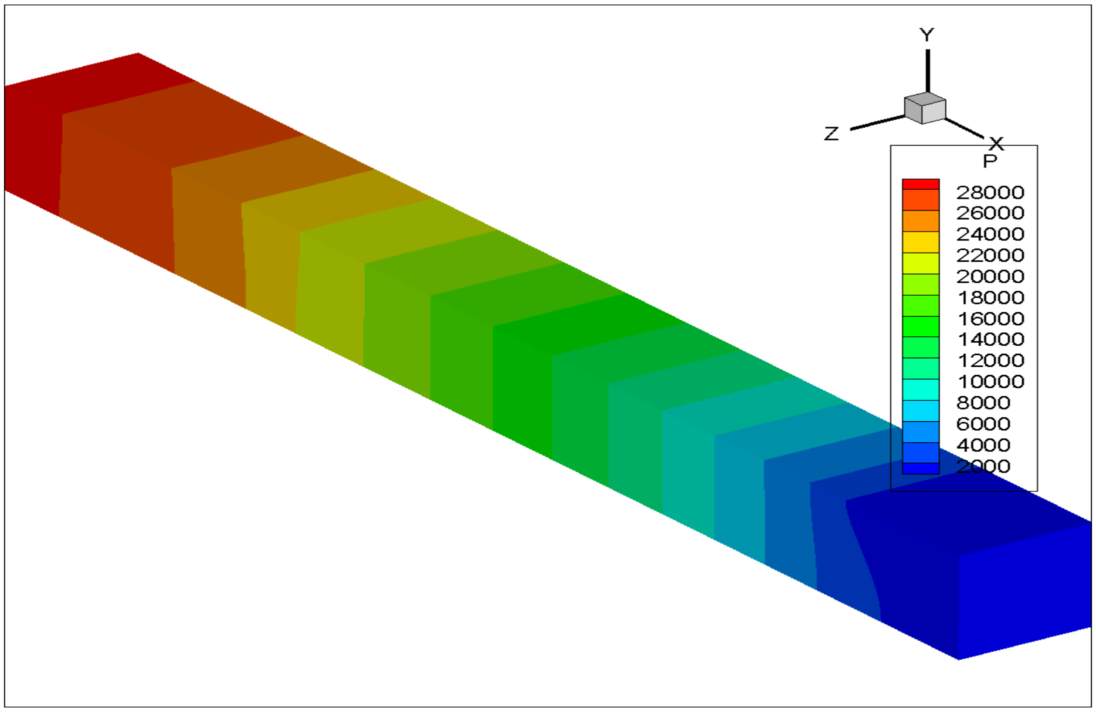


**Figure 5.7.4:** Pressure distribution (test case 4).

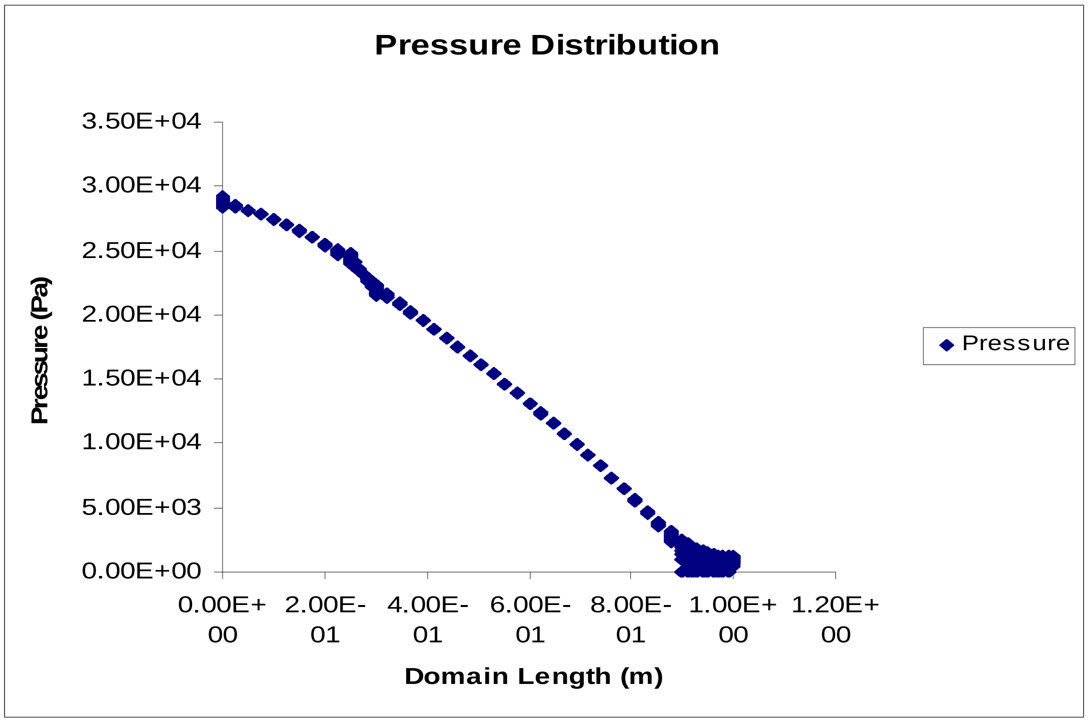


**Figure 5.7.5:** Graph of pressure distribution across the domain (test case 4).

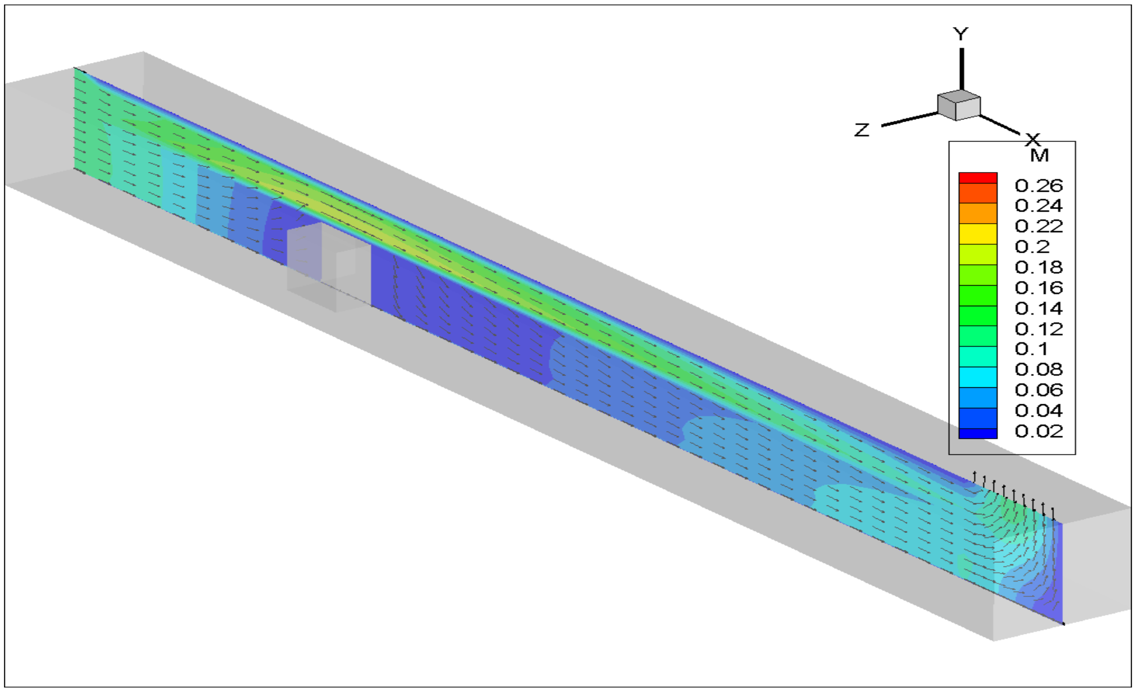
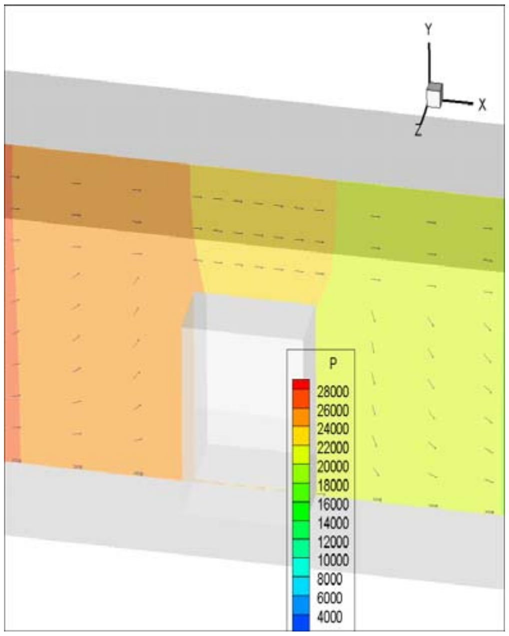


**Figure 5.7.6:** Vector plot profile coloured by the velocity magnitude contour in the z = 0.05m plane (test case 4).



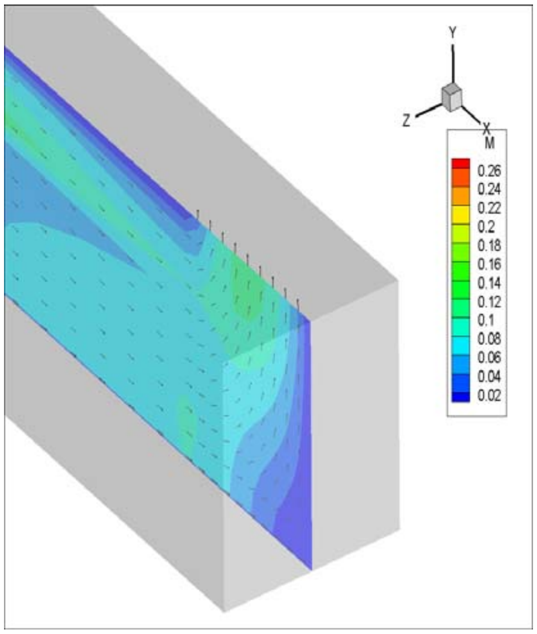


**Figure 5.7.7:** Vector profile coloured by pressure contour and zoomed around the obstruction in the z = 0.05m plane. test case 4.

**Figure 5.7.8:** Vector profile coloured by the velocity magnitude contour in the z = 0.05m plane test case 4.

## 5.8 Test case 5: Flow in a duct past a big square obstacle (0.05 × 0.075 × 0.05m) with an outlet placed at the bottom end of the domain.

### 5.8.1 Computational domain and boundary conditions

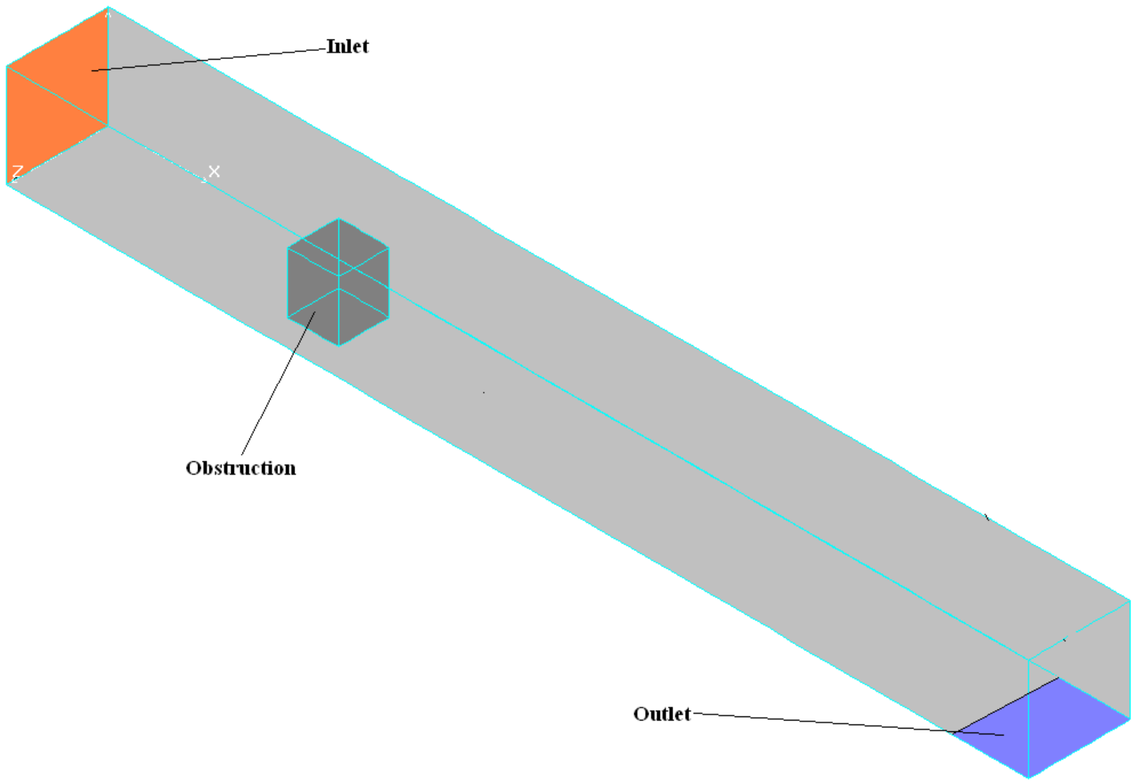


**Figure 5.8.1:** Computational domain for test case 5.

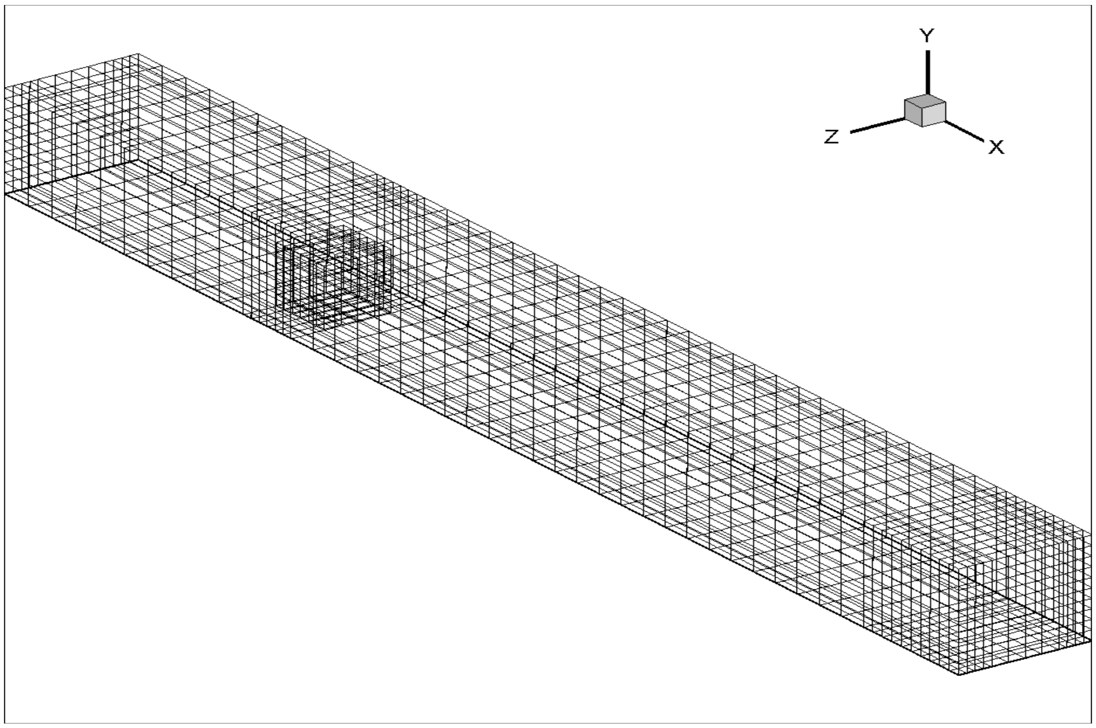


**Figure 5.8.2:** Finite element mesh for test case 5.

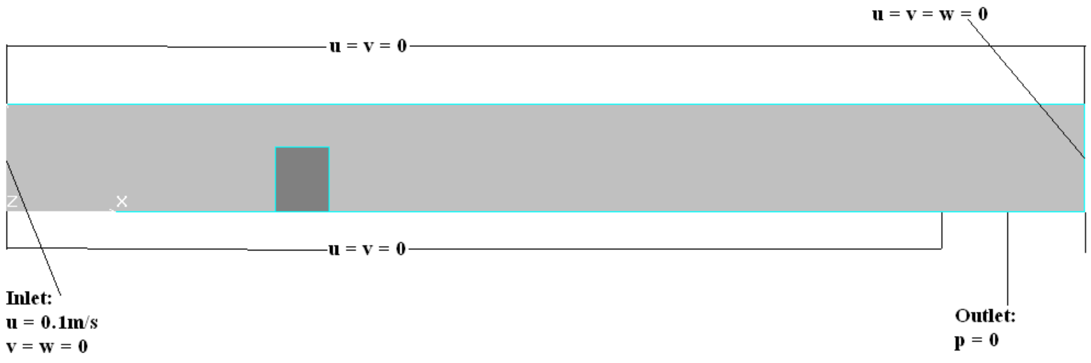


**Figure 5.8.3a:** 2-D schematic representation of the boundary condition in the xy plane (test case 5).

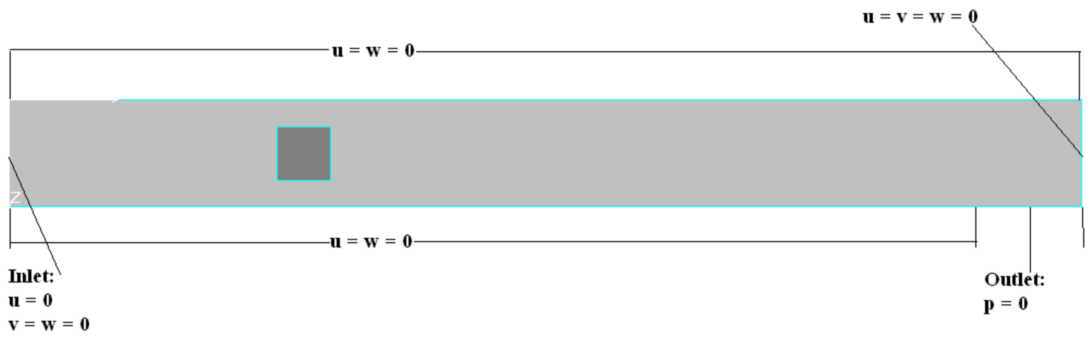


**Figure 5.8.3b:** 2-D schematic representation of the boundary condition in the xz Plane (test case 5).

## 5.8.2 Results

The results obtained (figure 5.8.4 through 5.8.6) for this case with the outlet placed at the bottom end of the geometry show a similar figure in term the pressure gradient and velocity magnitude with the results obtained in the previous test case where the outlet was placed at the top end of the geometry. The only difference for the present case is that the average velocity at the outlet is found to be 0.0958 $ms^{-1}$ giving a mass outflow of 0.958 kg $s^{-1}$ and hence the discrepancy between the inlet and the outlet masses gave an error of 4.2%.

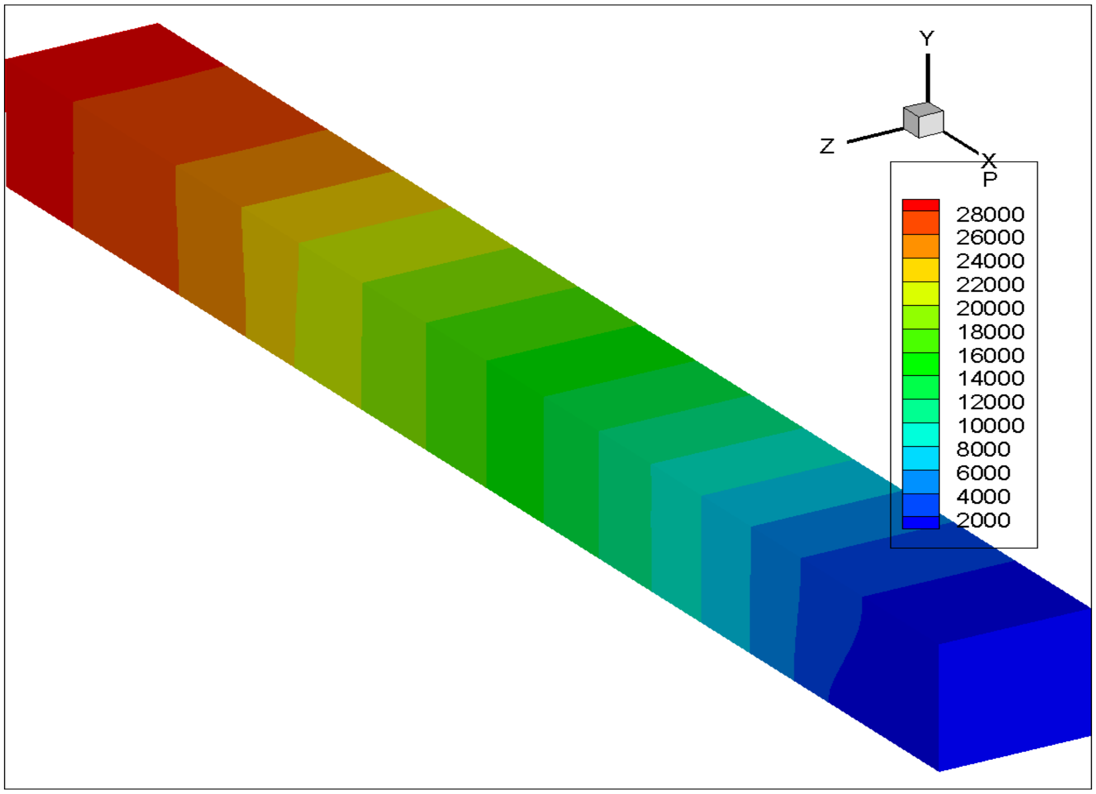


**Figure 5.8.4:** Pressure distribution (test case 5).

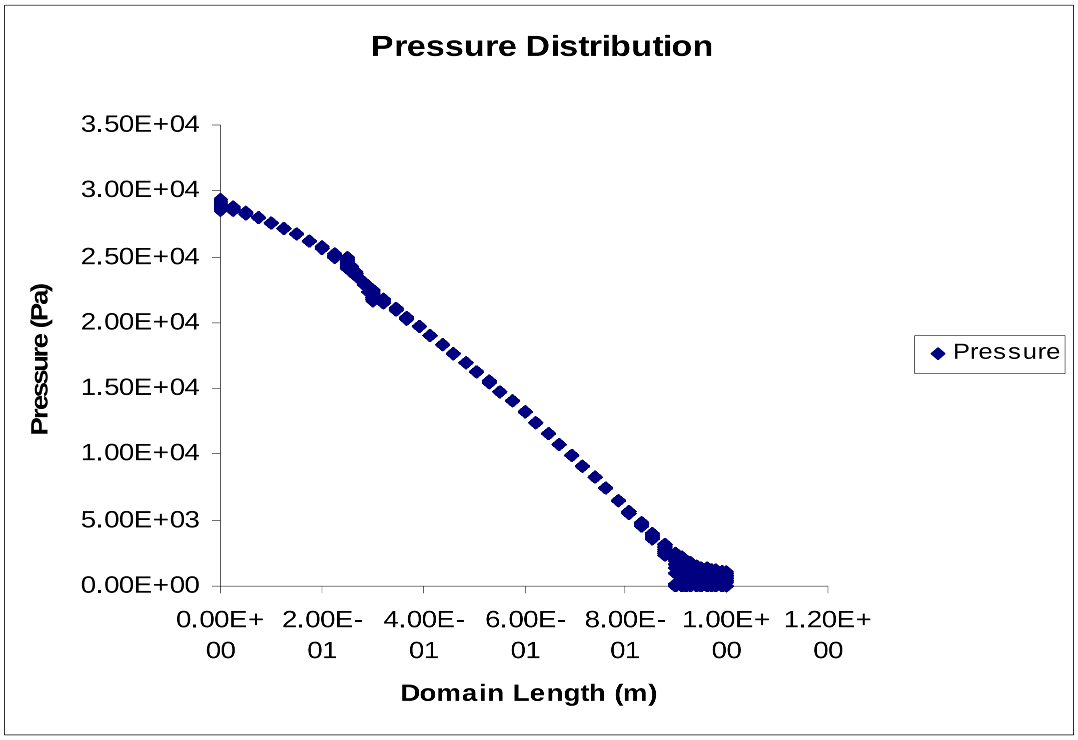


**Figure: 5.8.5:** Graph of pressure distribution across the domain (test case 5).

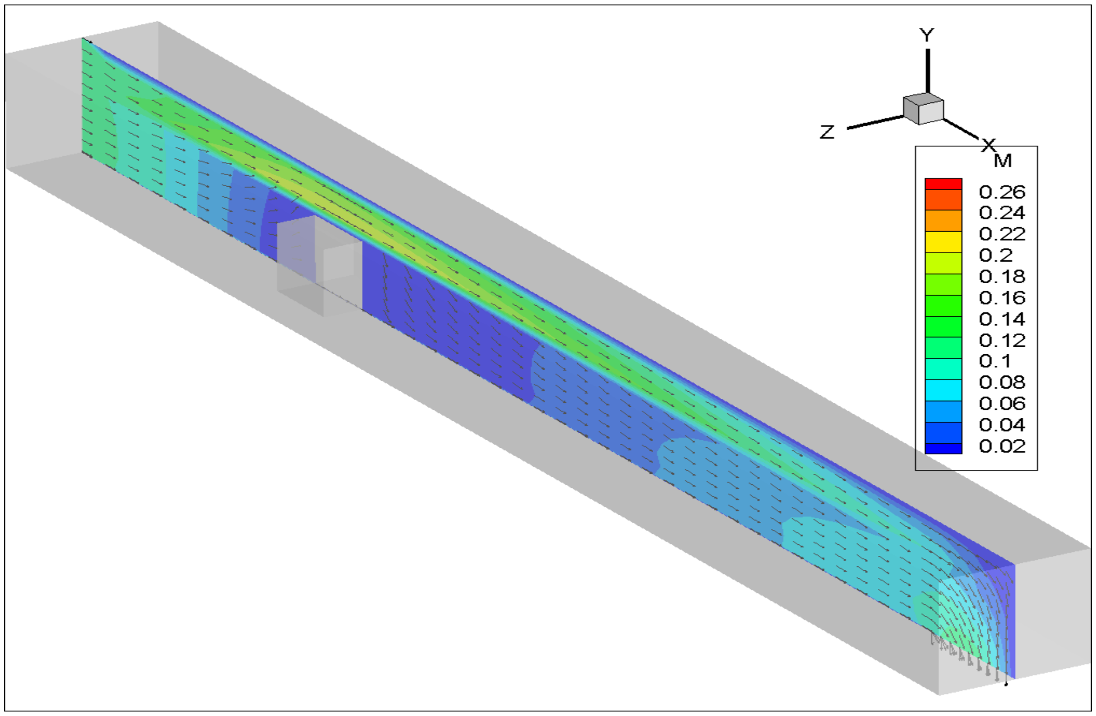


**Figure 5.8.6:** Vector plot profile coloured by the velocity magnitude contour in the z = 0.5m plane (test case 5).

## 5.9 Test case 6: Flow in a duct past a circular cylinder.

### 5.9.1 Computational domain and boundary conditions

The simulation for the sixth test case is carried out on a rectangular domain (figure 5.9.1) of same dimension than for the previous simulations, the domain is discretized using 8 nodes isoparametric hexahedral element into a finite element mesh consisting of 9062 nodes and 7560 elements (figure 5.9.2). But the obstacle in this case has a cylindrical shape. The imposed boundary conditions are given by figures 5.9.3a, b.

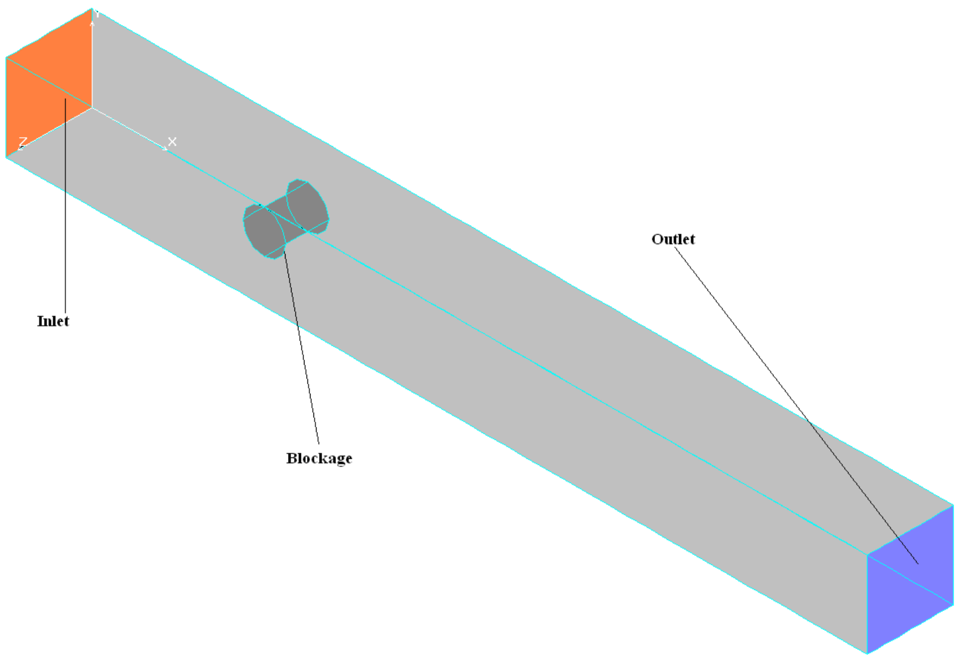


**Figure 5.9.1:** Computational domain for the case with a cylindrical blockage (test case 6).

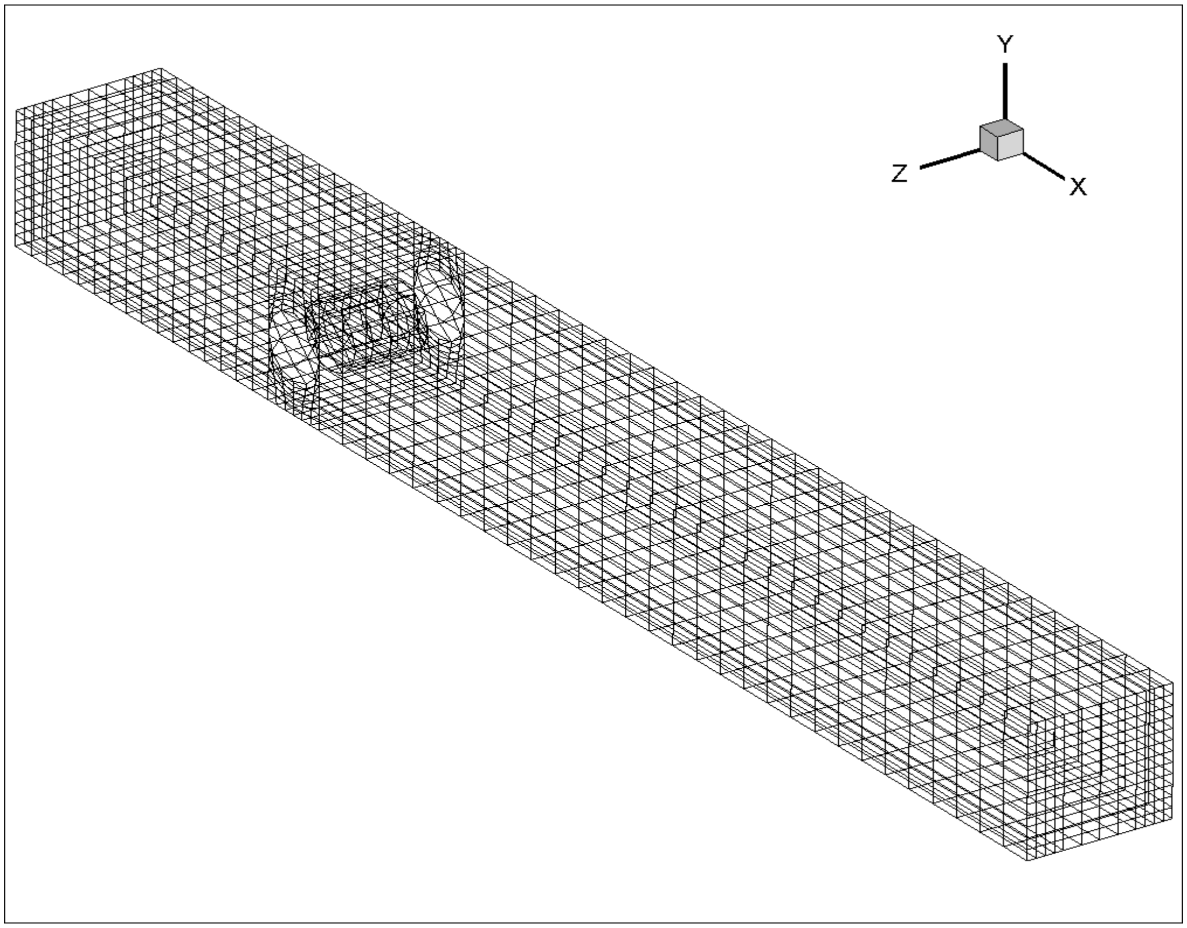


**Figure 5.9.2:** Finite element for test case 6.

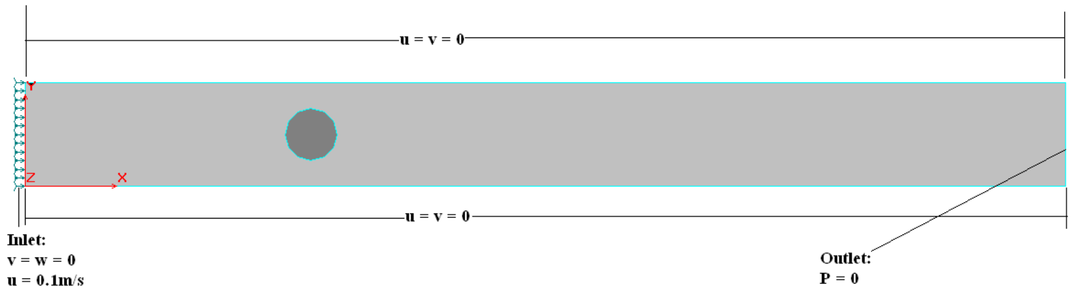


**Figure 5.9.3a:** 2-D schematic representation of the boundary condition in the xy Plane (test case 6).

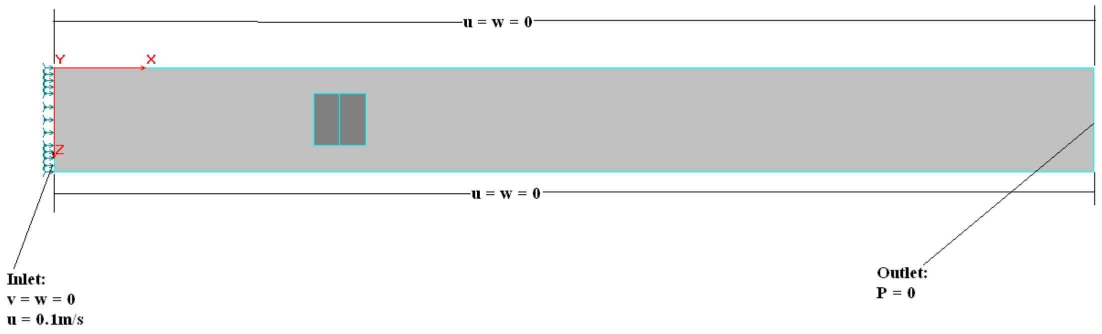


**Figure 5.9.3b:** 2-D schematic representation of the boundary condition in the xz Plane (test case 6).

### 5.9.2 Results

The results obtained with the case of cylindrical shaped obstacle show that the pressure decrease in the direction of the flow as in previous cases but with values from high pressure of about 45KPa to 0 imposed at the exit of the domain (figure 5.9.4 and 5.9.5). The velocity vectors plots given by figures 5.9.6 through 5.9.8 show that the flow remains fully developed throughout the length of the computational domain as it progress.

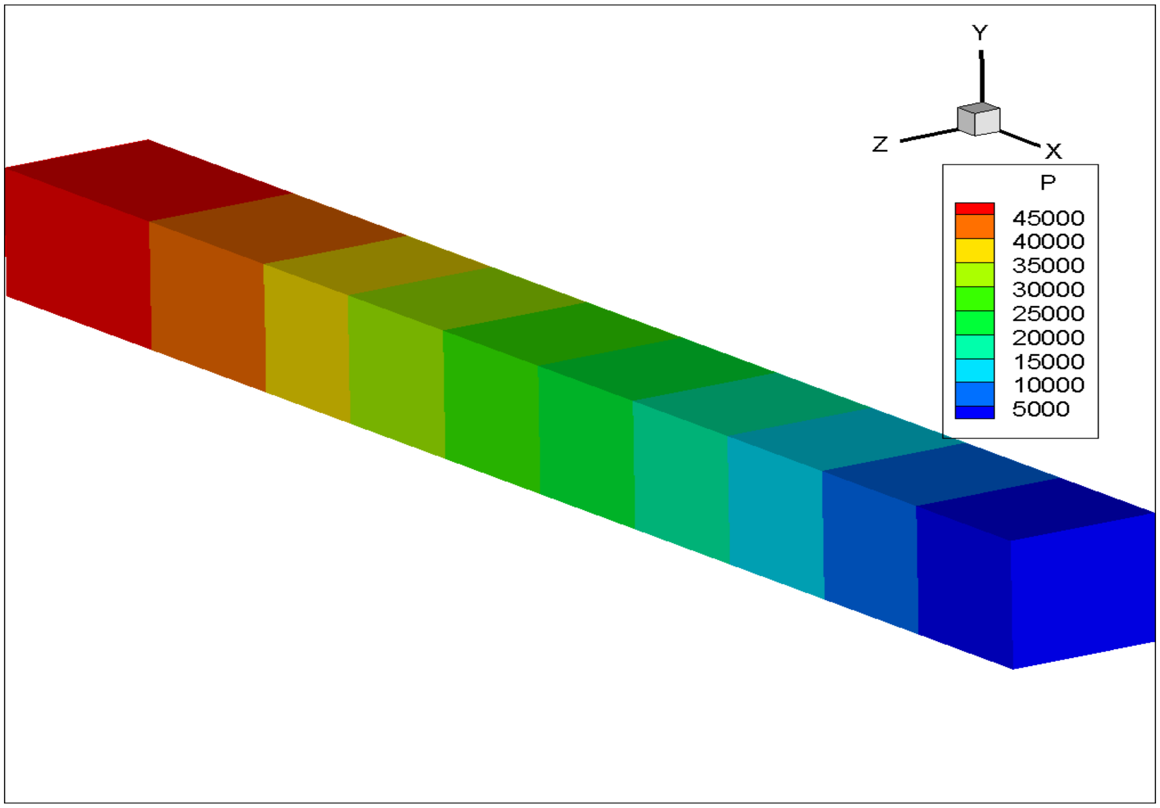


**Figure 5.9.4:** Pressure distribution (test case 6).

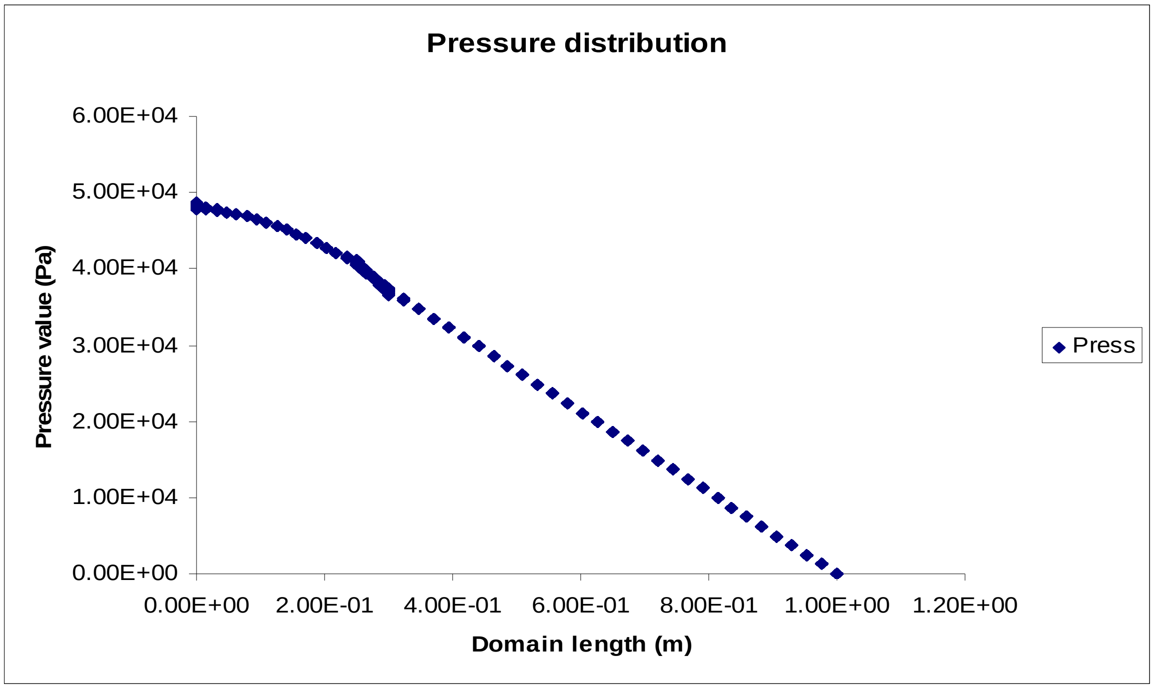


**Figure 5.9.5:** Graph of pressure distribution across the domain (test case 6).

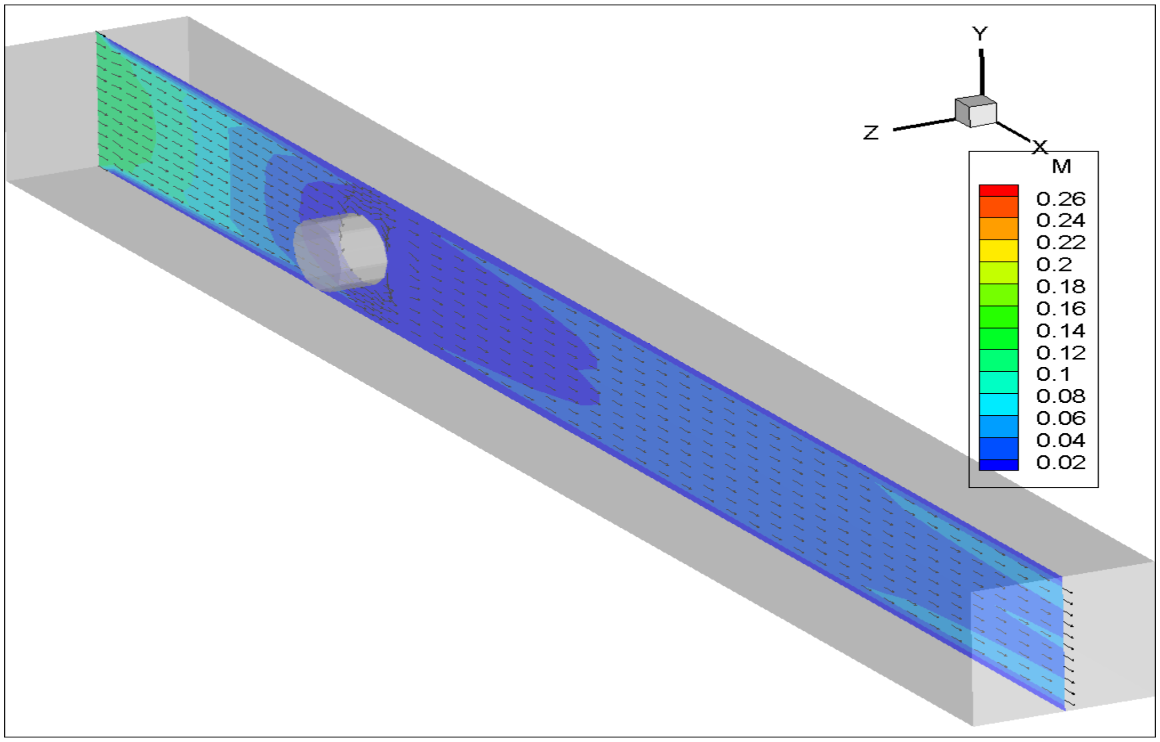


**Figure 5.9.6:** Vector plot profile coloured by the velocity magnitude contour in the z = 0.05m plane (test case 6).

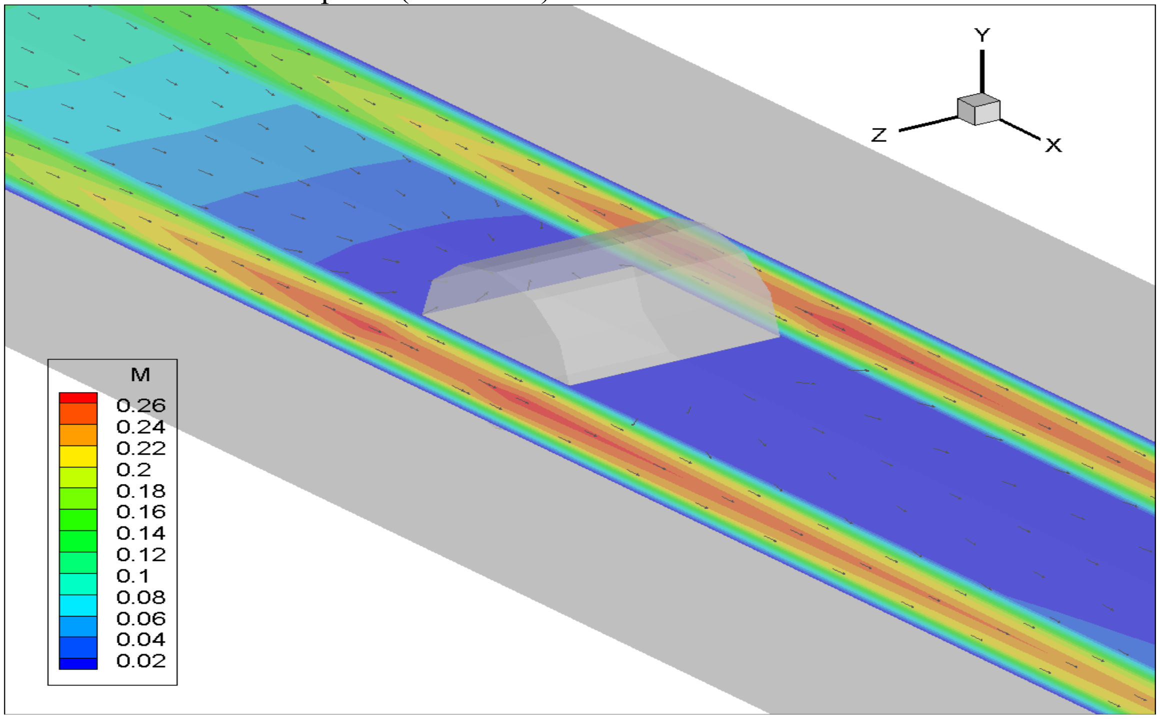


**Figure 5.9.7:** Velocity vector section plotted in the y = 0.05m plane (test case 6).

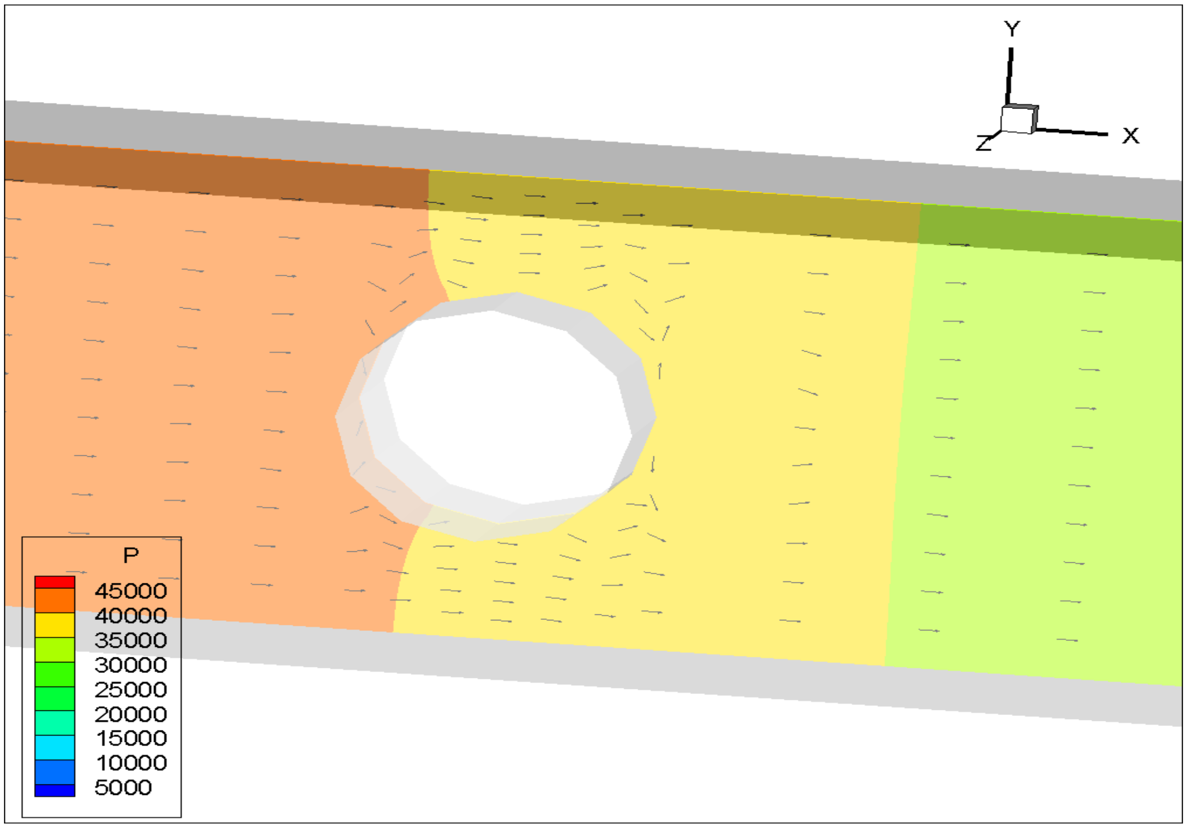


**Figure 5.9.8:** Velocity vector plotted in the z = 0.05m plane and coloured by pressure Contour (test case 6).

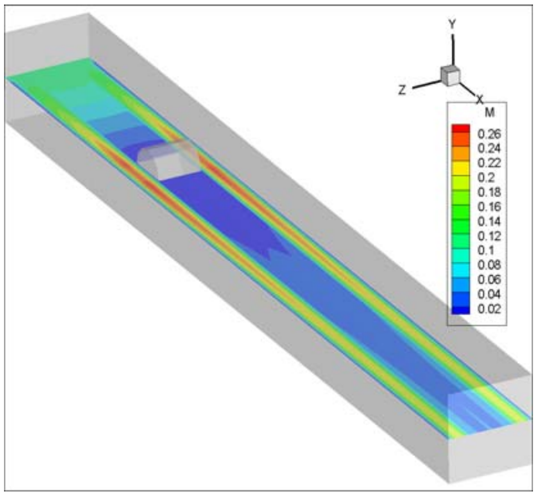


**Figure 5.9.9:** Profile of the contour of the velocity magnitude (y = 0.5m plane) test case 6.

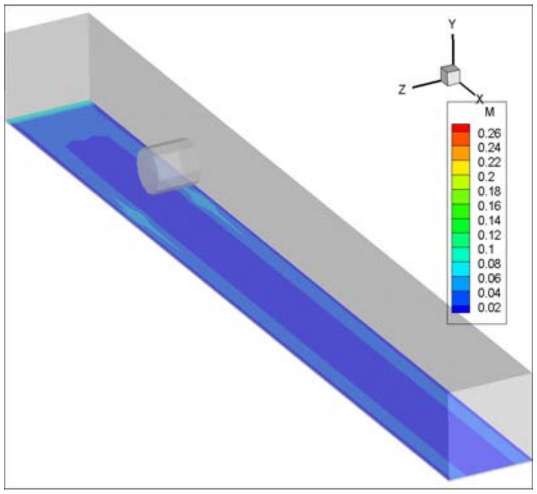


**Figure 5.9.10:** Profile of the contour of the velocity magnitude(y = 0.02m plane) test case 6.

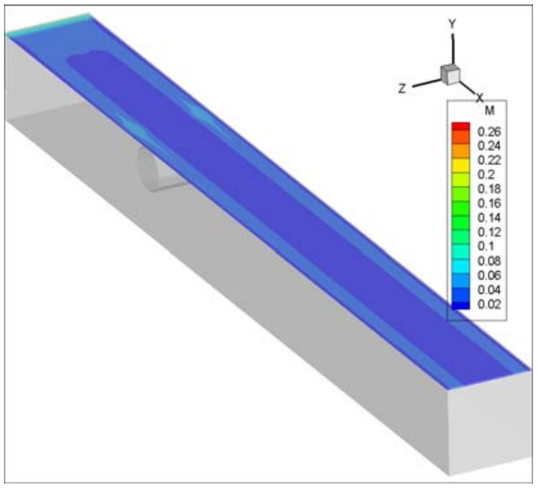


**Figure 5.9.11:** Profile of the contour of the velocity magnitude(y = 0.098m plane) test case 6.

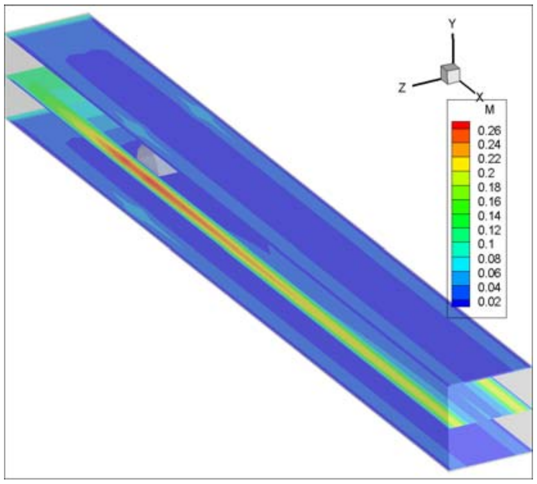


**Figure 5.9.12:** Combined profiles of the velocity magnitude (y = 0.02m, y = 0.05m, and y = 0.098m planes) test case 6.

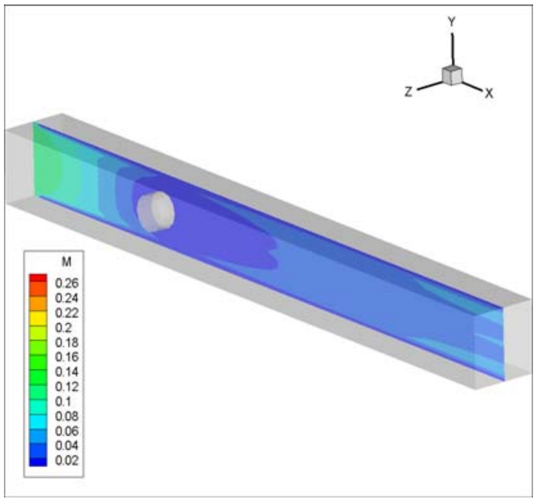


**Figure 5.9.13:** Profile of the contour of the velocity magnitude (z = 0.05m plane) test case 6.

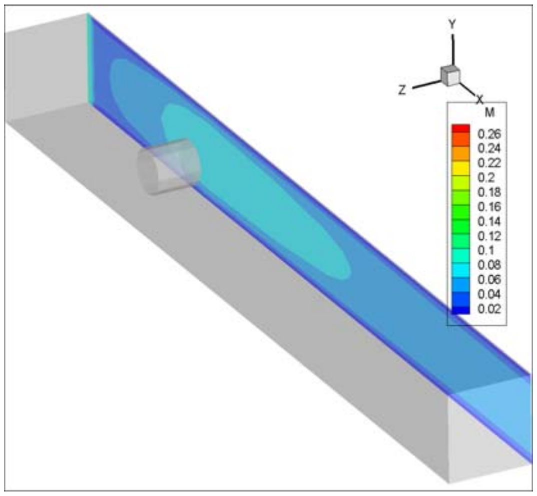


**Figure 5.9.14:** Profile of the contour of the velocity magnitude (z = 0.02m plane) test case 6.

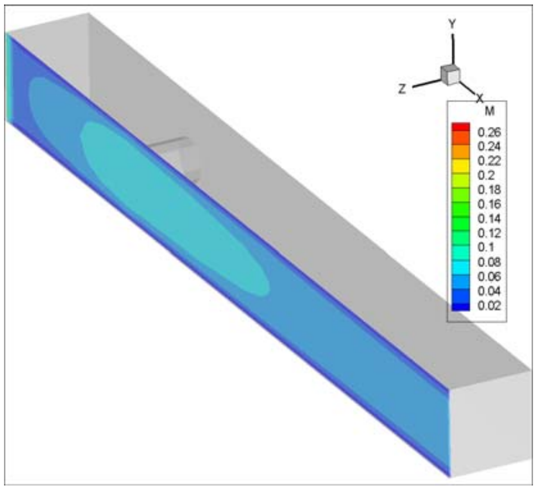


**Figure 5.9.15:** Profile of the contour of the velocity magnitude (z = 0.098m plane) test case 6.

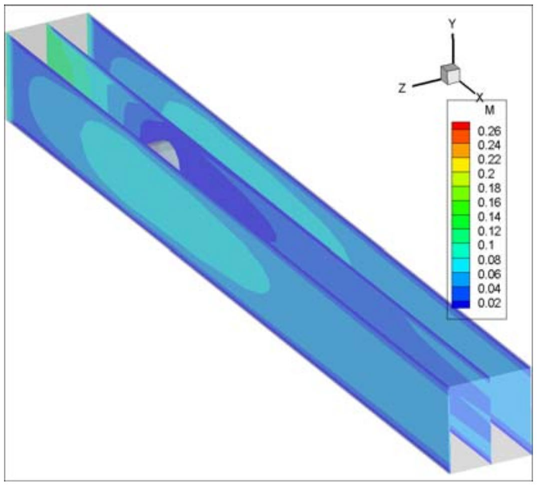


**Figure 5.9.16:** Combined profiles of the the velocity magnitude (z = 0.02m, z = 0.05m, and z = 0.098m planes) test case 6.

## 5.10 Test case 7: Flow in a duct past two cylindrical and one rectangular obstacles.

### 5.10.1 Computational domain and boundary conditions

For this final simulation case, three problems are solved in order to investigate the flow fields unknowns (velocity and pressure) using three different values of the power law parameter n. The three case are as follow; n = 0.87 (for shear thinning fluids), n = 1 (for purely Newtonian fluids), and n = 1.23 (for shear thickening fluids).

The domain in this case consists of a rectangular box with a similar dimension as for the previous problems solved, but the obstructions here consist of a combination of two cylindrical and one square blockages. The cylinders volumes are $4.91 \times 10^{-5}\ \text{m}^3$ each, the length, width, and height of the square are 0.025m as shown in figure 6.8.1, and the entire domain was discretized in a finite element mesh of 8834 nodes, and 7389 elements (figure 6.8.2). The imposed boundary conditions for this case are given by figures 6.8.3 and 6.8.4 respectively.

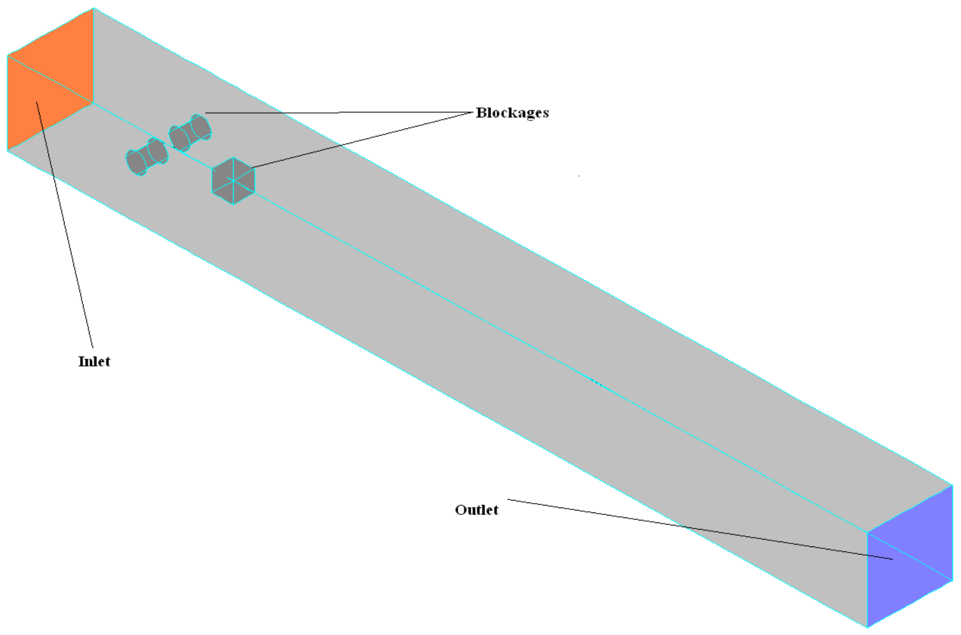


**Figure 5.10.1:** Computational domain for test case 7.

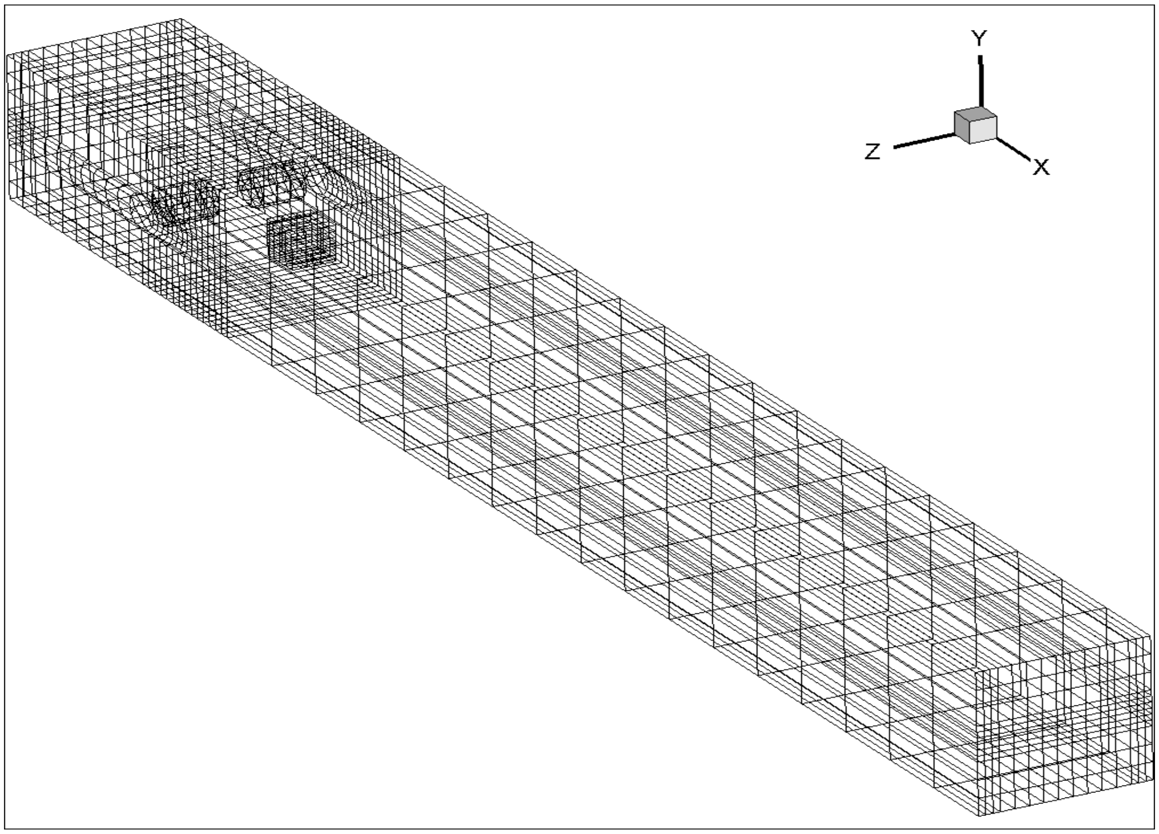


**Figure 5.10.2:** Finite element mesh for test case 7.

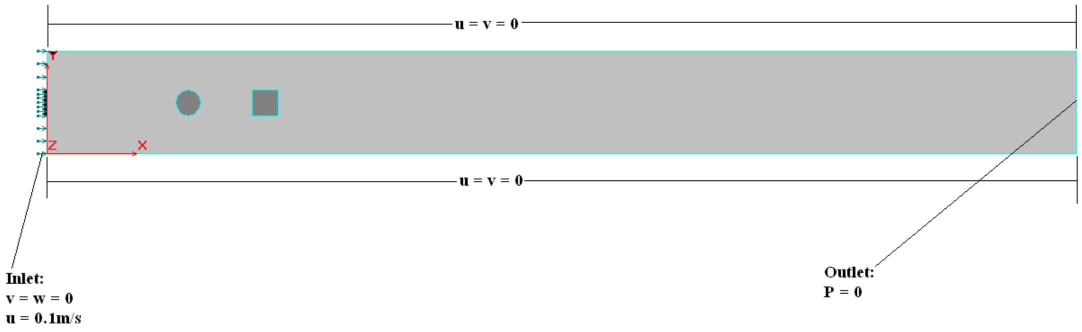


**Figure 5.10.3a:** 2-D schematic representation of the boundary condition in the xy plane (test case 7).

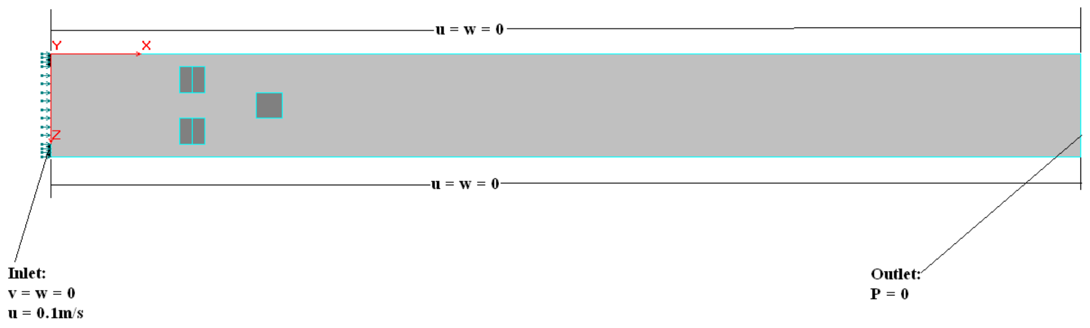


**Figure 5.10.3b:** 2-D schematic representation of the boundary condition in the xz plane (test case 7).

The results obtained in term of pressure and velocity contours and vector are presented (figure 5.10.4 through figure 5.10.19) and discussed below

### 5.10.2 Results

Figures 5.10.4 through 5.10.7 show that when the power law index (n) is set equal to 0.87 (shear thinning fluids) the pressure across the entire computational domain decrease in the direction of the flow from a maximum value of about 26KPa to 0 initially set at the exit of the domain. The same figure for the case of purely Newtonian fluids (n = 1) show a decrease of 22KPa to 0 whereas when n = 1.23 (shear thickening fluids), the decrease in pressure values is only form 20KPa to 0. In other hand, the velocity profile given by figure 5.10.8 through figure 5.10.19 represent the velocity contour plots and vector for the shear thinning fluids, which have the lowest velocity of the three cases simulated. As it can seen, the fluids move with a velocity of between $0.01\,\mathrm{m.s^{-1}}$ and $0.19\,\mathrm{m.s^{-1}}$, the computation of the mass balance for this case showed that 92% of the fluid entering the domain exited this indicated there is an error of 8% between the mass inflow and mass outflow. When the power law index is set equal to 1 (purely Newtonian flow case) given by figures 5.10.12 through 5.10.15, the mass balance calculated showed that 94.3% of the fluid entering the domain exited, thus in this case the discrepancy between the inlet and the outlet mass is 5.7%. The velocity magnitude of this case varies from $0.02\,m.s^{-1}$ to 0.3 $m.s^{-1}$ which is higher than the same figure obtained in the case of shear thinning fluids. For the last simulation (shear thickening fluids), the velocity plots obtained (figures 5.10.16 through 5.10.19) showed that the fluid moves with slightly faster than in the two previous cases with a velocity magnitude varying from $0.02\,m.s^{-1}$ to $0.4\,m.s^{-1}$ with 95.5% of fluid entering the domain exiting hence the discrepancy between the inlet and the outlet masses gave an error of 4.5% only.

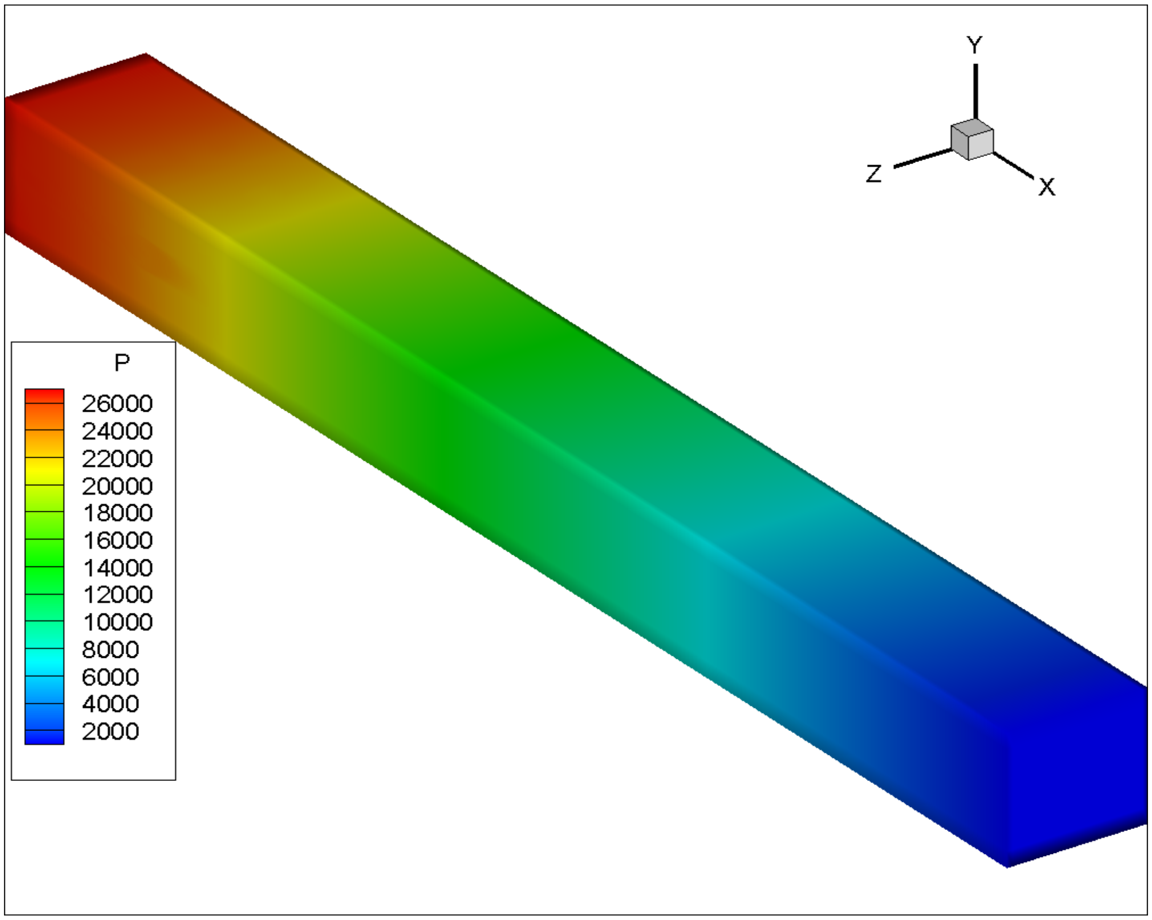


**Figure 5.10.4:** Pressure distribution (case n = 0.87) test case 7.

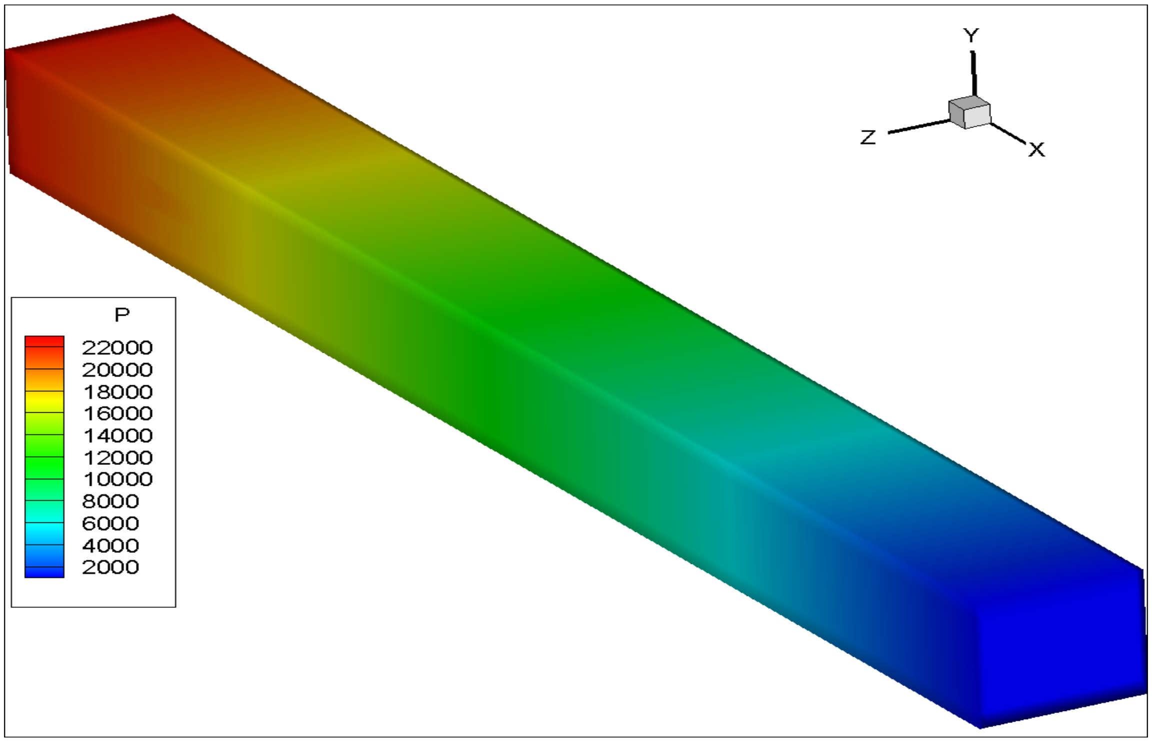


**Figure 5.10.5:** Pressure distribution (case n = 1) test case 7.

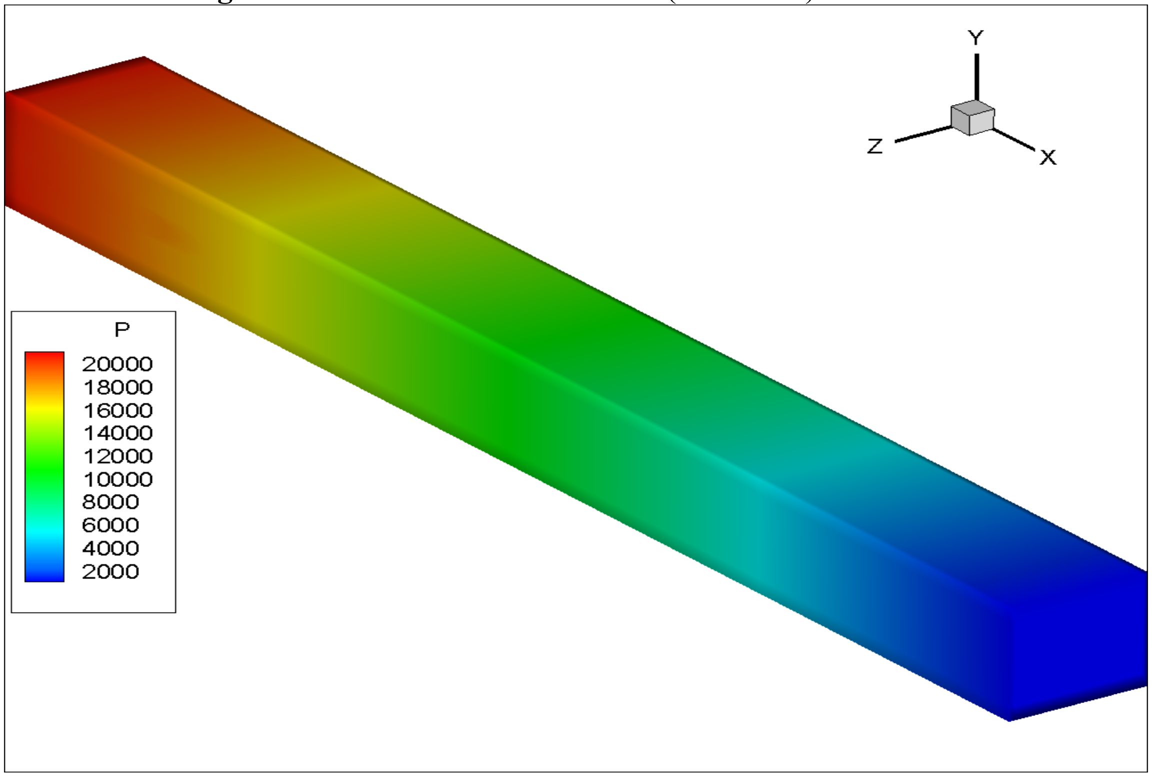


**Figure 5.10.6:** Pressure distribution (case n = 1.23) test case 7.

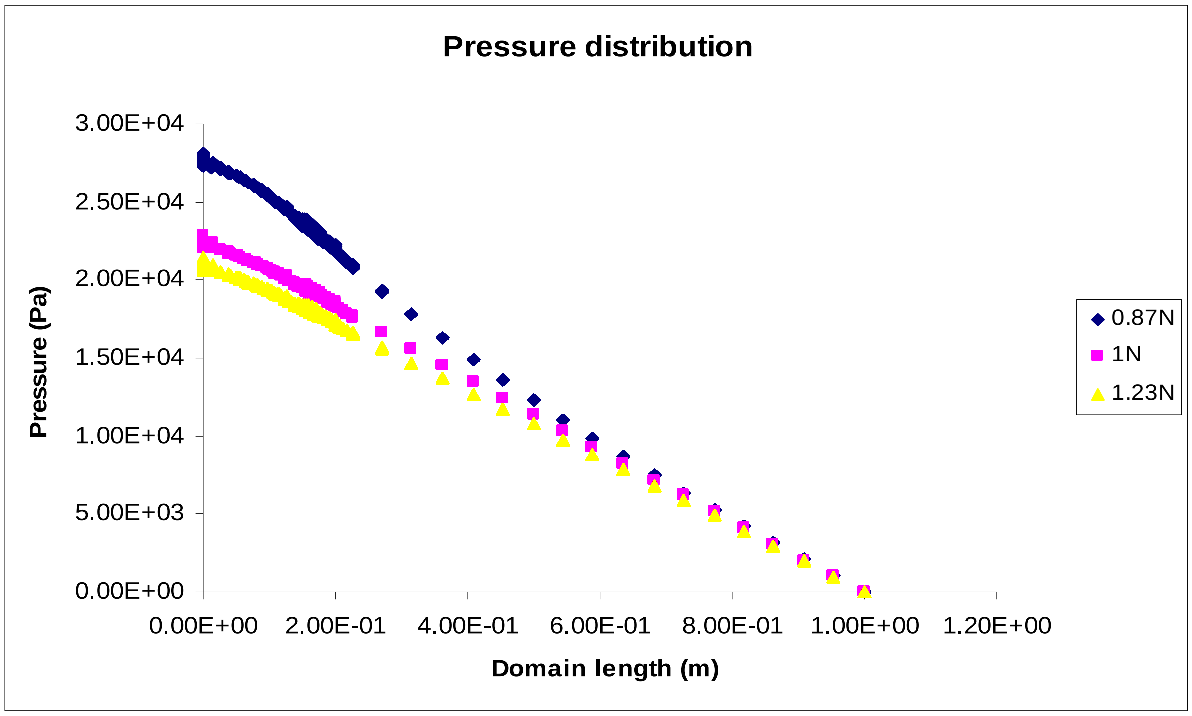


**Figure 5.10.7:** Graph of pressure distribution across the domain (cases n =0.87, n =1, n = 1.23) test case 7.

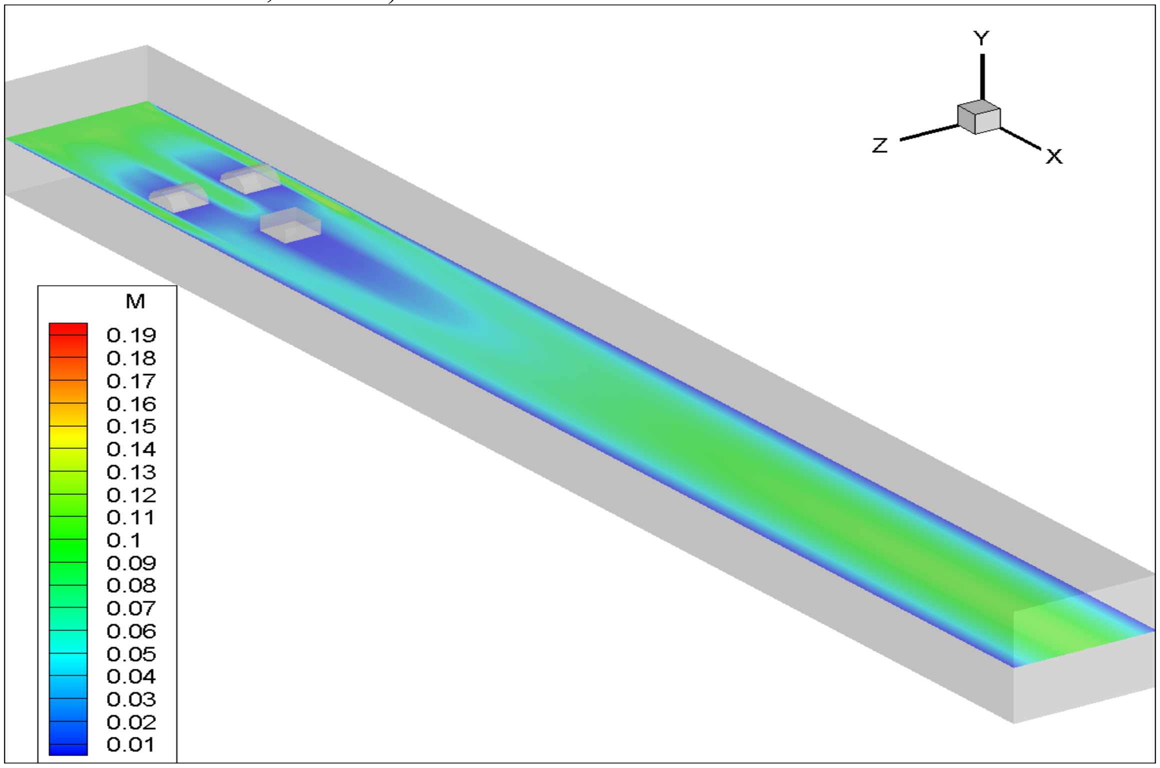


**Figure 5.10.8:** Velocity contour plot in the y = 0.05 plan (case n = 0.87) test case 7.

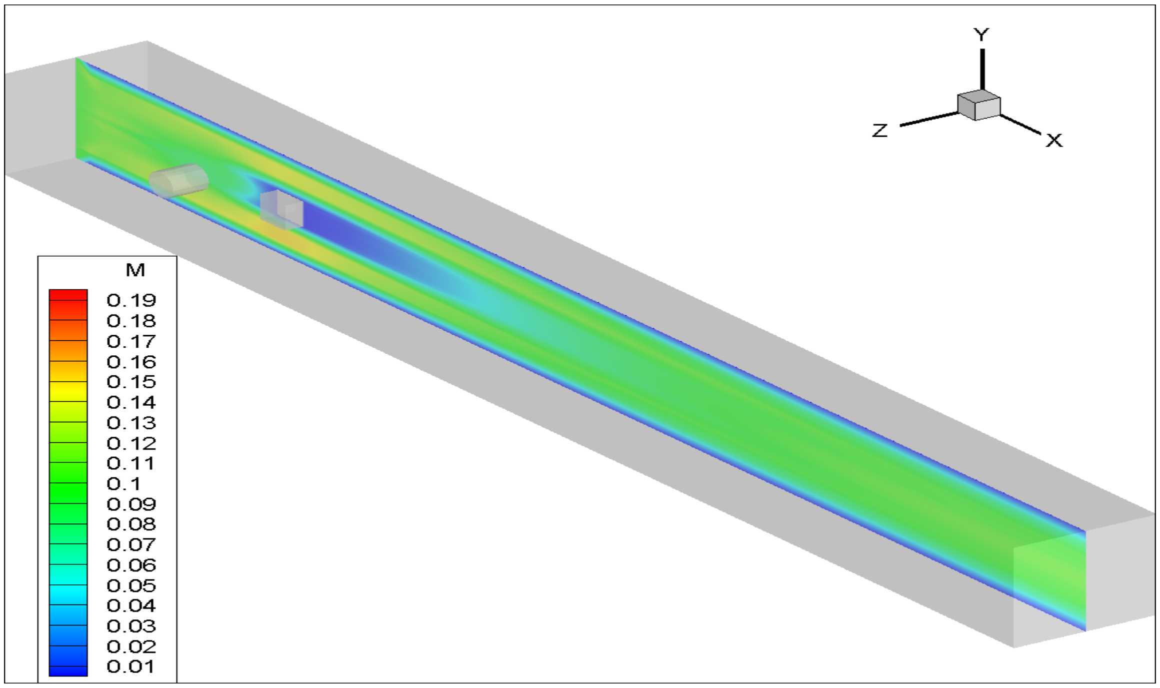


**Figure 5.10.9:** Velocity contour plot in the z = 0.05 plan (case n = 0.87) test case 7.

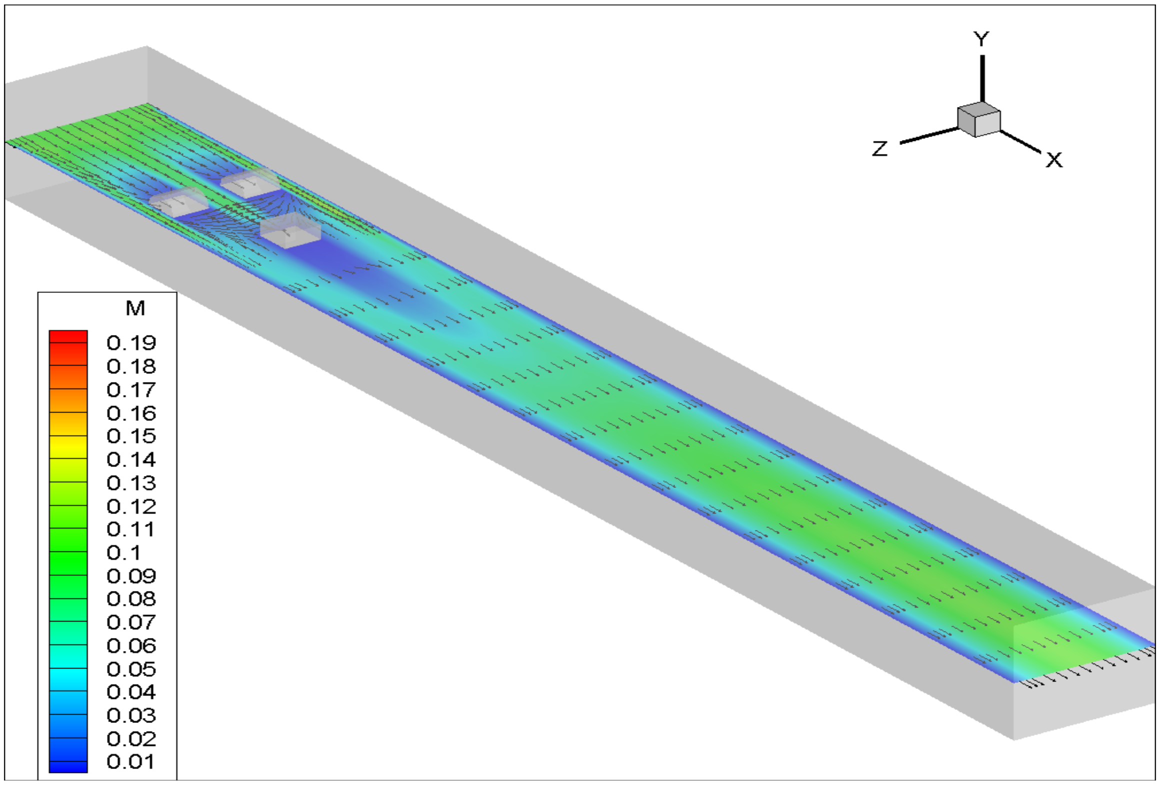


**Figure 5.10.10:** Velocity vector plot in the y = 0.05 plan (case n = 0.87) test case 7.

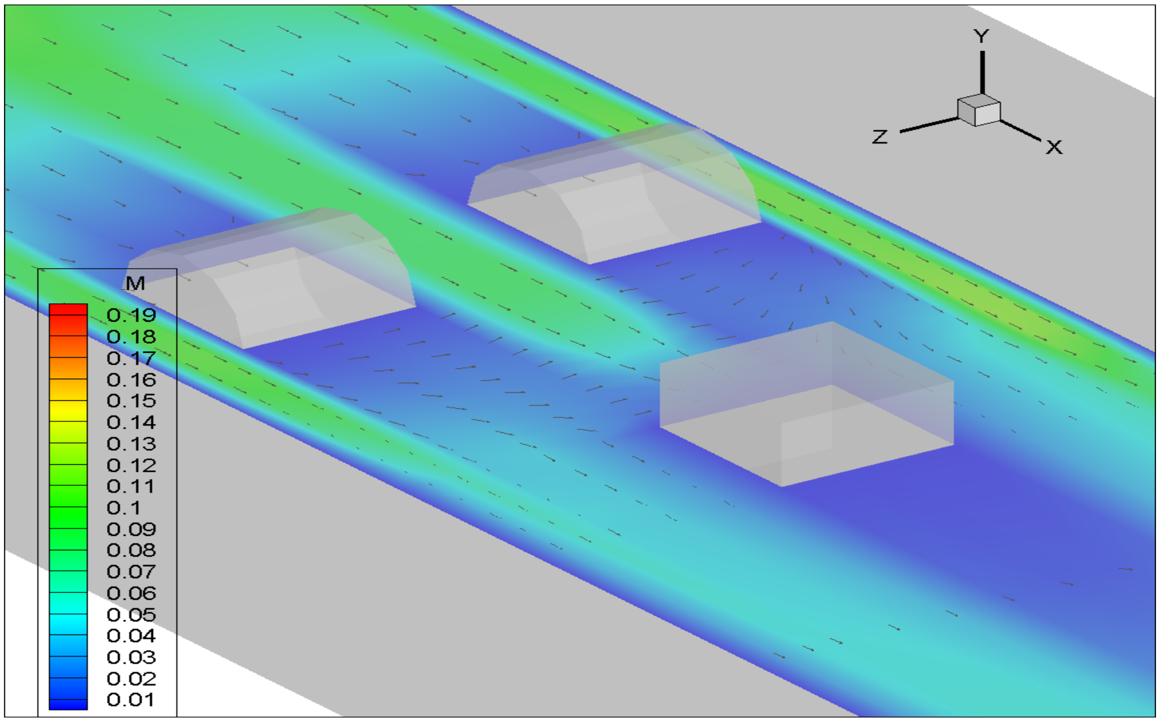


**Figure 5.10.11:** Velocity vector plot in the y = 0.05 plan zoomed around the obstacles (case n = 0.87) test case 7.

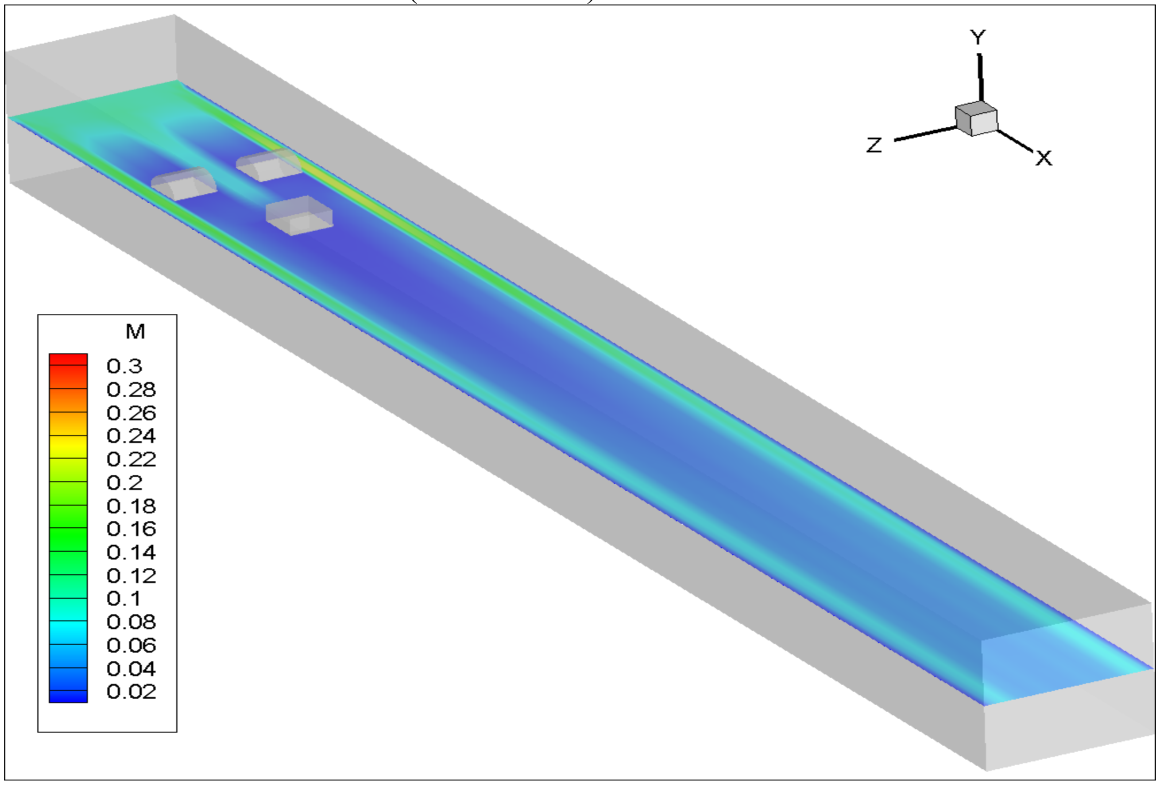


**Figure 5.10.12:** Velocity contour plot in the y = 0.05 plan (case n = 1) test case 7.

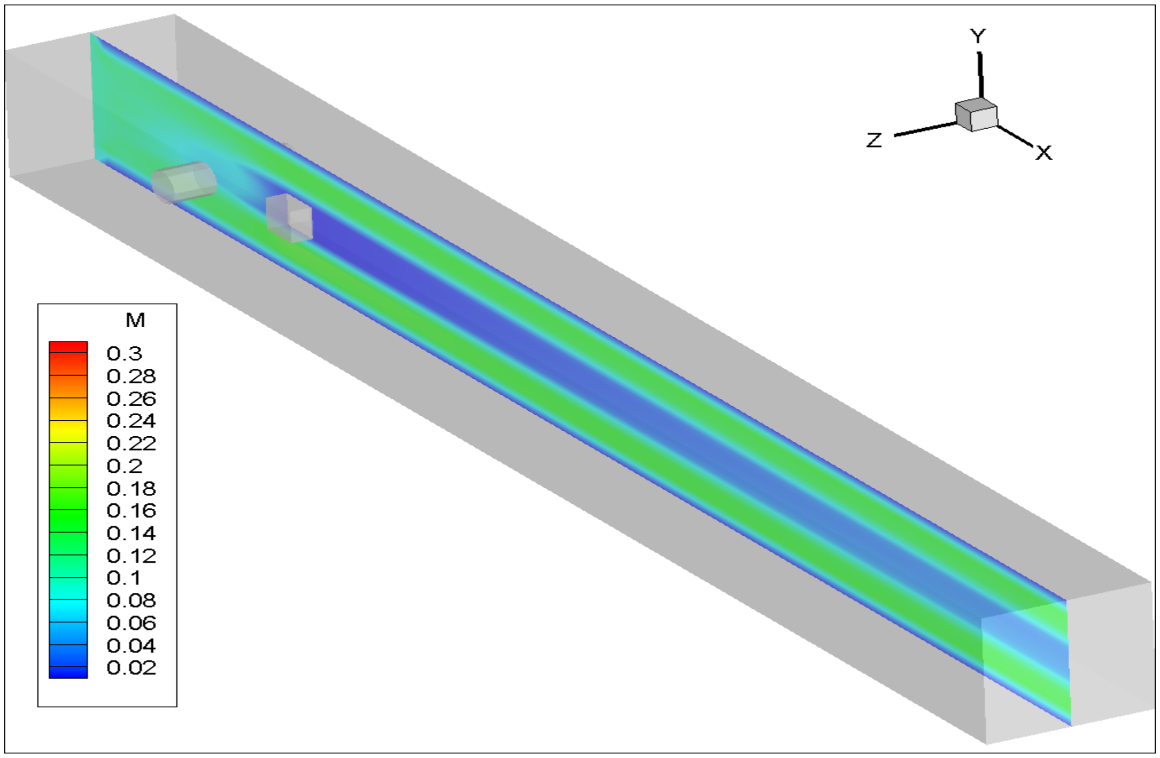


**Figure 5.10.13:** Velocity contour plot in the y = 0.05 plan (case n = 1) test case 7.

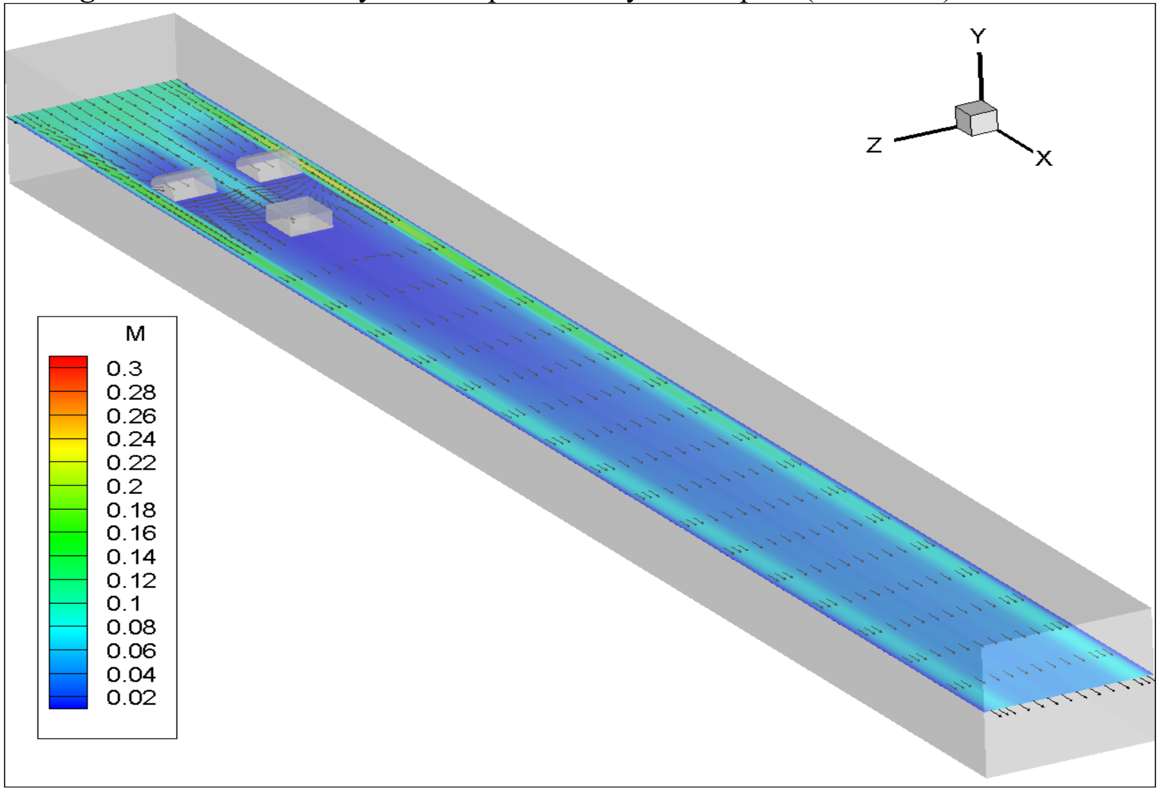


**Figure 5.10.14:** Velocity vector plot in the y = 0.05 plan (case n = 1) test case 7.

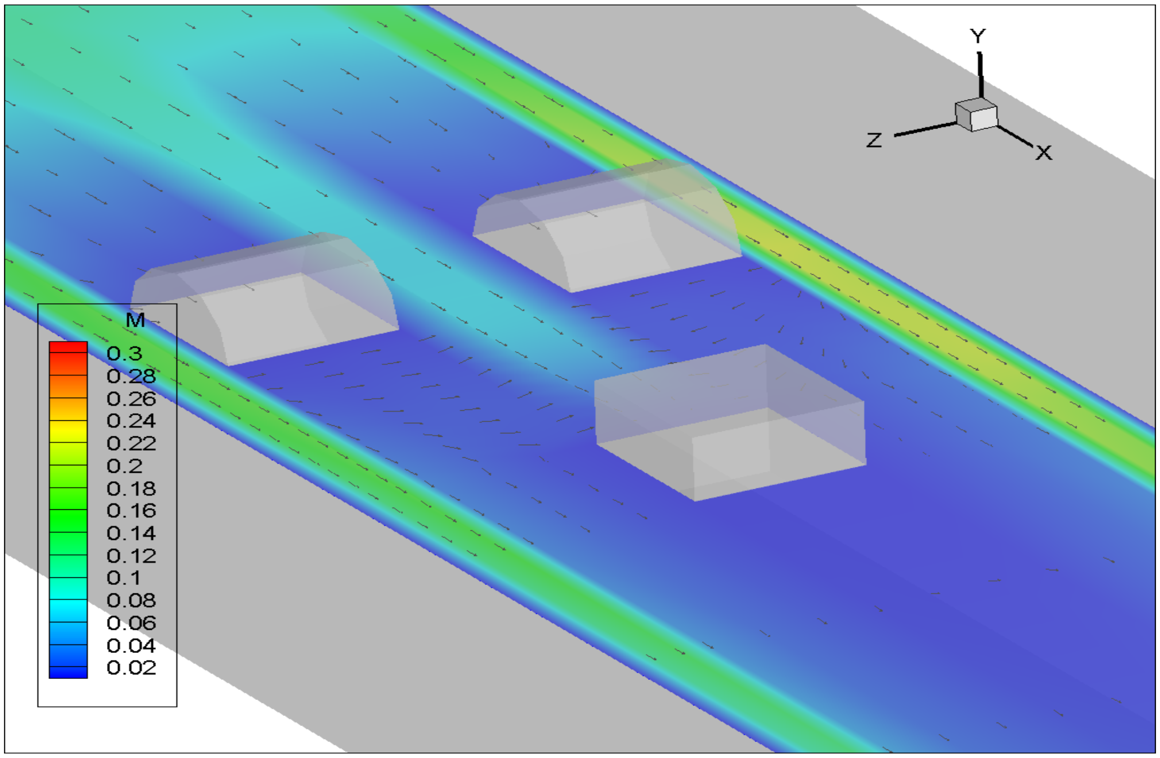


**Figure 5.10.15:** Velocity vector plot in the y = 0.05 plan zoomed around the obstacles (case n = 1) test case 7.

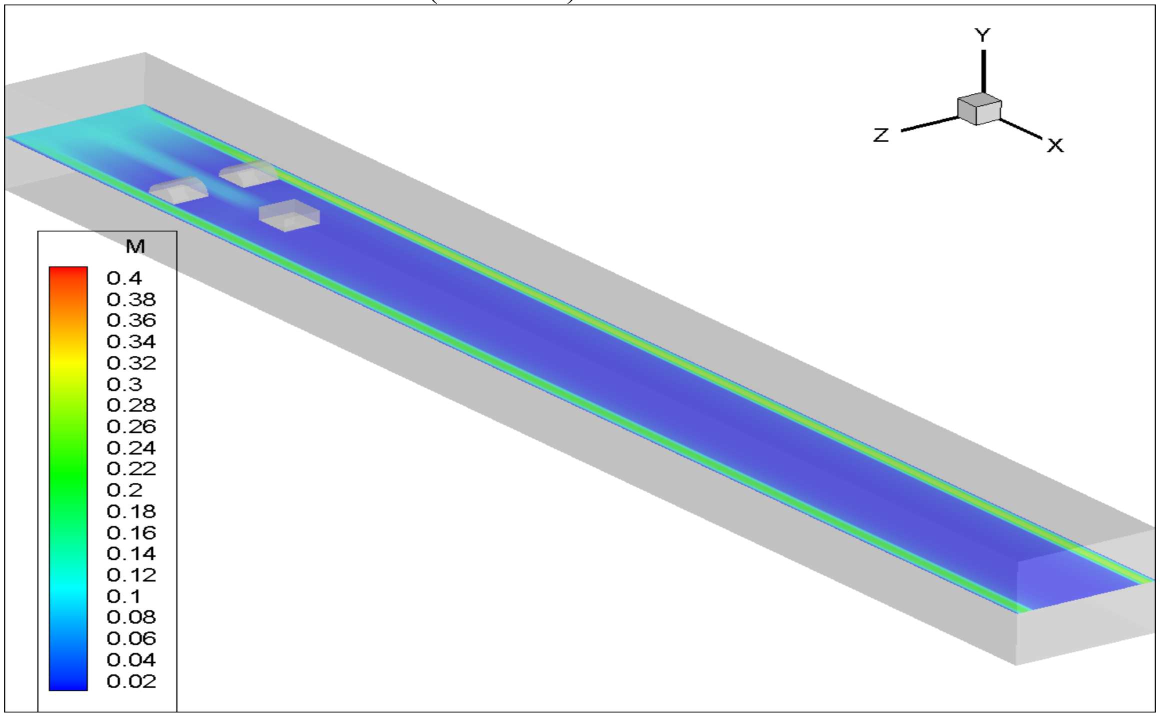


**Figure 5.10.16:** Velocity contour plot in the y = 0.05 plan (case n = 1.23) test case 7.

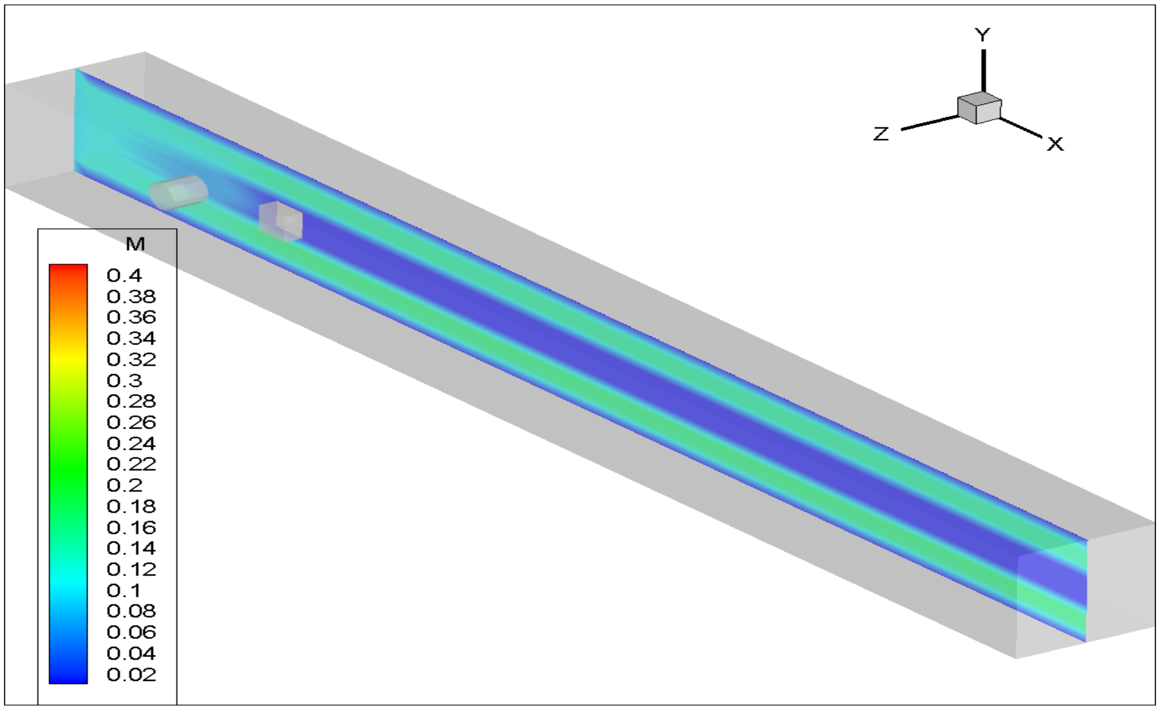


**Figure 5.10.17:** Velocity contour plot in the z = 0.05 plan (case n = 1.23) test case 7.

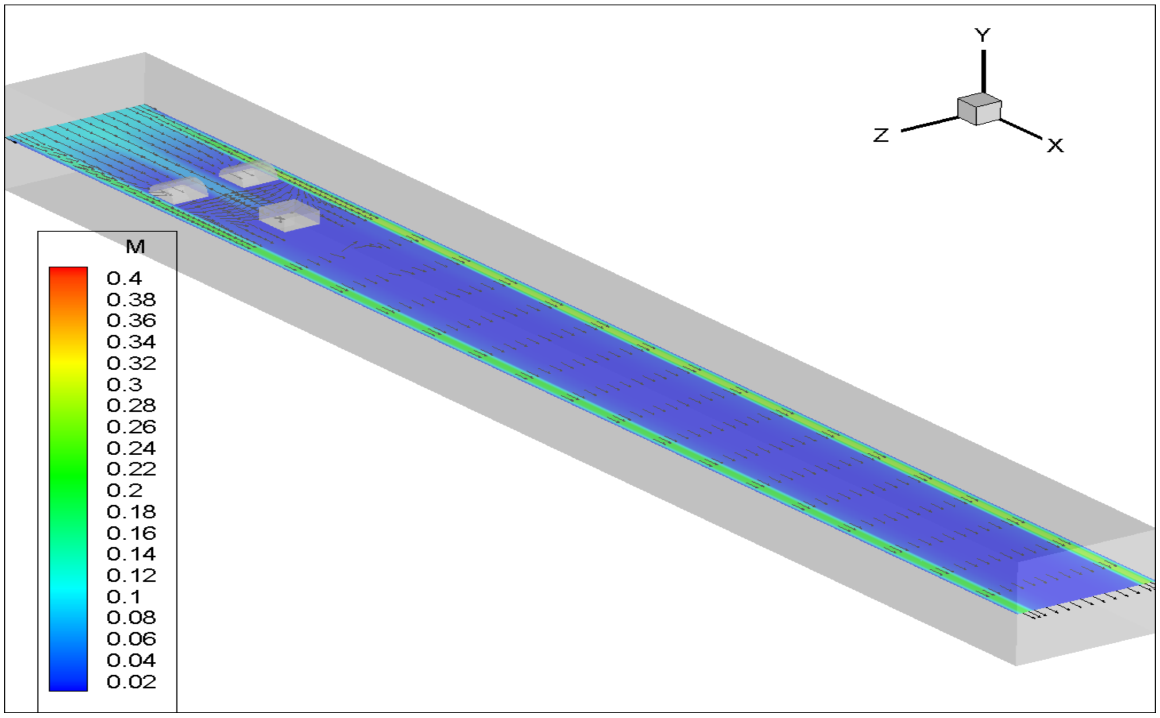


**Figure 5.10.18:** Velocity vector plot in the y = 0.05 plan (case n = 1.23) test case 7.

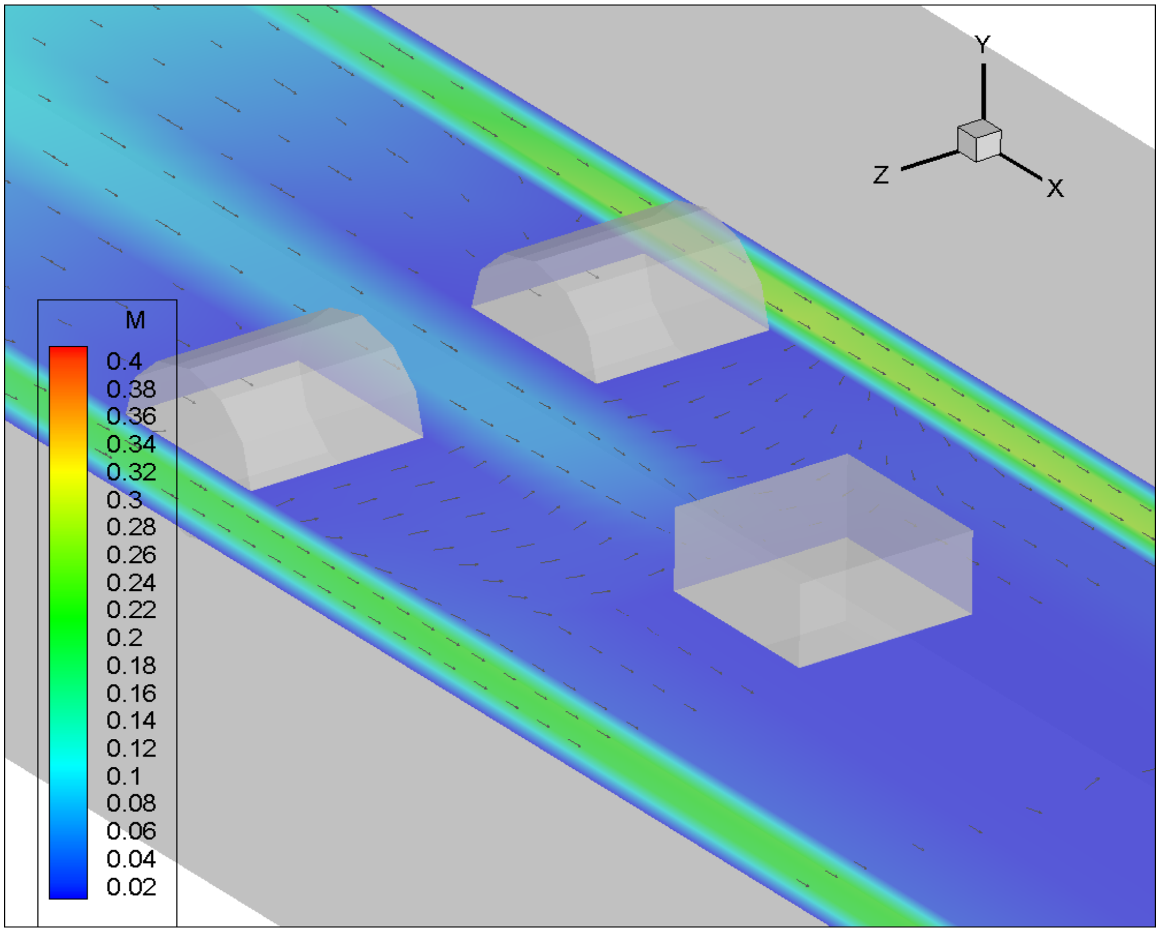


**Figure 5.10.19:** Velocity vector plot in the y = 0.05 plan zoomed around the obstacles (case n = 1.23) test case 7.

Large number of results discussed in this chapter all show self- consistency of the simulations obtained by the developed schemes. In addition where ever possible other evidence such as pattern of pressure drop or accuracy of the conservation of mass have been taken into account. Therefore the main conclusion of this chapter is the developed three dimensional finite element schemes can be used to solve realistic flow problems with minimum computational cost. Flexibility of an in-house developed scheme combined with the mathematical rigour and computational economy makes the outcome of this research a useful engineering tool.

# CHAPTER 6

# CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

## 6.1 Conclusions

Three dimensional finite element computer models for the solution of governing equations of generalized Newtonian fluids have been developed and used to simulate flow of power-law fluids through domains involving obstructions. These models are based on two different finite element schemes, namely the mixed velocity-pressure (UVWP) and a modification of the mixed velocity-pressure (UVWP) methods. Both models utilize 8-noded isoparametric $c^0$ continuous hexahedral elements to discretize velocity and pressure unknowns. Therefore, in contrast to the traditional mixed finite element schemes, the use of lower order of interpolation function for pressure is avoided and the required stability condition (i.e. the LBB condition) is satisfied, for the first time in three dimensional simulations, via the use of a perturbed continuity constraint. The discretizations of the temporal variables in both schemes are carried out using the first order Taylor-Galerkin scheme. The variation of the viscosity across the computational domain which is based on the power law model depends on the value of the power law index and the computer models can be used to simulate Newtonian fluid flow (power law index equal 1), shear thinning fluid (power law index less than 1) and shear thickening fluid (power law index greater than 1).

The accuracy and validity of the models have been evaluated by solving three benchmark problems and seven test cases.

Because there is no available experimental data to validate results obtained for the test cases problems, the validation processes of these cases have been based on the comparison of mass flow rate in and out of the computational domain, as well as the overall consistency and logical interpretation of the results. The comparisons done for all of the test cases results show that the differences between the mass flow from the outlets and the mass flow from the inlet were insignificant. These mass balance computations were backed up by the different plots representing flow and pressure drop patterns which show stable and theoretically expected forms. Therefore, the novel approach used here to simulate three dimensional flows via the use of equal order hexahedral discretization of pressure and velocity in conjunction with a perturbed continuity constraint has been justified. The main conclusions of this project can hence be summarised as:

1- Three dimensional finite element schemes for the simulation of incompressible regimes demonstrating non-linear rheological behaviour in complex domains which are computationally very efficient can be developed.
2- The use of perturbed continuity constraint in conjunction with an isoparametric tensor product element such as tri-linear 8 noded brick element results in a robust scheme which can readily satisfy LBB condition.
3- Although both mixed UVWP and Modified mixed UVWP generate stable accurate results the modified scheme converges much faster than the traditional approach.
4- Temporal discretization of the governing equations plays an important role in maintaining stability of the present schemes. Here first order Taylor –Galerkin approach has been used successfully. It is doubtful that the use of a simpler finite difference based techniques such as the *theta method* will work in three dimensional simulations.
5- The simulations presented here are obtained using a Pentium IV personal computer hence maintain maximum computing economy.

## 6.2 Recommendations for future work

There are many novel areas which using the schemes developed in this project can be further investigated. To provide some examples of such extensions the following examples can be considered.

**6.2.1** The code developed in this present study assumed isothermal regimes and hence omitted the energy equation from the system of equations representing incompressible fluid flow. However, in many engineering applications flow regimes are non-isothermal. Experience gained from two-dimensional flow simulations show that addition of a challenging. A very important extension will therefore be the inclusion of the energy equation in the discretization scheme.

**6.2.2** Another extension can be based on the application of the basic scheme with required modifications to model viscous flows carrying small amounts of solid particles. In these situations as the concentration is low the fundamental flow equations remain the same, however, need to be modified to include variable density. Variations of density can be tracked via the use of an equation of state which needs to be updated at the end of each time step.

**6.2.3** The governing equations can be put in the non-dimensional form by using the following non-dimensional variables.

$$x^* = \frac{x}{L}, \quad y^* = \frac{y}{L}, \quad z^* = \frac{z}{L}$$

$$u^* = \frac{u}{u_\infty}, \quad v^* = \frac{v}{u_\infty}, \quad w = \frac{w}{u_\infty}$$

$$t^* = \frac{t}{L/u_\infty}, \quad \rho^* = \frac{\rho}{\rho_\infty}, \quad \eta^* = \frac{\eta}{\eta_\infty}$$

$$p^* = \frac{p}{\rho_\infty u_\infty^2}$$

$$\mathrm{Re} = \frac{\rho_\infty u_\infty L}{\eta_\infty}$$

$$\eta = \frac{\eta}{\eta_\infty}$$

Where an asterisk represents non-dimensional variables, Infinity is the free-stream conditions and L denotes the reference length.

Although this seems as a trivial extension of the present work nevertheless its implementation can be lengthy and require care. This extension allows a more direct comparison of the performance of the scheme with respect to the type of discretization used.

**6.2.4** The program can be extended and use for the simulation for various flow regimes such as Darcy flow, flow through porous media, and combined free/porous flows. To achieve such simulation, the user will need to add one subroutine for the discretization of the flow governing equations and in case of combined free/porous flows some instructions to link the different flows regimes at interfaces.

## REFERENCES

Akin, J.E. (1982). Application and implementation of finite element methods.
Academic press, London.

Atluri, S. N, Gallagher, R. H. Zienkiewick, O. C. Pian, T. H. H. (c1983). Hybrid and Mixed finite element methods. Wiley, Chichester.

Altas, I., and Stephenson, J. w. 1991. A two-dimensional adaptive mesh generation methods. J. Comp. Phys., **94**, 201-24.

Axelsson, O (1994) Iterative solution methods. Cambridge: Cambridge University Press.

Babuska, B. 1971. Error bounds for finite element method. Numer. Math. **16**, 322-
333.

Babuska, I., and Suri, M., 1990. The p- and hp- versions of the finite element method. An overview. Comp. Meth. Appl. Mech. Eng., **80**, 5-26.

Babuska, I., Zienkiewicz, O.C., Gago, J., and Oliveira, E. R. A. (ed.)., 1986. Accuracy estimates and adaptive refinements in finite element computations. Chichester: Wiley.

Baker, A. J. (c1983). Finite element computational fluid mechanics. Washington
(D.C): London: Hemisphere.

Bathe, K. J. 1996. Finite element procedures. Prentice Hall, Englewood Cliffs, NJ.

Beer, G. and Haas, W (1982) A partitioned frontal solver for finite-element analysis. Int. J. Num. Meth. Eng., 18, No. ii, 1623-1654.

Beer, G. and Watson, J. O (c1992). Introduction to finite and boundary element met
hods for engineers. Wiley, Chichester.

Biggins, M. J. (1980). The numerical solution of elliptic partial differential equations
by finite difference methods. Thesis (PhD) Loughborough University of Technology.

Bochev, P. B. and Gunzburger, M. D. (2007). Least-squares finite element methods.
Springer, New York; London.

Bowen, M. (2005). High-order finite difference methods for partial differential equa
tions.  Thesis (PhD) Loughborough University.

Brackbill, J. U. and Saltzmann, J.S. 1982. Adaptive zoning for singular problems in two dimensions. J. Comp. Phys., **46**, 342.

Brenner, S.C. The mathematical theory of finite element methods. 2nd ed. New York; London: Springer, c2002.

Brezzi, F., 1974. On the existence, uniqueness and approximation of saddle point problems arising with Lagrange multipliers. RAIRO, Serie Rouge, **8R-2**, 129-151.
Butcher, J. C (1993) The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods. Chichester: Wiley, c1987.

Chen, C. and Shih, T. (c1998). Finite element methods for integro differential equa tions. Singapore, river edge, World Scientific, N. J.

Chorin, A.J. 1967. A Numerical method for solving incompressible viscous flows Problems. J. Comp. Phys., **2**, 12-26.

Ciarlet, P. G. Lions, J. L. Marchuk, G. I. Thomee, V. and Bjorck, A. (1980). Finite dif ference methods Part 1: Solution of equations in Rn (Part 1). Elsevier, Amsterdam, London.

Concus, P. Golub, G. H. and O'Leary, D. P (1976) A generalised conjugate gradient method for the numerical solution of elliptic partial differential equations. In J. R. Bunch and D J Rose (eds) sparse matrix computations, Academic Press, New York.

Courant, R. and Hilbert, D. (1953). Methods of mathematical physics, vol 1, John Wiley.

Courant, R. (1943). Variational methods for the solutions of problems of equilibrium and vibrations, Bull. Am. Math. Soc, vol 49, pp 1-23.

Curtis, F. G. and Patrick, O. W (1994) Applied numerical analysis, 5th ed; Reading, Mass.; Wokingham: Addison-Wesley.

Dahlquist G. and Bjork, A. (1974) Numerical methods. Englewood Cliffs, NJ: Prentice-Hall.

Darbandi, M. and Schneider, G. E. (1999). Application of an all-speed flow algorithm to heat transfer problems. Num. Heat Trans, **35**, 695-715.

Donea, J. and Huerta. (2003). Finite element methods for flow problems. Wiley, Chichester.

Duff, I. S (1986) Direct methods for sparse matrices, Oxford: Clarendon.

Duffy, D. J. (2006). Finite difference methods in financial engineering: A partial differential equation approach. Wiley, Chichester, England; Hoboken, NJ.

Duff, I. S. Erisman A. M. and Reid, J. K. (1986) Direct methods for sparse matrices, Oxford University Press, Inc., New York.

Duff, I. S. and Reid, J. K. (1983) The multifrontal solution of indefinite sparse symmetric linear, ACM Transactions on mathematical software (TOMS), vol 9, **3**, pp 302-325.

Dwyer, H. A., Smooke, D. Mitchell, and Kee, Robert. J. 1982. Adaptive gridding for finite difference solutions to heat and mass transfer problems. In J.F. Thompson (ed.). Numerical grid generation, New York: North-holland, 339.
D'Alessio, S. J. and Dennis, S. C. R. (1994). A vorticity model for viscous flow past a cylinder. Der. Comp. Fluids. **23**, 279-293.

Eiseman, P. R. 1985. Alternating direction adaptive grid generation. Comp. Meth. Appl. Mech. Eng., 64, 321-76.

El-Nakla, J. A. H. (1987). Finite difference methods for solving midly nonlinear Elliptic partial differential equations. Thesis (PhD) Loughborough University of Technology.

Fenner, R. T. (1996). Finite element methods for engineers. Imperial College Press, London.

Forsaith, J. and Moler, K. (1969) Numeric solution of systems of linear algebraic equations [Russian translation], Mir, Moscow.

Forsythe, G. E. and Wasow Wolfgang. R (1960). Finite difference methods for partial difference equations. Wiley.

Gallagher, R.H. and Zienkiewicz, O.C., 1983. Hybrid and mixed finite element meth ods. Wiley, Chichester.

Ghia, K. N., Osswald, G. A., and Ghia, U. 1989. Analysis of incompressible massively separated viscous flows using unsteady Navier-Stokes equations. Int. J. Num. Meth. Fl., **9**, 1025-50.

Ghia, U. Ghia, K.N. and Shin, C.T. (1982). High resolution for incompressible flow using the Navier-Stokes equations and a multigrid method. J. Comput. Phys. **48**, 387-411.

Girault. V and Raviart, P. A. (1986). Finite element methods for Navier-Stokes equa tions: Theory and algorithms. Berlin; New York: Springer-Verlag

Glowinski, R. and Wheeler, M. F (1987) Domain decomposition and mixed finite element methods for elliptic problems. In R. Glowinski et al. (ed.) Domain decomposition methods for partial differential equations, SIAM publication, 144-72.

Gnoffo, P. A. 1980. Complete supersonic flowfields over blunt bodies in a generalized orthogonal coordinate system. NASA TM 81784.

Greenbaun, A (1997) Iterative method for solving linear system. Philadelphia, Pa.: Society for industrial and applied mathematics.

Greenspan, D (1960) Theory and solution of ordinary differential equations, MacMillan 1960.

Gunzburger, M. D. (c1989). Finite element methods for viscous in compressible flows A guide to theory, practice, and algorithms. Boston Academic Press.

Hageman, L. A (1981) Applied iterative methods. New York; London: Academic Press.

Hanspal, N.S. Waghobe, A.N. Nassehi, V. and Wakeman, R.J. 2009. Development of a predictive mathematical model for coupled predictive mathematical model for coupled Stokes/Darcy flows in cross-flow membrane filtration. Chem.Eng. J.**149**, 132-142.

Heinrich, J. C. Huyakorn, P. S. Zienkiewick, O. C. and Mitchell, A. R (1977). An up Wind finite element scheme for two-dimensional convective transport equa tion. Int. J. Num. Meth. Eng, **11**, no. 1, 131-144.

Hestenes, M. R and Stiefel, E. L (1952) NBS J. Res., Vol 49, pp 409-436.

Hood, P. (1976) Frontal solution program for unsymmetric matrices. Int. J. Numer. Methods Eng. **10**, 379-399.

Irons, B. M. (1970) A frontal solution program for finite element analysis. Int J. Num. Meth. Eng., ~, No. I, 5-32.

Kardestuncer, H. Ed. Finite element handbook.

Kershaw, D. S (1978) The incomplete Cholesky conjugate gradient method for the iterative solution of linear equations. J. Comput. Phys., Vol 26, pp. 43-65.

KiKuchi, N. (1986). Finite element methods in mechanics. Cambridge University Press, Cambridge.

Kumar, B.V. Rathish and Naidu, K. B. (1998). A transient UVP finite element analy sis of a nonlinear pulsatile flow in a stenosed vessel. Int. J. Comp. Fluid. Dynamics, **9:1**, 71-76.

Ladyszhenskaya, O.A. 1969. The mathematical theory of viscous incompressible flow. Gordon and Breach, New York.

Lapidus, L (1971) Numerical solution of ordinary differential equations. New York; London: Academic Press.

Lapidus, L. and Pinder, G. F., 1982. Numerical Solution of Partial Differential Equations in Science and Engineering, Wiley, New York.

Lee, R.L. Gresho, P.M. and Sani, R.L. 1979. Smoothing techniques for certain primiti ve variable solutions of the Navier-Stokes equations. Int. J. Numer. Methods Eng. **14**, 1785-1804.

LeVeque, R. J. (2002). Finite volume methods for hyperbolic problems. Cambridge University Press, Cambridge.

Lewis, R.I. 1991. Vortex element methods for fluid dynamic analysis of engineering

Light, M. F. and Luxmoore, A. R. (1977) Application of the front solution to two-and three-dimensional elastoplastic crack problems. Int. J. Num. Meth. Eng., ii, No 2, 393-395.

Lions, P. L (1988) On the Scharz alternating method. In R. Gloswinski et al. (eds). Domain decomposition methods for partial differential equations Philadelphia: SIAM Publications, 1-42.

Lohner, R. (c2008). Applied computational fluid dynamics techniques: An introdu tion based on finite element methods, 2nd ed. John Wiley, Chichester.

Masson, C. Saabas, H. J. and Baliga, B. R. (1994). Co-located equal order control Volume finite element method for two-dimensional axisymmetric incompres sible fluid flow. Int. J. Num. Meth. Eng, **18,** 12-26.

Mura. T and Koya. T. (1992). Variational methods in mechanics. Oxford; New York.

Nakamura, S. 1982. Marching grid generation using parabolic partial differential equations. In J. F. Thompson (ed.). Numerical grid generation, New York: North-Holland. 775.

Nassehi, V. Hanspal, N. S. Waghode, A. N. Ruziwa, W. R. and Wakeman, R.J. (2005) finite element simulation of flow through pleated cartridge filters. Chem. Eng. Sci **. 60 (4),** 995-1006.

O'Brien, V. and Ehrlich, L.W. 1985. Simple pulsatile flow in an artery with a constric tion, J. Bio. Mech. **18**, pp 117-127.

Oden, J. T. (1972). Finite elements of nonlinear continua. McGraw-hill, New York.

Oden, J. T. 1988. Adaptive FEM in complex flow problems. In J. R. Whiteman (ed.). The mathematics of finite elements with applications, Vol 6, London: Academic Press Lt., 1-29.

Oden, J. T., Babuska, I., and Baumann, C. E., 1998. A discontinuous hp finite element method for diffusion problems. J. Comp. Phys., 146, 491-519.

Oden, J. T. 1989. Progress in adaptive methods in computational fluid dynamics. In J. Flaherty, et al (ed.). Adaptive methods for partial differential equations; Philadelphia: SIAM publications.

Oden, J. T., Strouboulis, T., and Devloo, P. 1986. Adaptive finite element methods for the analysis of inviscid compressible flow: I. Fast refinement/unrefinement and moving mesh methods for unstructured meshes. Comp. Meth. Appl. Mech. Eng., **59,** no 3, 327-62.

Oden, J. T., Wu, W., and Legat, V. 1995. An hp adaptive strategy for finite element approximations of the Navier-Stokes equations. Int. J. Num. Meth. Fl., **20**, 831-51.

Ozi Sik, M. N. (1994). Finite difference methods in heat transfer. Boca Raton, London: CRC.

Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O.C. 1987. Adaptive remeshing for compressible flow computations. J. comp. Phys., **72**, no 2, 449-66.

Pian, T. H. H and Wu Chang-Chun. (2006). Hybrid and incompatible finite element methods. Boca-Raton; Fla.: Chapman & Hall/CRC.

Pironneau, O. (1989). Finite element methods for flows. Wiley, New York.

Pittman, J.F.T. 1989. Finite elements for field problems. In: Tucker, C.L. III (ed); Computer modelling for polymer processing, chapter 6, Hanser publishers, Munich, pp. 237-331.

Platonov, A. D. and Trivailo, P. M. (1979) An algorithm for eliminating unknowns in the frontal method. Probl. Prochn., No 8, 30-31.

Poceski, A. 1992. Mixed finite element method. Springer-Verlag, Berlin; New York.

Postnov, V. A., Dmitriev, S. A., Eltyshev, B. K. and Rodionov, A. A (1979) Superelement method in computations of engineering structures. Postnov, V. A (ed.) [in Russian], Sudostroenie, - Leningrad.

Press, W. H., Flannery, B.P., Tewkolsky, S. A. and Velterling, W. T (1992). Numerical recipes- The art of scientific computing (FORTRAN version), Cambridge University Press, Cambridge.

Probert, J., Hassan, O., Peraire, J, and Morgan, K., 1991. An adaptive finite element method for transient compressible flows. Int. J. Num. Meth. Eng., **32**, 1145-59.

Ram Prakash Bharti, Chhabra, R. P. and Eswaran. V. (2006). Steady flow of power law fluids across a circular cylinder. The Can. J. Chem. Eng, Vol 84.

Reddy, J. N. (1986). Applied functional analysis and variational methods in engineers, McGraw-Hill, New York.

Saad, Y (1996) Iterative methods for sparse linear systems. Boston: PWS Publishing.

Sako, Y. 1962. Effect of turbulent blood flow and hypertension on experimental atherosclerosis. J. Am. Med. Ass. **179,** 36-40.

Scharz, H. A (1869) Uber einige abbidungsaufgauben J. fur die reine und Angewaudt Mathematik, **70**, 1005-20

Schneider, G. E. and Raw, M. J. (1987). Control volume finite element method for Heat transfer and fluid flow using collocated variables-1. Computational proce dure. Num. Heat Trans, **11**, 363-399.

Sewell, G (1988) The numerical solution of ordinary and partial differential equations. Academic Press.

Shampine, L. F (1994) Numerical solution of ordinary differential equations. New York; London: Chapman & Hall.

Shashkov, M. and Steinberg, S. (1996). Conservative finite difference methods on general grids. Boka Raton, Flo; CRC press London.

Smith, G. D. (1985). Numerical solution of partial differential equations: Finite differ rence methods, 3rd ed; Oxford, Clarendon.

Soares, A. A. Ferriera, J. M. and Chhabra, R. P. (2005). Flow and forced convection heat transfer in cross flow of non-Newtonian fluids over a circular cylinder. Ind. Eng. Chem. Res. 44, 5815-5827.

Thomas, J. W. (1995). Numerical partial differential equations: Finite difference methods. Springer, New York, London.

Traub, J. F. (1964) Iterative methods for the solution of equations. Prentice-Hall.

Varga, R. S (1962) Matrix iterative analysis. Englewood cliffs, NJ: Prentice-Hall.

Verma, A. K. and Eswaran, V. (1999). An overlapping control volume method for the Navier-Stokes equations on non-staggered grids. Int. J. Numer. Meth. Fluids. **30,** 279-308.

Versteeg, H. K and Malalasekera, W. (2007). An introduction to computational fluid dynamics: The finite volume method, 2$^{nd}$ ed. Harlow: Pearson.

Voller, V. R. Basic control volume finite element methods for fluids and solids. Hackensack, NJ. ; London: World scientific, c2009.

Wachspress, E. L (1966) Iterative solution of elliptic systems. Englewood cliffs, NJ: Prentice-Hall.

Wang, H. (c1982). Introduction to groundwater modelling: Finite difference and finite Element methods. Academic press, San Diego.

Wriggers, P. Nonlinear finite element methods. Berlin; London: Springer, c2008.

Yanenko, N.N. 1971. The method of fractional steps. Springer-Verlag, New York.

Zhang, Xing. 2006. Computation of viscous incompressible flow using pressure cor rection method on unstructured Chimera grid. Int. J. Comp. Fluid. Dynamics, **20:9**, 637-650.

Zienkiewick, O. C. and Cheung, Y. K. (1965). Finite elements in the solution of field problems. The engineer, 507-510.

Zienkiewick, O. C. and Codina, R. (1995). A general algorithm for compressible And incompressible flow-Part I. Characteristic-based scheme. Int. J. Num. Methods in Fluids, **20,** 869-885.

Zienkiewick, O. C. and Morgan, K. (1983). Finite element and approximation. Wiley New York.
Zienkiewick, O. C. and Taylor, R. L (1991). The finite element method, Vol 2. New York: McGraw-Hill.

Zienkiewicz, O.C. and Taylor, R.L. 1994. The finite element method, 4th ed, Vol 1 and 2, McGraw-Hill, London.

Zienkiewicz, O.C. and Wu, J. 1991. Incompressibility without tears-How to avoid Restrictions on mixed formulation. Int. J. Numer. Methods Eng. **32**, 1189-1203.

Zienkiewicz, O.C. 1971. The finite element method in engineering science, 2nd ed. New York: McGraw-Hill.

# APPENDICES

## APPENDIX 1

## SHAPE FUNCTIONS DERIVATION

A brief detail on how the isoparametric interpolation function used in this study is given in this appendix, reader interested in further information about this interpolation function may refer to Zienkiewicz (1971), Chung (2002), and Nassehi (2002).

Recall that name isoparametric is used to describe the element because the same parametric function which describes the geometry may be used to interpolate the field unknowns within an element. The isoparametric element is classified as one of the natural coordinate elements because of its use of the nondimensionalized coordinate.

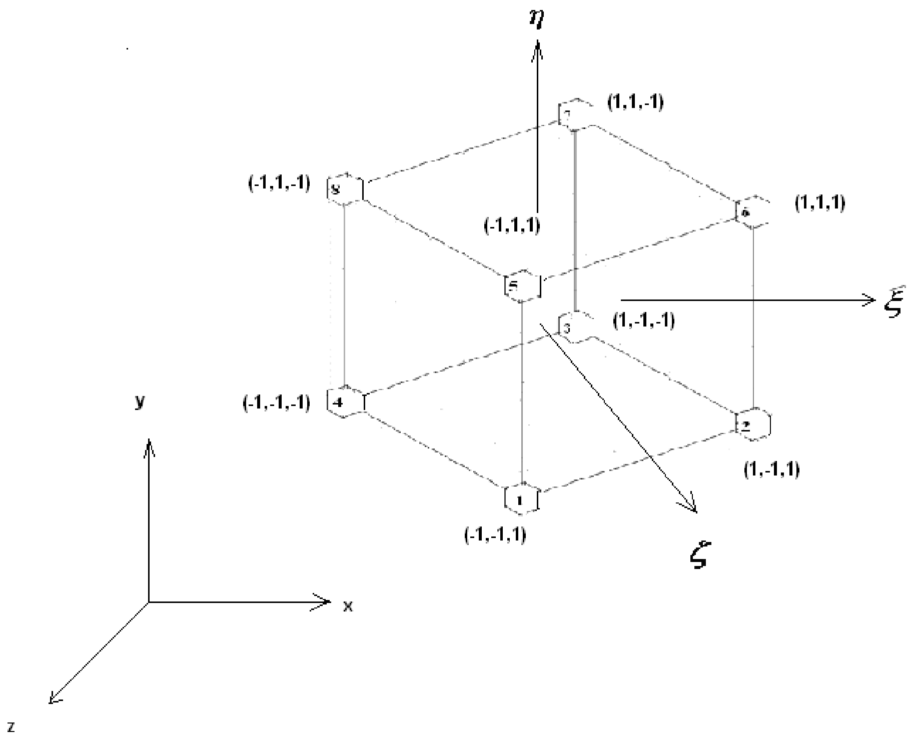


**Figure A1:** 8-noded hexahedral isoparametric element.

Consider the hexahedral element as shown in figure A1, the natural coordinates ($\xi, \eta, \zeta$) are related to the reference Cartesian coordinates (x, y, z) through the relation

$$x, y, z = \alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\zeta + \alpha_5\xi\eta\zeta + \alpha_6\xi\eta + \alpha_7\eta\zeta + \alpha_8\xi\zeta \qquad \text{(A1)}$$

It usually recommended to place the natural coordinates ($\xi, \eta, \zeta$) at the centroid of the element so that their values could range from 0 to ± 1, thus writing relation (A1) in term of the nodal values gives

$$\begin{aligned}
x_1 &= \alpha_1 + \alpha_2(-1) + \alpha_3(-1) + \alpha_4(1) + \alpha_5(-1)(-1)(1) + \alpha_6(-1)(-1) + \alpha_7(-1)(1) + \alpha_8(-1)(1) \\
x_2 &= \alpha_1 + \alpha_2(1) + \alpha_3(-1) + \alpha_4(1) + \alpha_5(1)(-1)(1) + \alpha_6(1)(-1) + \alpha_7(-1)(1) + \alpha_8(1)(1) \\
x_3 &= \alpha_1 + \alpha_2(1) + \alpha_3(-1) + \alpha_4(-1) + \alpha_5(1)(-1)(-1) + \alpha_6(1)(-1) + \alpha_7(-1)(-1) + \alpha_8(1)(-1) \\
x_4 &= \alpha_1 + \alpha_2(-1) + \alpha_3(-1) + \alpha_4(-1) + \alpha_5(-1)(-1)(-1) + \alpha_6(-1)(-1) + \alpha_7(-1)(-1) + \alpha_8(-1)(-1) \\
x_5 &= \alpha_1 + \alpha_2(-1) + \alpha_3(1) + \alpha_4(1) + \alpha_5(-1)(1)(1) + \alpha_6(-1)(1) + \alpha_7(1)(1) + \alpha_8(-1)(1) \\
x_6 &= \alpha_1 + \alpha_2(1) + \alpha_3(1) + \alpha_4(1) + \alpha_5(1)(1)(1) + \alpha_6(1)(1) + \alpha_7(1)(1) + \alpha_8(1)(1) \qquad \text{(A2)} \\
x_7 &= \alpha_1 + \alpha_2(1) + \alpha_3(1) + \alpha_4(-1) + \alpha_5(1)(1)(-1) + \alpha_6(1)(1) + \alpha_7(1)(-1) + \alpha_8(1)(-1) \\
x_8 &= \alpha_1 + \alpha_2(-1) + \alpha_3(1) + \alpha_4(-1) + \alpha_5(-1)(1)(-1) + \alpha_6(-1)(1) + \alpha_7(1)(-1) + \alpha_8(-1)(-1)
\end{aligned}$$

The system of equations given by (A2) can be written in a matrix form as

$$[x] = [M][\alpha] \qquad \text{(A3)}$$

Where the matrix [M] is given by

$$[M]=\begin{bmatrix}1 & -1 & -1 & 1 & 1 & 1 & -1 & -1\\ 1 & 1 & -1 & 1 & -1 & -1 & -1 & 1\\ 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1\\ 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1\\ 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & -1 & -1 & 1 & -1 & -1\\ 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1\end{bmatrix}$$

And $[\alpha]$ by

$$[\alpha]=[M]^{-1}[x] \qquad \text{(A4)}$$

With

$$[M]^{-1}=\frac{1}{8}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1\\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1\\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1\\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1\\ -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1\\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1\end{bmatrix}$$

Substituting (A4) into (A1) yields

$$x_i=\phi_N^{(e)}x_{N_i}$$

Where

$$\phi_N^{(e)} = \frac{1}{8}(1+\xi_{N1}\xi_1)(1+\xi_{N2}\xi_2)(1+\xi_{N3}\xi_3) \qquad \text{(A5)}$$

If one sets $\xi_1 = \xi, \xi_2 = \eta, \text{and}\, \xi_3 = \zeta$ then substituting the nodal values of $\xi_{N1}, \xi_{N2}, \text{and}\, \xi_{N3}$ into equation (A5) give the height interpolation functions as

$$\phi_1^{(e)} = \frac{1}{8}(1-\xi)(1-\eta)(1+\zeta)$$
$$\phi_2^{(e)} = \frac{1}{8}(1+\xi)(1-\eta)(1+\zeta)$$
$$\phi_3^{(e)} = \frac{1}{8}(1+\xi)(1-\eta)(1-\zeta)$$
$$\phi_4^{(e)} = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta)$$
$$\phi_5^{(e)} = \frac{1}{8}(1-\xi)(1+\eta)(1+\zeta)$$
$$\phi_6^{(e)} = \frac{1}{8}(1+\xi)(1+\eta)(1+\zeta)$$
$$\phi_7^{(e)} = \frac{1}{8}(1+\xi)(1+\eta)(1-\zeta)$$
$$\phi_8^{(e)} = \frac{1}{8}(1-\xi)(1+\eta)(1-\zeta)$$

## APPENDIX 2

## INPUT FILE FORMAT

| Heading & Formats | Variables | Description |
|---|---|---|
| Line 1 Format (A) | Title | Title of the input file |
| **Basic control variables** | | |
| Line 2 Format(2I5) | | |
| Variable 1 | ncn | Number of nodes per element |
| Variable 2 | ngaus | Number of integration full points |
| **Mesh data** | | |
| Line 3 Format(4I5) | | |
| Variable 3 | nnp | Total number of nodes |
| Variable 4 | nel | Total number of elements |
| Variable 5 | nbc | Total number of boundary conditions |
| Variable 6 | nmat | Total number of materials |
| **Output control** | | |
| Line 4 Format(2I5) | | |
| Variable 7 | ntep | Number of iteration |
| Variable 8 | icord | Coordinate system selection ( 0 for Cartesian and 1 for cylindrical) |
| **Gravity force data** | | |
| Line 5 Format(3F10.0) | | |
| Variable 9 | grav1 | Body force in x-direction |
| Variable 10 | grav2 | Body force in y-direction |
| Variable 11 | grav3 | Body force in z-direction |
| **Convergence tolerance** | | |
| Line 6 Format(3F10.5) | | |
| Variable 12 | tolv | Velocity convergence tolerance factor |
| Variable 13 | tolp | Pressure convergence tolerance factor |
| Variable 14 | tolc | Concentration convergence tolerance factor |
| **Rheological & physical data** | | |
| Line 7 Format(9D10.5) | | |
| Variable 15 | rvisc | Consistency coefficient in the power law model |
| Variable 16 | power | Power law index |
| Variable 17 | tref | Reference temperature |
| Variable 18 | tbco | Coefficient b in the power law model |
| Variable 19 | taco | Coefficient a in the power law model |
| Variable 20 | dispc | Coefficient for convective equation |
| Variable 21 | pref | Reference pressure |
| Variable 22 | roden | Density |
| Variable 23 | gamad | Shear rate |

| Heading & Formats | Variables | Description |
|---|---|---|
| **Nodal coordinates** | | |
| Line8-linemFormat(I7,3E20.12) | | |
| Variable 24 | m | Node number m |
| Variable 25 | x(m) | X-coordinate of node m |
| Variable 26 | y(m) | Y-coordinate of node m |
| Variable 27 | z(m) | Z-coordinate of node m |
| **Element connectivity data** | | |
| Line m-line n Format(21I7) | | |
| Variable 28 | n | Element number n |
| Variable 29 | node(n,1) | Node number 1 of element number n |
| Variable 30 | node(n,2) | Node number 2 of element number n |
| Variable 31 | node(n,3) | Node number 3 of element number n |
| Variable 32 | node(n,4) | Node number 4 of element number n |
| Variable 33 | node(n,5) | Node number 5 of element number n |
| Variable 34 | node(n,6) | Node number 6 of element number n |
| Variable 35 | node(n,7) | Node number 7 of element number n |
| Variable 36 | node(n,8) | Node number 8 of element number n |
| **Boundary condition data** | | |
| Line n-line k Format(2I5,F10.4) | | |
| Variable 37 | ibc | Node number at which the boundary condition is applicable |
| Variable 38 | jbc | = 1 for x-direction velocity<br>= 2 for y-direction velocity<br>= 3 for z-direction velocity<br>= 4 for pressure |
| Variable 39 | vbc | Boundary condition value |

# APPENDIX 3

# PROGRAM LISTING

## 3.1 Sample input file

Sample input file

```
   8    3
 9062 7560 8390    1
   1    0
    0.000     0.000     0.000
  0.00001   0.00001   0.00001
.80000D+02.10000D+01.29300D+03.14000D-
01.20000D+00.20000D+00.10132D+06.10000D+04.20000D+00
      1  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
      2  0.000000000000E+00  0.833333470000E-02  0.000000000000E+00
      3  0.000000000000E+00  0.166666680000E-01  0.000000000000E+00
      4  0.000000000000E+00  0.250000000000E-01  0.000000000000E+00
      5  0.000000000000E+00  0.333333350000E-01  0.000000000000E+00
      6  0.000000000000E+00  0.416666680000E-01  0.000000000000E+00
      7  0.000000000000E+00  0.500000010000E-01  0.000000000000E+00
      .
      .
      .
   9056  0.100000000000E+01  0.583333340000E-01  0.937500000000E-01
   9057  0.100000000000E+01  0.100000000000E+00  0.100000000000E+00
   9058  0.100000000000E+01  0.916666690000E-01  0.100000000000E+00
   9059  0.100000000000E+01  0.833333430000E-01  0.100000000000E+00
   9060  0.100000000000E+01  0.750000030000E-01  0.100000000000E+00
   9061  0.100000000000E+01  0.666666700000E-01  0.100000000000E+00
   9062  0.100000000000E+01  0.583333340000E-01  0.100000000000E+00
     1    79    2    1    78    86    9    8    85
     2    80    3    2    79    87    10    9    86
     3    81    4    3    80    88    11    10    87
     4    82    5    4    81    89    12    11    88
     5    83    6    5    82    90    13    12    89
     6    84    7    6    83    91    14    13    90
     7    86    9    8    85    93    16    15    92
     .
     .
     .
  7554   8802   8774   9026   9050   8809   8781   9032   9056
  7555   9052   9028   9027   9051   9058   9034   9033   9057
  7556   9053   9029   9028   9052   9059   9035   9034   9058
  7559  9056  9032  9031  9055  9062  9038  9037  9061
  7560  8809  8781  9032  9056  8816  8788  9038  9062
   1   1   0.1000
   1   2   0.0000
   1   3   0.0000
   2   1   0.1000
   2   2   0.0000
   2   3   0.0000
   3   1   0.1000
   3   2   0.0000
   3   3   0.0000
   4   1   0.1000
   4   2   0.0000
   4   3   0.0000
   5   1   0.1000
   5   2   0.0000
   5   3   0.0000
   6   1   0.1000
   6   2   0.0000
   6   3   0.0000
   7   1   0.1000
   7   2   0.0000
   7   3   0.0000
   .
   .
   .
 9060   3   0.0000
 9061   1   0.0000
 9061   2   0.0000
 9061   3   0.0000
 9062   1   0.0000
 9062   2   0.0000
 9062   3   0.0000
   .
   .
   .
 9021   4   0.0000
 9027   4   0.0000
 9033   4   0.0000
 9039   4   0.0000
 9045   4   0.0000
 9051   4   0.0000
 9057   4   0.0000
```

```
      Program StokesSolution

c ========================================================== c
c                                                            c
c   This program is for the solution of generalised newtonian fluids of          c
c   incompressible fluids governed by Stokes equations. The solution is          c
c   obtained via the weighted residual galerkin finite element method in         c
c   conjunction with the use of 8 nodes isoparametric hexahedral elements.       c
c                                                            c
c   The entire computational domain is assumed to be isothermal.                 c
c                                                            c
c   Two numerical schemes are developed for the problem solutions:               c
c   The U-V-W-P method in which velocity components and pressure are regarded    c
c   as primitive variables and discretized as unknowns.                          c
c                                                            c
c   The second scheme is a modification of the U-V-W-P method with  a faster     c c
convergence  rate                                                  c
c                                                            c
c   During the program running after typing in some basic data, the user will prompt  c  c
to select with scheme to use.                                      c
c                                                            c
c   The apparent viscosity is calculated and updated using the power law model   c
c                                                            c
c   The system of algebraic equations obtained after the discretization process is  c
c   solved by frontal method.                                                    c
c                                                            c
c   A complete list of options is given on the program listing.                  c
c                                                            c
c   The program consists of a main module and subroutines and among the are two  c
c                                                            c
c   output subroutines written for Tecplot and Cosmos Geostar for visualizations.  c
c                                                            c
c   The program is written in FORTRAN programming language                       c
c                                                            c
c   Developed (not from scratch) by N. Rupert. Boukanga (last updated April 2010)  c    c
c
c   The author thanks Prof V. Nassehi for his hard work by originally developing the c
c   the majority of the subroutines                                              c
c                                                            c

c========================================================== c
```

```
c    unit                 contents                                                  c
c============================================================ c
c      51    i            input data file                                          c
c            i                                                                     c
c      60    i            output file for documentation                            c
c            i                                                                     c
c      11    i            output file containing velocity field data for  plotting c
c            i            plotting                                                 c
c            i                                                                     c
c      12    i   output file containing concentration data for  contour plotting   c
c            i                                                                     c
c      14    i   used as a work file in the solver routine                         c
c            i                                                                     c
c      15    i   stores shape functions and their derivatives at   'full'          c
c            i    integration points                                               c
c            i                                                                     c
c      17    i   output file containing pressure data for                          c
c            i   contour plotting                                                  c
c            i                                                                     c
c      20    i   output file containing elemental stiffness matrix                 c
c            i   for element number 14 as seen on the mesh                         c
c            i                                                                     c
c      610   i    output for Cosmos Geostar post-processing                        c
c      614   i    output for Tecplot post-processing                               c
c                                                                                  c
c ============================================================ c
c                                                                                  c
c    List of variables                                                             c
c    =================                                                             c
c    aa   (   27,  27)   element coefficient matrices on LHS                       c
c    K    (   27,  27)   element coefficient matrices on RHS                       c
c    b    (    3,  20)   global derivatives of shape functions                     c
c    bc   (maxdf     )   nodal constraints (boundary conditions)                   c
c    conc (maxnp     )   nodal concentrations                                      c
c    cord (maxnp,ndim)   nodal coordinates                                         c
c    del  (    3,  20)   local  derivatives of shape functions                     c
c    vel  (maxdf     )   nodal velocities  (displacements)                         c
c    dsc1, dsc2          depths of slip layers                                     c
c    grav1               first  component of the applied body force                c
c    grav2               second component of the applied body force                c
c    icord               indicates whether the coordinate system is cartesian (planar)  c
c                        or cylindrical (axisymmetric)                             c
c    tolc                convergence tolerance factor for concentration            c
c    tolp                convergence tolerance factor for pressures                c
c    tolv                convergence tolerance factor for velocities               c
```

```
c    ndf               degree of freedom per node                                 c
c    ndim              dimensions of the solution domain                          c
c    nel               total number of elements                                   c
c    ngaus             number of integration points                               c
c    nnp               total number of nodal points                               c
c    node (maxel,maxst)   element connectivity                                    c
c    nter              maximum number of iterations for non-newtonian case        c
c    num               number of integration points per element                   c
c    p   (   20   )   shape functions                                      c
c    press(maxnp    )   nodal pressures                                    c
c    r1   (maxdf    )   global load vector  (r.h.s.)                        c
c    rfrct             friction coefficient (slip)                         c
c    rr   (   27    )   element load vector                                c
c    stiff(maxar    )   global stiffness matrix ( a in ax=r.h.s.)           c
c    rvisc             mu nought;consistency coefficient in power-law model c
c    power             power law index                                     c
c    stemp             temperature                                         c
c    rtem              reference temperature                               c
c    spress            pressure                                            c
c    rpress            reference pressure                                  c
c    tco               coefficient relating viscosity to temperature       c
c    pco               coefficient relating viscosity to pressure          c
c    gamad             shear rate                                          c
c    nwr               no. of sample nodes for recording transient solutions c
c                                                                          c
c ======================================================================== c
c    List of Subroutines                                                   c
c    ===================                                                   c
c                                                                          c
c    bacsub            backsubstitution method for finding the final       c
c                      solution vector                                     c
c    clean             cleans the arrays and prepares them for             c
c                      solution                                            c
c    conc              calculates the concentrations                       c
c    contol            makes a check for the convergence                   c
c    deriv             calculates the jacobian matrix, its determinant     c
c                      and   global derivatives of the shape functions     c
c    flow Stokes       calculates the velocities and pressures via the UVWP method  c
c    flow Stokes2      calculates the velocities and pressures via the modified     c
c                      UVWP method                                         c
c    front             frontal method for solving the final set            c
c                      of equations                                        c
```

```
c     getmat          reads the input material data                    c
c     getnod          reads the nodal co-ordinates for cartesian       c
c                     and axisymmetric systems                         c
c     lumpm           evaluates the terms of the mass matrix           c
c     output          prints the final solution                        c
c     putbcv          imposes the primary boundary conditions for      c
c                     velocity                                         c
c     putbcc          imposes the primary boundary conditions for      c
c                     concentration                                    c
c     setprm          Sets the location data for nodal degrees of      c
c                     freedom                                          c
c     shape           calculates the shape functions and their         c
c                     derivatives                                      c
c     slip            identifies the upper and lower boundary          c
c                     layers.                                          c
c     stress          calculates stress components at integration      c
c                     points                                           c
c     visca           calculates the viscosity                         c
c     viscb           calculates virtual viscosity for slip walls      c
c                                                                      c
c===================================================================== c

      parameter (maxel  = 60000    )
      parameter (maxnp  = 37000    )
      parameter (maxbc  = 20000    )
      parameter (maxdf  = maxnp*4  )
      parameter (maxst  = 80       )
      parameter (maxfr  = 5000     )
      parameter (ndim   = 3        )

      implicit real*8 (a-h,o-z)
c
c     Storage allocation
c     ==================
c
      dimension title (       80)
      dimension node  (maxel,maxst) ,pmat (maxel,  9) ,cord (maxnp,ndim)
      dimension ncod  (maxdf      ), bc   (maxdf    )
      dimension ibc   (maxbc      ) ,jbc  (maxbc    ) ,vbc  (maxbc    )
      dimension vel   (maxdf      ) ,conc (maxnp    ) ,press(maxnp    )
      dimension r1    (maxdf      )
      dimension clump (maxnp      ) ,stres(maxnp,  6)
      dimension vet   (maxdf      ) ,cet  (maxnp    ) ,pet  (maxnp    )
      dimension ldest (maxst      ) ,kdest(maxst    ) ,nk   (maxst    )
      dimension eq    (maxfr,maxfr) ,lhed (maxfr    ) ,khed (maxfr    )
      dimension kpiv  (maxfr      ) ,lpiv (maxfr    ) ,jmod (maxfr    )
      dimension qq    (maxfr      ) ,pvkol(maxfr    ) ,sinv (maxel, 27)
      dimension mdf   (maxdf      ) ,ndn  (maxdf    )
      dimension ldsc  (22         )
      dimension temp  (maxnp      ) ,actpress(maxnp)
      dimension rmat1 (maxel,   13) ,rmat2(maxel, 13)
      character *20 filnam
c
c    Opening of input and output data files
c==================================================c

      call GFMFEM

         print*,'enter the name of your data file'
         read(*,2000) filnam

      open(unit=51,File=filnam,access='sequential',form='formatted',
     1   status="unknown",iostat=ios)

         open(unit=60,file='res.txt',access='sequential',form='formatted',
     1   status="unknown",iostat=ios)

         open(unit=17,file='stress.txt',access='sequential',form='formatted',
     1   status="unknown",iostat=ios)
      open(unit=20,file='stiffmat',access='sequential',form='formatted',
     1   status="unknown",iostat=ios)


      open(unit=14,form='unformatted',status='scratch',iostat=ios)
      open(unit=15,form='unformatted',status='scratch',iostat=ios)

       if(ios==0)then
                    print*,"files opened"
          else
                    print*,"files not opened"
          stop
          end if

         rewind 51
         rewind 60
         rewind 20
      do 5010 itl = 1,maxel
      do 5010 ivl = 1,80
                      node (itl,ivl) = 0
5010  continue
      do 5020 itl = 1,maxel
      do 5020 ivl = 1,8
                      pmat (itl,ivl) = 0.0
5020  continue
      do 5030 itl = 1,maxnp
      do 5030 ivl = 1,3
                      cord (itl,ivl) = 0.0
5030  continue
      do 5040 itl = 1,maxnp
      do 5040 ivl = 1,6
                      stres(itl,ivl) = 0.0
5040  continue
      do 5050 itl = 1,maxdf
                           vel  (itl    ) = 0.0
5050  continue
      do 5060 itl = 1,maxdf
                      ncod   (itl)   = 0
                      r1     (itl)   = 0.0
                      bc     (itl)   = 0.0
                      vet    (itl)   = 0.0
                      mdf    (itl)   = 0
                      ndn    (itl)   = 0
                      nopp   (itl)   = 0
5060  continue
      do 5070 itl = 1,maxnp
                      clump  (itl)   = 0.0
                      cet    (itl)   = 0.0
                      conc   (itl)   = 0.0
                                                        pet    (itl)   = 0.0
                                                        press  (itl)   = 0.0
5070  continue
      do 5080 itl = 1,maxbc
                      ibc   (itl)    = 0
                      jbc   (itl)    = 0
                      vbc   (itl)    = 0.0
5080  continue
      do 5090 itl = 1,20
                      del   (1,itl)  = 0.0
                      del   (2,itl)  = 0.0
```

```
                 kdest (itl  )  = 0
                 nk    (itl  )  = 0
5100  continue
    do 5110 itl = 1,maxfr
                 lhed  (itl  )  = 0
                 khed  (itl  )  = 0
                 kpiv  (itl  )  = 0
                 lpiv  (itl  )  = 0
                 jmod  (itl  )  = 0
                 qq    (itl  )  = 0.0
                 pvkol (itl  )  = 0.0
    do 5110 ill = 1,maxfr
                 eq    (itl,ill)= 0.0
5110  continue
c---------------------------------------------------------------------

c    Title of the program
c    ====================

        if(.not. eof(51)) read (51,2010) title
    write(60,4010) title

c    Element description data
c    ========================

    if (.not. eof(51)) read (51,2020) ncn ,ngaus
        print*, "ncn, ngaus read"
    write(60,4020) ncn ,ngaus

c    Mesh, boundary condition and material parameters
c    ================================================

        if (.not. eof(51)) read (51,2030) nnp  ,nel  ,nbc  ,nmat
        print*, "nnp, nel ,nbc ,nmat read"
    if (.not. eof(51)) read (51,2040) ntep ,icord
c
    if(icord.eq.0) write(60,4030)
    if(icord.eq.1) write(60,4040)

    write(60,4050)

    if(ntep.eq.0) ntep=1
```

```
c    if ntep = 1 then computed result after every iteration will
c    be printed ;if you do not need the result of intermediate
c    computations choose your own ntep;the result of first and
c    converged solutions will always be printed.
c    ==========================================================

           if(nnp   .eq.0  .or.nnp  .gt.maxnp) then
                              write(60,4060)
        elseif(nel   .eq.0  .or.nel  .gt.maxel) then
                              write(60,4060)
        elseif(nbc   .eq.0  .or.nbc  .gt.maxbc) then
                              write(60,4060)
        elseif(nmat  .eq.0  .or.nmat .gt.maxel) then
                              write(60,4060)

                 print*, "the program is aborted"
        stop

                    endif

        write(60,4070) nnp ,nel ,nbc ,nmat

    if (.not. eof(51)) read (51,2050) grav1, grav2, grav3
        print*, "grav1 grav2 grav3 read"
    write(60,4080) grav1, grav2, grav3

    if (.not. eof(51)) read(51,2060) tolv ,tolp, tolc
    print*, "tolv, tolp, tolc read"

    maxer=maxel

c
=============================================================
============
c    Read input data from main data file and prepare arrays for solution process
c
=============================================================
============

     call getmat(nel,nmat,pmat,51,60,maxel,rtem,rpef)
     call getnod(nnp,cord,51,60,maxnp,ndim,icord)
     call getelm(nel,ncn,node,51,60,maxer)
     call getbcd(nbc,ibc,jbc,vbc,51,60,maxbc)
```

```
c==============================================================c
c                              Start of the time loop
c==============================================================c
c
c     Set control parameters (default values are overwritten by input data
c     if specified)
c
c     ncn      number of nodes per element
c     ngaus    number of integration points
c     nter     maximum number of iterations for non-newtonian case
c     ndim     number of space dimensions in the solution domain
c
c      nter  = 5
      num = 13
                                    do 5125 iel = 1,maxel
                                      do 5125 lg = 1,num
                                      sinv(iel,lg)=0.0
5125                              continue

                                do 5130 ivel= 1,maxdf
                                  vel (ivel)  =   0.0
5130                              continue

                                do 5140 item= 1,maxnp
                                  temp(item)  =    rtem
5140                              continue

c===============================================================c
c                        Transient data
c
c===============================================================c
c
c     stime                   starting time
c
c     deltat                  time increment
c
c     alpha      indicates the choice of method being employed in alpha
c                  time stepping technique (backward difference,
c                                   forward difference, central difference, galerkin)
c
c     nter       maximum number of time steps being employed for finding solution
c==============================================================
     print*,"  "
```

```
      print*,"Enter the number of time steps desired"
c
      read*, nter

      write(60,4100) nter

      print*,"Enter the delta t desired"
c
      read*, deltat

      write(60,4110) deltat


1111    print*," Enter the value of alpha: any number between 0 and 1 "

        read*, alpha

        if(alpha < 0 .OR. alpha > 1) then

        print*, " Invalid alpha value entered, type in another value"

        goto 1111

        end if

      print*, "alpha=",alpha

        tcode = 0

         Print *,'          '

          print*," Select a scheme "


         Print *,'1: UVW-P scheme with Taylor-Galerkin method'
         Print *,'2: Modified UVW-P scheme with Taylor-Galerkin method'

         read(*,*) tcode

      do 5150 iter = 1 ,nter
      print*,'iter=',iter
        time = iter*deltat
```

```
c===========================================================c
c                   Calculate Nodal Velocities & Pressures
c===========================================================c
     icho=1
                      rewind 11
                      rewind 14
                      rewind 15
                      ndf   = 4
                      ntov  = ndf * nnp
                      ntrix = ndf * ncn
     call clean
    1    (ncn  ,nel  ,ndf  ,node ,r1    ,maxel,maxst,maxdf,
    2     bc   ,ncod ,icho )

     call setprm
    1    (nnp  ,nel  ,ncn  ,node ,ndf  ,maxel,maxst,ndn  ,ntrix,
    2     maxdf,ntov ,mdf  ,nopp )

     call putbcv
    1  (nnp ,nbc ,ibc ,jbc ,vbc ,ncod ,bc,maxbc,maxdf,maxel,maxst,
    2   node)


c    idv4 is the file specifier for unit=20
c    ======================================

     idv4=20


       do 5160 iel=1,nel



       if (tcode ==1) then

      call flowStokes(node ,cord ,pmat ,nopp ,mdf ,ndn ,ncod ,
         1bc ,vel  ,press, r1, temp,ldest,kdest,nk ,eq ,lhed ,khed ,kpiv ,lpiv,
    2jmod, qq, pvkol, iter ,nel ,ncn , ngaus,grav1,
    3grav2, grav3, p, del, b, ntrix, maxel, maxnp, maxst, maxfr, maxdf,
    4ndim ,aa ,xg ,da ,ntov ,num, icord, rr, iel, del1,deltat,alpha,
    5idv4,sinv, icho, nnp, tref)
```

```
   2jmod, qq, pvkol, iter ,nel ,ncn , ngaus,grav1,
   3grav2, grav3, p, del, b, ntrix, maxel, maxnp, maxst, maxfr, maxdf,
   4ndim ,aa ,xg ,da ,ntov ,num, icord, rr, iel, del1,deltat,alpha,
   5idv4,sinv, icho, nnp, tref)

     else
          stop
          endif


5160  end do

c============================================================c
c           calculates the second invariant of rate of deformation          c
c           tensor at integration points.                                   c
c
c============================================================c
     call secinv
   1   (nel  ,nnp  ,ncn  ,ngaus,node ,sinv ,cord ,p  ,b,
   2    del  ,da   ,vel  ,maxnp,maxel,maxst,ndim ,icord,
   3    maxdf,num)

c============================================================c
c                        Convergence check
c    c============================================================c

     call contol(vel ,conc ,iter ,ntov ,nnp,maxnp,maxdf,errov, errop
          1,vet ,cet, pet, press)

c=============================================================c
c *** calculation of the nodal stress;using variational recovery
c=============================================================c
     call lumpm
   1    (clump,nnp  ,maxnp,nel  ,ngaus,p   ,del   ,b  ,maxst,
   2     node ,maxel,ncn  )
c
     call stress
   1     (nel,nnp,ncn ,node ,p , b , da ,vel  ,maxnp, maxel,  maxst ,
   2      maxdf, stres, press, rvisc ,clump ,ngaus  )


c  ===============
c    Print the output
```

```
     if(iter.eq.1.or.iiter.eq.iter) then

          call output
          1   (nnp  ,vel  ,press, maxdf,maxnp,icord, stres)

          end if

          if(iter.eq.nter) then


          call cosmos
          1      (nnp  ,  vel  ,  press   ,  maxdf  ,  maxnp  , icord   ,
     2      pmat ,  maxel,  actpress, nel                       )

     call cosmos2
          1      (nnp  ,  vel  ,  press   ,  maxdf  ,  maxnp  , icord   ,
     2      pmat ,  maxel,  actpress, nel                       )

     call tecplot
          1      (nnp  ,  vel  ,  press   ,  maxdf  ,  maxnp  , icord   ,
     2      pmat ,  maxel,  actpress,  cord   ,  ncn    , nel     ,
     3      node ,  ndim  )

     call tecplot2
          1      (nnp  ,  vel  ,  press   ,  maxdf  ,  maxnp  , icord   ,
     2      pmat ,  maxel,  actpress,  cord   ,  ncn    , nel     ,
     3      node ,  ndim  )

          endif
c
c
c=========================================================== =c
c                    End of time loop
c    c c
c============================================================c

5150  continue

     close(51)
     close(unit=60)
     close(unit=11)
     close(unit=14)
     close(unit=15)
```

```
c    c==========================================================c
c
2000  format(a)
2010  format(80a)
2020  format(2i5)
2030  format(4i5)
2040  format(2i5)
2050  format(3f10.0)
2060  format(3f10.5)
c
c
c==========================================================c
c                    Write statements
c==========================================================c

4010  format(' ',5(/),' ',20x,60('*'),/' ',20x,'*',58x,'*',/
     1' ',20x,'*',' A three dimensional finite element model of a ',
     27x,'*',/' ',20x,'*',' non-newtonian isothermal flow using ',
     320x,'*',/' ',20x,'*',' the UVWP method. ',38x,'*',/' ',20x,'*',
     558x,'*',/' ',20x,60('*')///,' ',20x,80('-'),/' ',20x,80a,/' ',
     620x,80('-'),///)
c
4020  format(' ',20x,3('['),' element description data',10('.'),/
     125x,'no.of nodes per element              =',i10,/
     225x,'no.of integration points             =',i10,/
     3//)
c
4030  format('    *** coordinate system is cartesian (planar) ***')
4040  format('*** coordinate system is cylindrical(axisymmetric) ***')
4050  format(' ')
4060  format(' ',10('['),'input data unacceptable',10(']')///)
c
4070  format(' ',20x,3('['),' mesh description data ',10('.'),/
       125x,'no.of nodal points                  =',i10,/
     225x,'no.of elements                      =',i10,/
     325x,'no.of nodal constraints on boundary    =',i10,/
     425x,'no.of different materials            =',i10,//)
c
4080  format(' ',20x,3('['),' uniform body force vector ',10('.'),/
     125x,'grav1                          =',f15.4,/
     225x,'grav2                          =',f15.4,/
     325x,'grav3                          =',f15.4,//)
4090  format(///'     iteration no.',i5,//)
4100  format(///' Total number of time steps  =',i5,//)
```

```
c============================================================c
      end program
c============================================================c

      subroutine gaussp(ngaus,xg,cg)
c
      implicit double precision(a-h,o-z)
c
c      x(g)   specifies the coordinates of the Gauss points
c      c(g)   specifies the Gauss weights
c
      dimension xg(3),cg(3)

      if(ngaus.eq.1) then
      xg(1)=0.0
      cg(1)=2.0
      elseif(ngaus.eq.2)  then
      xg(1) = 0.57735026919d00
      xg(2) = -xg(1)
      cg(1) = 1.00
      cg(2) = 1.00
      else
      xg(1) = 0.77459666924d00
      xg(2) = 0.0
      xg(3) = -xg(1)
      cg(1) = 0.55555555556d00
      cg(2) = 0.88888888889d00
      cg(3) = cg(1)
c
      endif
      return
      end
c
c===========================================================c




      subroutine shape ( xi , eta , zeta, p ,del , ncn )
      implicit double precision (a-h,o-z)
c
      DIMENSION p(20) ,del(3,20)
      if (ncn.eq.8) then
        del(1,1)=-0.125*(1-eta)*(1-zeta)
        del(1,5)=-0.125*(1-eta)*(1+zeta)
        del(1,6)=-0.125*(1+eta)*(1+zeta)
        del(1,7)= 0.125*(1+eta)*(1+zeta)
        del(1,8)= 0.125*(1-eta)*(1+zeta)
C.....................................................
        del(2,1)=-0.125*(1-xi)*(1-zeta)
        del(2,2)= 0.125*(1-xi)*(1-zeta)
        del(2,3)= 0.125*(1+xi)*(1-zeta)
        del(2,4)=-0.125*(1+xi)*(1-zeta)
        del(2,5)=-0.125*(1-xi)*(1+zeta)
        del(2,6)= 0.125*(1-xi)*(1+zeta)
        del(2,7)= 0.125*(1+xi)*(1+zeta)
        del(2,8)=-0.125*(1+xi)*(1+zeta)
C.........................................................
        del(3,1)=-0.125*(1-xi)*(1-eta)
        del(3,2)=-0.125*(1-xi)*(1+eta)
        del(3,3)=-0.125*(1+xi)*(1+eta)
        del(3,4)=-0.125*(1+xi)*(1-eta)
        del(3,5)= 0.125*(1-xi)*(1-eta)
        del(3,6)= 0.125*(1-xi)*(1+eta)
        del(3,7)= 0.125*(1+xi)*(1+eta)
        del(3,8)= 0.125*(1+xi)*(1-eta)
C.........................................................

        p(1)=0.125*(1-xi)*(1-eta)*(1-zeta)
        p(2)=0.125*(1-xi)*(1+eta)*(1-zeta)
        p(3)=0.125*(1+xi)*(1+eta)*(1-zeta)
        p(4)=0.125*(1+xi)*(1-eta)*(1-zeta)
        p(5)=0.125*(1-xi)*(1-eta)*(1+zeta)
        p(6)=0.125*(1-xi)*(1+eta)*(1+zeta)
        p(7)=0.125*(1+xi)*(1+eta)*(1+zeta)
        p(8)=0.125*(1+xi)*(1-eta)*(1+zeta)
C.........................................................

     endif
     return
     end
c=======================================================c


c
      subroutine deriv
     1    (iel  ,ig   ,jg   ,kg, p    ,del  ,b ,ncn ,da ,cg ,node,
      dimension node(maxel,27),cord(maxnp,3)
c



         do 6010 j=1,3
      do 6010 l=1,3
      gash=0.0

      do 6020 k=1,ncn

      nn=iabs(node(iel,k))


6020  gash=gash + del(j,k)*cord(nn,l)

         cj(j,l)=gash

6010  continue



      detj =  cj(1,1)*cj(2,2)*cj(3,3)+cj(2,1)*cj(3,2)*cj(1,3)
     1      + cj(1,2)*cj(2,3)*cj(3,1)-cj(1,3)*cj(2,2)*cj(3,1)
     2      - cj(1,2)*cj(2,1)*cj(3,3)-cj(2,3)*cj(3,2)*cj(1,1)

         if(detj.le.0.0) then
      write(60,3010) iel,detj
3010  format(1x ,' Error: Zero or Negative Jacobian. ', i6,g20.5)
      stop

c
      endif

      cji(1,1) =  (cj(2,2)*cj(3,3)-cj(3,2)*cj(2,3))  / detj
      cji(1,2) = ((cj(1,2)*cj(3,3)-cj(3,2)*cj(1,3))) / detj
      cji(1,3) =  (cj(1,2)*cj(2,3)-cj(2,2)*cj(1,3))  / detj
      cji(2,1) = ((cj(2,1)*cj(3,3)-cj(3,1)*cj(2,3))) / detj
      cji(2,2) =  (cj(1,1)*cj(3,3)-cj(3,1)*cj(1,3))  / detj
      cji(2,3) = ((cj(1,1)*cj(2,3)-cj(2,1)*cj(1,3))) / detj
      cji(3,1) =  (cj(2,1)*cj(3,2)-cj(3,1)*cj(2,2))  / detj
      cji(3,2) = ((cj(1,1)*cj(3,2)-cj(3,1)*cj(1,2))) / detj
      cji(3,3) =  (cj(1,1)*cj(2,2)-cj(2,1)*cj(1,2))  / detj
      do 6030 l=1,ncn
      b(j,l)=0.0
      do 6030 k=1,3
6030  b(j,l) = b(j,l) + cji(j,k) * del(k,l)

          da= detj*cg(ig)*cg(jg)*cg(kg)
c

      return
      end

c
c============================================================c
c
      subroutine front
     1 (aa   ,rr   ,iel  ,nop  ,maxel,maxst,ldest,kdest,nk   ,maxfr,
     2  eq   ,lhed ,khed ,kpiv ,lpiv ,jmod ,qq   ,pvkol,vel  ,r1   ,
     3  ncod ,bc   ,nopp ,mdf  ,ndn  ,maxdf,nel  ,maxte,ntov ,lcol ,
     4  nell ,ntra, press,icho,c,akf,ak )
c
c     Frontal elimination routine using diagonal pivoting
c
      implicit double precision(a-h,o-z)
      dimension aa   (maxst,maxst) ,rr   (maxst)
      dimension nop  (maxel,maxst)
      dimension ldest(maxst)       ,kdest(maxst)       ,nk   (maxst)
      dimension eq   (maxfr,maxfr) ,lhed (maxfr)       ,khed (maxfr)
      dimension kpiv (maxfr)       ,lpiv (maxfr)
      dimension jmod (maxfr)       ,qq   (maxfr)       ,pvkol(maxfr)
      dimension vel  (maxte)       ,r1   (maxdf)       ,ncod (maxdf)
      dimension bc   (maxdf)       ,nopp (maxdf)       ,mdf  (maxdf)
      dimension ndn  (maxdf)       ,press(maxdf)
c
c     nlp and ndl are the file specifiers for units 60 and 14 respectively
c     c============================================================c

      nlp=60
      nd1=14
c
c     Prefront
c     ========
      nmax=maxfr
      ncrit=20
      if(ntra.eq.0) goto 6040
      nmax = maxfr
      ntra = 0
      ncrit = 20
      lfron = 0
      nlarg = nmax-10
c
c     Find last appeareance of each node
c     ===================================
      nlast = 0
      do 6010 i = 1,ntov
      do 6020 n = 1,nel
      jdn = ndn(n)
      do 6030 l = 1,jdn
      if(nop(n,l).ne.i)go to 6030
      nlast1 = n
      nlast = n
      l1 = l
6030  continue
6020  continue
      if(nlast.eq.0) go to 6010
      nop(nlast,l1) = -nop(nlast,l1)
      nlast = 0
6010  continue
      ntrix = jdn
c
c     Assembly
c     ========
6040  continue
      if(iel.gt.1) go to 6060
      lcol = 0
      do 6050 i = 1,nmax
      do 6050 j = 1,nmax
      eq(j,i) = 0.
6050  continue
6060  nell = nell+1
      n = nell
      jdn = ndn(nell)
      kc = 0
      do 6070 j = 1,jdn
      nn = nop(n,j)
      m = iabs(nn)
      k = nopp(m)
      idf = mdf(m)
      if(nn.lt.0)ii = -ii
      nk(kc) = ii
6070  continue
c
c     Set up heading vectors
c     ======================
c
      do 6080 lk = 1,kc
      node = nk(lk)
      if(lcol.eq.0)goto 6100
      do 6090 l = 1,lcol
      ll = l
      if(iabs(node).eq.iabs(lhed(l)))go to 6110
6090  continue
6100  lcol = lcol+1
      ldest(lk) = lcol
      lhed(lcol) = node
      go to 6080
6110  ldest(lk) = ll
      lhed(ll) = node
6080  continue
      if(lcol.le.nmax)go to 6130
      nerror = 2
      write(nlp,3010)nerror
      stop
6130  continue
      do 6140 l = 1,kc
      ll = ldest(l)
      do 6140 k = 1,kc
      kk = ldest(k)
      eq(kk,ll) = eq(kk,ll)+aa(k,l)
6140  continue
      if(lcol.lt.ncrit.and.nell.lt.nel) return
c
c     Find out which matrix elements are fully assembeled
c     ===================================================
6150  lc = 0
      ir = 0
      do 6160 l = 1,lcol
      kt = lhed(l)
      if(kt.ge.0)go to 6160
      lc = lc+1
      lpiv(lc) = l
      ncod(kro) = 2
      r1(kro) = bc(kro)
6160  continue
c
c     Modify equations with applied boundary conditions
c     ================================================
      if(ir.eq.0)go to 6190
      do 6170 irr = 1,ir
      k = jmod(irr)
      kh = iabs(lhed(k))
      do 6180 l = 1,lcol
      eq(k,l) = 0.
      lh = iabs(lhed(l))
      if(lh.eq.kh)eq(k,l) = 1.
6180  continue
6170  continue
6190  continue
      if(lc.gt.0)go to 6200
      ncrit = ncrit+10
c     write(nlp,3020)ncrit
      if(ncrit.le.nlarg) return
      nerror = 3
      write(nlp,3030)nerror
      stop
6200  continue
c
c     Search for absolute pivot
c     ========================
      pivot = 0.
      do 6210 l = 1,lc
      lpivc = lpiv(l)
      kpivr = lpivc
      piva = eq(kpivr,lpivc)
      if(abs(piva).lt.abs(pivot))go to 6220
      pivot = piva
      lpivco = lpivc
      kpivro = kpivr
6220  continue
6210  continue
      if(pivot.eq.0.0) return
c
c     Normalise pivotal row
c     ====================
c6230 continue

      if(abs(pivot).lt.0.1d-28) write(nlp,3050)
      do 6240 l = 1,lcol
      qq(l) = eq(kpivro,l)/pivot
6240  continue
      rhs = r1(kro)/pivot
      r1(kro) = rhs
      pvkol(kpivro) = pivot
c
c     Eliminate then delete pivotal row and column
c     ==========================================
      if(kpivro.eq.1)go to 6300
      kpivr = kpivro-1
      do 6250 k = 1,kpivr
      krw = iabs(lhed(k))
      fac = eq(k,lpivco)
      pvkol(k) = fac
      if(lpivco.eq.1.or.fac.eq.0.)go to 6270
      lpivc = lpivco-1
      do 6260 l = 1,lpivc
      eq(k,l) = eq(k,l)-fac*qq(l)
6260  continue
6270  if(lpivco.eq.lcol)go to 6290
      lpivc = lpivco+1
      do 6280 l = lpivc,lcol
      eq(k,l-1) = eq(k,l)-fac*qq(l)
6280  continue
6290  r1(krw) = r1(krw)-fac*rhs
6250  continue
6300  if(kpivro.eq.lcol)go to 6360
      kpivr = kpivro+1
      do 6310 k = kpivr,lcol
      krw = iabs(lhed(k))
      fac = eq(k,lpivco)
      pvkol(k) = fac
      if(lpivco.eq.1)go to 6330
      lpivc = lpivco-1
      do 6320 l = 1,lpivc
      eq(k-1,l) = eq(k,l)-fac*qq(l)
6320  continue
6330  if(lpivco.eq.lcol)go to 6350
      lpivc = lpivco+1
```

```
6310  continue
6360  continue
c
c     Write pivotal equation on disc
c     ================================
      write(nd1) kro,lcol,lpivco,(lhed(l),qq(l),l = 1,lcol)
      do 6370 l = 1,lcol
      eq(l,lcol) = 0.
      eq(lcol,l) = 0.
6370  continue
c
c     Rearrange heading vectors
c     ========================
      lcol = lcol-1
      if(lpivco.eq.lcol+1)go to 6390
      do 6380 l = lpivco,lcol
      lhed(l) = lhed(l+1)
6380  continue
6390  continue
c
c     Determine whether to assemble,eliminate,or backsubstitute
c     ========================================================
      if(lcol.gt.ncrit)go to 6150
      if(nell.lt.nel) return
      if(lcol.gt.1)go to 6150
      lco = iabs(lhed(1))
      kpivro = 1
      pivot = eq(1,1)
      kro = lco
      lpivco = 1
      qq(1) = 1.

c     if(nit.eq.0.or.npra.eq.0)go to 6400
c     write(nlp,3040)lco,kro,pivot

      if(abs(pivot).lt.1d-28)go to 6410

c6400 continue

      r1(kro) = r1(kro)/pivot
      write(nd1) kro,lcol,lpivco,lhed(1),qq(1)
c
c     start back-substitution
c     ======================
c     main exit with solution
c     ======================
6410  continue
c
3010  format(/'  nerror=',i5//
     1 '  the difference nmax-ncrit is not sufficiently large'
     1/'  to permit the assembly of the next element---'
     1/'  either increase nmax or lower ncrit'
     1/)
c3020  format('  frontwidth value=',i4)
3030  format(/'  nerror=',i5//
     1 '  there are no more rows fully summed,this may be due to---'
     1/'  (1)incorrect coding of nop or nk arrays'
     1/'  (2)incorrect value of ncrit. increase ncrit to permit'
     1/'     whole front to be assembled'
     1/)

c3040  format(13h pivotal row=,i4,16h pivotal column=,i4,7h pivot=,e20.10
c    1)

3050  format('  warning-matrix singular or ill conditioned')

      return
      end
c
c
===============================================================
===
      subroutine bacsub
     1    (ntotl,ifix ,vfix ,rhs  ,soln ,soln1, mfrnt,rwork,iwork,idv2,
     2     icho )
c
c
      implicit double precision(a-h,o-z)
      dimension ifix (ntotl),vfix (ntotl),rhs (ntotl),soln (ntotl)
      dimension rwork(mfrnt) ,iwork(mfrnt) ,soln1(ntotl)
c
c
c
                  do 6010 ipos=1,ntotl
                  soln(ipos) =0.0
                  if(ifix(ipos).ne.0) soln(ipos)=vfix(ipos)
6010                 continue
```

```
c
     backspace idv2
     read(idv2)     ipos,ifrnt,jfrnt,(iwork(k),rwork(k),k=1,ifrnt)
     backspace idv2
c
     if(ifix(ipos).ne.0)  go to 6020
c
     ww          = 0.0
     rwork(jfrnt) = 0.0
c
                    do 6030 k=1,ifrnt
                    jpos=iabs(iwork(k))
                    ww  =ww - rwork(k)*soln(jpos)
6030                   continue
c
     soln (ipos)=rhs(ipos)+ww

6020  continue

           if (icho .eq. 2) goto 6050

     do  6040 ipos  = ((3*ntotl)/4)+1 , ntotl
              j       = ipos -((3*ntotl)/4)
             soln1(j) = soln(ipos)

6040  continue

6050    continue

     return
     end
c

c==============================================================c
c     Stokes Solution based on UVW-P scheme via Taylor-Galerkin
c     time stepping method.
c==============================================================c

     subroutine flowStokes(node ,cord ,pmat ,nopp ,mdf ,ndn ,ncod ,
         1bc ,vel  ,press, r1, temp,ldest,kdest,nk ,eq ,lhed ,khed ,kpiv ,lpiv,
    2jmod, qq, pvkol, iter ,nel ,ncn , ngaus,grav1,
    3grav2, grav3, p, del, b, ntrix, maxel, maxnp, maxst, maxfr, maxdf,
    4ndim ,aa ,xg ,da ,ntov ,num, icord, rr, iel, del1,deltat,alpha,
    5idv4,sinv, icho, nnp, tref)
     dimension node (maxel,maxst),pmat (maxel,   9),cord (maxnp, ndim)
     dimension ncod (maxdf      ),bc   (maxdf     ),sinv (maxel,   27)
     dimension vel  (maxnp ,   3),r1   (maxdf     ),conc (maxnp       )
     dimension aa   (maxst,maxst),rr   (maxst     ),ldest(maxst       )
     dimension xg   (          3),cg   (         3),kdest(maxst       )
     dimension x    (          3),v    (         3),nk   (maxst       )
     dimension bicn (          2),hh   (         3)
     dimension p    (         20),del  (   3,   20),b    (   3,    20)
     dimension eq   (maxfr,maxfr),nopp (maxdf     )
     dimension ldsc (         22)
     dimension lhed (maxfr      ),khed (maxfr     ),jmod (maxfr       )
     dimension lpiv (maxfr      ),kpiv (maxfr     ),qq   (maxfr       )
     dimension pvkol(maxfr      ),mdf  (maxdf     ),ndn  (maxdf       )
     dimension ppp  (20    ,  20),pp   (20        )
         dimension ak   (100,100)
         dimension akf  (100    )
         dimension NQ   (20    ,  20),NP   (3    ,   4)
         dimension C    (maxst      ),temp (maxnp     )
         dimension DEL1 (3          )
         dimension press(maxnp      ),clump(maxnp     ),SHAPE1D(3        )
         dimension gdsf (    3,   20)
         dimension dmass(100, 100)
         dimension aa01 (maxst,maxst)
         dimension aa02 (maxst,maxst)
         dimension ak01 (maxst,maxst)
         dimension ak02 (maxst,maxst)



                                      rvisc = pmat(iel,1)
                                      rpef  = pmat(iel,2)
                                      power = pmat(iel,3)
                                      rtem  = pmat(iel,4)
                                      tbco  = pmat(iel,5)
                                      taco  = pmat(iel,6)
                                      roden = pmat(iel,8)

 gamad = pmat(iel,9)


     velsound = 1500.0
         beta     =    0.0
```

```
            akf(idf)      = 0.0
            C  (idf)      = 0.0
           do 6010 jdf= 1,ntrix

             aa   (idf,jdf)=0.0
                  dmass(idf,jdf)=0.0
             ak   (idf,jdf)=0.0
             aa01 (idf,jdf) = 0.0
             aa02 (idf,jdf)= 0.0
             ak01 (idf,jdf)= 0.0
             ak02 (idf,jdf)= 0.0
6010            continue


         if (ncn==4) then
         call gausspt(ngaus,xg,cg,ncn)
         else if (ncn==8) then
    call gaussp(ngaus,xg,cg,ncn)
         end if

              lg=0
    do 6020 ig=1,ngaus
                        g = xg(ig)
    do 6020 jg=1,ngaus
                        h = xg(jg)
    do 6020 kg=1,ngaus
                        f = xg(kg)

                                              lg = lg + 1


    if(iter.eq.1) then


    call shape (g,h,f,p,del,ncn)

    call deriv (iel,ig,jg,kg,p,del,b,ncn,da,cg,node,cord,
   1         maxel,maxnp)

                  iig=ig
                  jjg=jg
              kkg=kg
```

```
      endif
c
c      calculation of viscosity based on the constitutive equation.
c
      spress = 0.0
      stemp  = 0.0

                  do 5333 ip = 1,ncn
            jp    = iabs(node(iel,ip))
                  stemp = stemp + temp(jp) * p(ip)
 5333             continue
            epsii = 1.d-10
            gamad = sinv(iel,lg)
            if(gamad.lt.epsii) gamad = epsii

      call visca
          1(rvisc,power,visc,stemp,rtem,tbco,spress,rpef,taco,gamad)

                  do 6050 idff= 1,3
                  x(idff)     = 0.0
                  v(idff)     = 0.0
                  hh(idff)    = 0.0
6050                  continue
        do 6060 icn = 1 ,ncn
              jcn = iabs(node(iel,icn))
        do 6060 idff= 1 ,  3
        x(idff)     = x(idff) + p(icn)*cord(jcn,idff)
        v(idff)     = v(idff) + p(icn)*vel (jcn,idff)
6060              continue


      if(icord.eq.1) then
c
c      modify da for axisymmetric computations.
c
      da = da * x(1)
      endif


c      column index
```

```
                   j13= i + 2*ncn
               j14= i + 3*ncn

      do 6070 j=1,ncn
               j21= j
               j22= j + ncn
                   j23= j + 2*ncn
                   j24= j + 3*ncn


c         if (iel.le.3000) then


c     Dicretized form of 3D Stokes Equation
c
c --- Stiffness Matrix of Left Hand Side ------------------------------------------------
c
c     For Transient state (Cartesian co-ordinate system)


      aa(j11,j21)=aa(j11,j21) +  p(i)*p(j)*da
           1                   +  (alpha*deltat*2.0*(visc/roden)
           2                   +  0.5*alpha*alpha*deltat*deltat*velsound*velsound)
           3                   *  (b(1,i)*b(1,j)*da)
     3                    +  alpha*deltat*(visc/roden)
     4                    * ((b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)


      aa(j11,j22)=aa(j11,j22) +  alpha*deltat*(visc/roden)
           1                                        *  (b(2,i)*b(1,j)*da)
           2                   +  0.5*alpha*alpha*deltat*deltat*velsound
           3                                        *  velsound*(b(1,i)*b(2,j)*da)

      aa(j11,j23)=aa(j11,j23) +  alpha*deltat*(visc/roden)
           1                                        *  (b(3,i)*b(1,j)*da)
           2                   +  0.5*alpha*alpha*deltat*deltat*velsound
           3                                        *  velsound*(b(1,i)*b(3,j)*da)

      aa(j11,j24)=aa(j11,j24) +  alpha*deltat*(b(1,j)
           1                      *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~
```

```
        3                                   *  velsound*(b(2,i)*b(1,j)*da)

    aa(j12,j22)=aa(j12,j22) +  p(i)*p(j)*da
        1               +  (alpha*deltat*2.0*(visc/roden)
        2                                   +
0.5*alpha*alpha*deltat*deltat*velsound*velsound)
        3                                   *  (b(2,i)*b(2,j)*da)
    3               +  alpha*deltat*(visc/roden)
    4               * ((b(1,i)*b(1,j)+b(3,i)*b(3,j))*da)


    aa(j12,j23)=aa(j12,j23) +  alpha*deltat*(visc/roden)
        1                                   *  (b(3,i)*b(2,j)*da)
        2               +  0.5*alpha*alpha*deltat*deltat*velsound
        3                                   *  velsound*(b(2,i)*b(3,j)*da)

    aa(j12,j24)=aa(j12,j24) +  alpha*deltat*(b(2,j)
        1                 *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

    aa(j13,j21)=aa(j13,j21) +  alpha*deltat*(visc/roden)
        1                                   *  (b(1,i) *b(3,j)*da)
    2               +  0.5*alpha*alpha*deltat*deltat*velsound
        3                                   *  velsound*(b(3,i)*b(1,j)*da)


    aa(j13,j22)=aa(j13,j22) +  alpha*deltat*(visc/roden)
        1                                   *  (b(2,i) *b(3,j)*da)
    2               +  0.5*alpha*alpha*deltat*deltat*velsound
        3                                   *  velsound*(b(3,i)*b(2,j)*da)

    aa(j13,j23)=aa(j13,j23) +  p(i)*p(j)*da
        1               +  (alpha*deltat*2.0*(visc/roden)
        2                                   +
0.5*alpha*alpha*deltat*deltat*velsound*velsound)
        3                                   *  (b(3,i)*b(3,j)*da)
    3               +  alpha*deltat*(visc/roden)
    4               * ((b(1,i)*b(1,j)+b(2,i)*b(2,j))*da)


    aa(j13,j24)=aa(j13,j24) +  alpha*deltat*(b(3,j)
```

```
     aa(j14,j21)=aa(j14,j21) +  alpha*deltat*velsound*velsound
      1                   *  (p(i)*b(1,j)*da)


     aa(j14,j22)=aa(j14,j22) +  alpha*deltat*velsound*velsound
      1                   *  (p(i)*b(2,j)*da)


     aa(j14,j23)=aa(j14,j23) +  alpha*deltat*velsound*velsound
      1                   *  (p(i)*b(3,j)*da)


     aa(j14,j24)=aa(j14,j24) +  p(i)*p(j)*da
      1                   +  (0.5*alpha*alpha*deltat*deltat
      2                   *  velsound*velsound)*((b(1,i)*b(1,j)
     3                   +  b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)

c --- Matrix on Right Hand Side -------------------------------------------------
c
c    For Transient State (Cartesian co-ordinate system)


     ak(j11,j21)=ak(j11,j21) +  p(i)*p(j)*da
      1                   -  ((1.0-alpha)*deltat*2.0*(visc/roden)
      2                                        +  0.5*alpha*(1.0-
alpha)*deltat*deltat*velsound*velsound)
      3                                        *  (b(1,i)*b(1,j)*da)
     3                   -  (1.0-alpha)*deltat*(visc/roden)
     4                   * ((b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)


     ak(j11,j22)=ak(j11,j22) -  (1.0-alpha)*deltat*(visc/roden)
      1                                        *  (b(2,i)*b(1,j)*da)
      2                   -  0.5*alpha*(1.0-alpha)*deltat*deltat*velsound
      3                                        *  velsound*(b(1,i)*b(2,j)*da)

     ak(j11,j23)=ak(j11,j23) -  (1.0-alpha)*deltat*(visc/roden)
      1                                        *  (b(3,i)*b(1,j)*da)
      2                   -  0.5*alpha*(1.0-alpha)*deltat*deltat*velsound
      3                                        *  velsound*(b(1,i)*b(3,j)*da)


     ak(j11,j24)=ak(j11,j24) -  (1.0-alpha)*deltat*(b(1,j)
     ak(j12,j21)=ak(j12,j21) -  (1.0-alpha)*deltat*(visc/roden)
        1                                          *  (b(1,i) *b(2,j)*da)
   2                  -  0.5*alpha*(1.0-alpha)*deltat*deltat
        3                                          *
velsound*velsound*(b(2,i)*b(1,j)*da)


     ak(j12,j22)=ak(j12,j22) +  p(i)*p(j)*da
        1                  -  ((1.0-alpha)*deltat*2.0*(visc/roden)
        2                                          +  0.5*alpha*(1.0-
alpha)*deltat*deltat*velsound*velsound)
        3                                          *  (b(2,i)*b(2,j)*da)
   3                  -  (1.0-alpha)*deltat*(visc/roden)
   4                  * ((b(1,i)*b(1,j)+b(3,i)*b(3,j))*da)


     ak(j12,j23)=ak(j12,j23) -  (1.0-alpha)*deltat*(visc/roden)
        1                                          *  (b(3,i)*b(2,j)*da)
        2                  -  0.5*alpha*(1.0-alpha)*deltat*deltat*velsound
        3                                          *  velsound*(b(2,i)*b(3,j)*da)


     ak(j12,j24)=ak(j12,j24) -  (1.0-alpha)*deltat*(b(2,j)
        1                  *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

     ak(j13,j21)=ak(j13,j21) -  (1.0-alpha)*deltat*(visc/roden)
        1                                          *  (b(1,i) *b(3,j)*da)
   2                  -  0.5*alpha*(1.0-alpha)*deltat*deltat
        3                                          *
velsound*velsound*(b(3,i)*b(1,j)*da)

     ak(j13,j22)=ak(j13,j22) -  (1.0-alpha)*deltat*(visc/roden)
        1                                          *  (b(2,i) *b(3,j)*da)
   2                  -  0.5*alpha*(1.0-alpha)*deltat*deltat
        3                                          *
velsound*velsound*(b(3,i)*b(2,j)*da)



     ak(j13,j23)=ak(j13,j23) +  p(i)*p(j)*da
```

```
    3                    -  (1.0-alpha)*deltat*(visc/roden)
    4                    * ((b(1,i)*b(1,j)+b(2,i)*b(2,j))*da)


     ak(j13,j24)=ak(j13,j24) -  (1.0-alpha)*deltat*(b(3,j)
          1                    *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

     ak(j14,j21)=ak(j14,j21) -  (1.0-alpha)*deltat*velsound*velsound
          1                  *  (p(i)*b(1,j)*da)



     ak(j14,j22)=ak(j14,j22) -  (1.0-alpha)*deltat*velsound*velsound
          1                  *  (p(i)*b(2,j)*da)


     ak(j14,j23)=ak(j14,j23) -  (1.0-alpha)*deltat*velsound*velsound
          1                  *  (p(i)*b(3,j)*da)


     ak(j14,j24)=ak(j13,j24) +  p(i)*p(j)*da
          1                  -  (0.5*alpha*(1.0-alpha)*deltat*deltat
          2                  *  velsound*velsound)*((b(1,i)*b(1,j)
    3                    +  b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)


c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~
c     end if

c
c    Body Force Effect (for Elemental Load Vector Calculation)....
c    ============================================================
c
         C(j11) =C(j11) + (1.0-alpha)*deltat*p(j)*grav1*da

     C(j12) =C(j12) + (1.0-alpha)*deltat*p(j)*grav2*da

     C(j13) =C(j13) + (1.0-alpha)*deltat*p(j)*grav3*da
```

```
6070  continue

6020  continue


c     Writing the Stiffness Matrix for Element Number 14

        If (iel==14) then

          write (idv4,3045)
          write (idv4,3050) iter
      write (idv4,3055) ((aa(i,j),j=1,ntrix),i=1,ntrix)

3045  format(///,' ',10('*'),' element stiffness matrix for element
     114',10('*'),///)
3050  format(///,' ','Iteration number =',i5,///)
3055  format(27(E15.8,3x))

        end if




c     For Transient State (Cartesian Co-ordinate System)
c     ==================================================
c     Term one on RHS is evaluated
c     ===========================

      do 6080 i=1,ncn

                        j11= i
              j12= i + ncn
                   j13= i + 2*ncn
                   j14= i + 3*ncn

      do 6080 j=1,ncn
              j21= j
              j22= j + ncn
                   j23= j + 2*ncn
                   j24= j + 3*ncn

      nn=iabs(node(iel,j))
```

```
   3              ak(j11,j24)*press(nn)


        akf(j12)=akf(j12) + ak(j12,j21)*vel(nn,1) +
   1               ak(j12,j22)*vel(nn,2) +
   2              ak(j12,j23)*vel(nn,3) +
   3              ak(j12,j24)*press(nn)


        akf(j13)=akf(j13) + ak(j13,j21)*vel(nn,1) +
   1               ak(j13,j22)*vel(nn,2) +
   2              ak(j13,j23)*vel(nn,3) +
   3              ak(j13,j24)*press(nn)


        akf(j14)=akf(j14) + ak(j14,j21)*vel(nn,1) +
   1               ak(j14,j22)*vel(nn,2) +
   2               ak(j14,j23)*vel(nn,3) +
   3              ak(j14,j24)*press(nn)

6080  continue

c
c   Evaluation of Elemental Load Vector
c   ==================================

      do 6085 i=1,ncn

                         j11= i
               j12= i + ncn
                    j13= i + 2*ncn
                    j14= i + 3*ncn

c    For Transient State  (Cartesian Co-ordinate System)

          rr(j11)= rr(j11) + akf(j11) + C(j11)
      rr(j12)= rr(j12) + akf(j12) + C(j12)
          rr(j13)= rr(j13) + akf(j13) + C(j13)
          rr(j14)= rr(j14) + akf(j14) + C(j14)

6085  continue
```

```
     2,eq   ,lhed ,khed ,kpiv ,lpiv ,jmod ,qq   ,pvkol,vel  ,r1
     3,ncod ,bc   ,nopp ,mdf  ,ndn  ,maxdf,nel  ,maxte,ntov ,lcol
     4,nell ,ntra, press,icho )
c
      return
      end

c
==================================================================
===
c     Stokes Solution based on the modified UVW-P scheme via
c     Taylor-Galerkin time stepping method.
c
==================================================================
===

      subroutine flowStokes2(node ,cord ,pmat ,nopp ,mdf ,ndn ,ncod ,
           1bc ,vel  ,press, r1, temp,ldest,kdest,nk ,eq ,lhed ,khed ,kpiv ,lpiv,
     2jmod, qq, pvkol, iter ,nel ,ncn , ngaus,grav1,
     3grav2, grav3, p, del, b, ntrix, maxel, maxnp, maxst, maxfr, maxdf,
     4ndim ,aa ,xg ,da ,ntov ,num, icord, rr, iel, del1,deltat,alpha,
     5idv4,sinv, icho, nnp, tref)



      implicit double precision(a-h,o-z)

      dimension node (maxel,maxst),pmat (maxel,   9),cord (maxnp, ndim)
      dimension ncod (maxdf      ),bc   (maxdf     ),sinv (maxel,   27)
      dimension vel  (maxnp ,   3),r1   (maxdf     ),conc (maxnp      )
      dimension aa   (maxst,maxst),rr   (maxst     ),ldest(maxst      )
      dimension xg   (          3),cg   (          3),kdest(maxst      )
      dimension x    (          3),v    (          3),nk   (maxst      )
      dimension bicn (          2),hh   (          3)
      dimension p    (         20),del  (   3,   20),b    (   3,    20)
      dimension eq   (maxfr,maxfr),nopp (maxdf     )
      dimension ldsc (         22)
      dimension lhed (maxfr      ),khed (maxfr     ),jmod (maxfr      )
      dimension lpiv (maxfr      ),kpiv (maxfr     ),qq   (maxfr      )
      dimension pvkol(maxfr      ),mdf  (maxdf     ),ndn  (maxdf      )
      dimension ppp  (20    ,  20),pp   (20        )
           dimension ak   (100,100)
      dimension gdsf (    3,   20)
      dimension dmass(100, 100)
      dimension aa01 (maxst,maxst)
      dimension aa02 (maxst,maxst)
      dimension ak01 (maxst,maxst)
      dimension ak02 (maxst,maxst)



   rvisc = pmat(iel,1)
   rpef  = pmat(iel,2)
   power = pmat(iel,3)
   rtem  = pmat(iel,4)
   tbco  = pmat(iel,5)
   taco  = pmat(iel,6)
   roden = pmat(iel,8)
      gamad = pmat(iel,9)


   velsound = 1500.0
      beta     =    0.0

    lambda   =  10E2
c     lambda   = (10E7-10E8)/visc    ! From Zienkienwicz
c


   do 6010 idf= 1,ntrix
         rr (idf)      = 0.0
         akf(idf)      = 0.0
            C  (idf)      = 0.0
                     do 6010 jdf= 1,ntrix

         aa   (idf,jdf)=0.0
            dmass(idf,jdf)=0.0
         ak   (idf,jdf)=0.0
         aa01 (idf,jdf) = 0.0
         aa02 (idf,jdf)= 0.0
         ak01 (idf,jdf)= 0.0
         ak02 (idf,jdf)= 0.0
6010          continue


      if (ncn==4) then
```

```
        lg=0
       do 6020 ig=1,ngaus
                          g = xg(ig)
       do 6020 jg=1,ngaus
                          h = xg(jg)
       do 6020 kg=1,ngaus
                          f = xg(kg)


                                                          lg = lg + 1


       if(iter.eq.1) then


       call shape (g,h,f,p,del,ncn)

       call deriv (iel,ig,jg,kg,p,del,b,ncn,da,cg,node,cord,
      1        maxel,maxnp)

                     iig=ig
                     jjg=jg
                  kkg=kg

                  write(15)   iel ,ig ,jg ,kg, p ,del ,b ,da
       else
       if(.not. EOF(15))read(15) iel,iig,jjg,kkg,p ,del ,b , da

       endif
c
c     calculation of viscosity based on the constitutive equation.
c
       spress = 0.0
       stemp  = 0.0

                   do 5333 ip = 1,ncn
                jp    = iabs(node(iel,ip))
                   stemp = stemp + temp(jp) * p(ip)
5333           continue
             epsii = 1.d-10
              gamad = sinv(iel,lg)
             if(gamad.lt.epsii) gamad = epsii
```

```
c     preparation of the convective acceleration terms/balancing
c     dissipation is used
c
                  do 6050 idff= 1,3
                  x(idff)     = 0.0
                  v(idff)     = 0.0
                  hh(idff)    = 0.0
6050                  continue
        do 6060 icn = 1 ,ncn
              jcn = iabs(node(iel,icn))
        do 6060 idff= 1 ,  3
        x(idff)     = x(idff) + p(icn)*cord(jcn,idff)
        v(idff)     = v(idff) + p(icn)*vel (jcn,idff)
6060                continue

      if(icord.eq.1) then
c
c     modify da for axisymmetric computations.
c
      da = da * x(1)
      endif


c     column index

      do 6070 i=1,ncn

                            j11= i
                 j12= i + ncn
                      j13= i + 2*ncn
                 j14= i + 3*ncn

      do 6070 j=1,ncn
                 j21= j
                 j22= j + ncn
                      j23= j + 2*ncn
                      j24= j + 3*ncn




c     Dicretized form of 3D Stokes Equation
```

```
    aa(j11,j21)=aa(j11,j21) +  p(i)*p(j)*da
         1                 +  (alpha*deltat*2.0*(visc/roden)
         2                 +  0.5*lambda*alpha*alpha*deltat*deltat*velsound*velsound)
         3                                           *  (b(1,i)*b(1,j)*da)
    3                  +  alpha*deltat*(visc/roden)
    4                  * ((b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)


    aa(j11,j22)=aa(j11,j22) +  alpha*deltat*(visc/roden)
         1                                           *  (b(2,i)*b(1,j)*da)
         2              +  0.5*lambda*alpha*alpha*deltat*deltat*velsound
         3                                           *  velsound*(b(1,i)*b(2,j)*da)

    aa(j11,j23)=aa(j11,j23) +  alpha*deltat*(visc/roden)
         1                                           *  (b(3,i)*b(1,j)*da)
         2             +  0.5*lambda*alpha*alpha*deltat*deltat*velsound
         3                                           *  velsound*(b(1,i)*b(3,j)*da)

    aa(j11,j24)=aa(j11,j24) +  alpha*deltat*(b(1,j)
         1                 *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

    aa(j12,j21)=aa(j12,j21) +  alpha*deltat*(visc/roden)
         1                                           *  (b(1,i) *b(2,j)*da)
    2           +  0.5*lambda*alpha*alpha*deltat*deltat*velsound
         3                                           *  velsound*(b(2,i)*b(1,j)*da)

    aa(j12,j22)=aa(j12,j22) +  p(i)*p(j)*da
         1                 +  (alpha*deltat*2.0*(visc/roden)
         2          +  0.5*lambda*alpha*alpha*deltat*deltat*velsound*velsound)
         3                                           *  (b(2,i)*b(2,j)*da)
    3                  +  alpha*deltat*(visc/roden)
    4                  * ((b(1,i)*b(1,j)+b(3,i)*b(3,j))*da)


    aa(j12,j23)=aa(j12,j23) +  alpha*deltat*(visc/roden)
         1                                           *  (b(3,i)*b(2,j)*da)
         2         +  0.5*lambda*alpha*alpha*deltat*deltat*velsound
         3                                           *  velsound*(b(2,i)*b(3,j)*da)

    aa(j12,j24)=aa(j12,j24) +  alpha*deltat*(b(2,j)
```

```
      aa(j13,j21)=aa(j13,j21) +  alpha*deltat*(visc/roden)
     1                                        *  (b(1,i) *b(3,j)*da)
     2         +  0.5*lambda*alpha*alpha*deltat*deltat*velsound
     3                                        *  velsound*(b(3,i)*b(1,j)*da)


      aa(j13,j22)=aa(j13,j22) +  alpha*deltat*(visc/roden)
     1                                        *  (b(2,i) *b(3,j)*da)
     2           +  0.5*lambda*alpha*alpha*deltat*deltat*velsound
     3                                        *  velsound*(b(3,i)*b(2,j)*da)

      aa(j13,j23)=aa(j13,j23) +  p(i)*p(j)*da
     1                  +  (alpha*deltat*2.0*(visc/roden)
     2           +  0.5*lambda*alpha*alpha*deltat*deltat*velsound*velsound)
     3                                        *  (b(3,i)*b(3,j)*da)
     3                 +  alpha*deltat*(visc/roden)
     4                 * ((b(1,i)*b(1,j)+b(2,i)*b(2,j))*da)


      aa(j13,j24)=aa(j13,j24) +  alpha*deltat*(b(3,j)
     1                   *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

      aa(j14,j21)=aa(j14,j21) +  lambda*alpha*deltat*velsound*velsound
     1                  *  (p(i)*b(1,j)*da)


      aa(j14,j22)=aa(j14,j22) +  lambda*alpha*deltat*velsound*velsound
     1                  *  (p(i)*b(2,j)*da)


      aa(j14,j23)=aa(j14,j23) +  lambda*alpha*deltat*velsound*velsound
     1                  *  (p(i)*b(3,j)*da)


      aa(j14,j24)=aa(j14,j24) +  p(i)*p(j)*da
     1                  +  (0.5*lambda*alpha*alpha*deltat*deltat
     2                  *  velsound*velsound)*((b(1,i)*b(1,j)
     3                 +  b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)

c --- Matrix on Right Hand Side -------------------------------------------
c
```

```
      1            -  ((1.0-alpha)*deltat*2.0*(visc/roden)
      2 +  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat*velsound*velsound)
      3                                            *  (b(1,i)*b(1,j)*da)
   3               -  (1.0-alpha)*deltat*(visc/roden)
   4               * ((b(2,i)*b(2,j)+b(3,i)*b(3,j))*da)


   ak(j11,j22)=ak(j11,j22) -  (1.0-alpha)*deltat*(visc/roden)
      1                                            *  (b(2,i)*b(1,j)*da)
      2     -  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat*velsound
      3                                            *  velsound*(b(1,i)*b(2,j)*da)

   ak(j11,j23)=ak(j11,j23) -  (1.0-alpha)*deltat*(visc/roden)
      1                                            *  (b(3,i)*b(1,j)*da)
      2      -  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat*velsound
      3                                            *  velsound*(b(1,i)*b(3,j)*da)


   ak(j11,j24)=ak(j11,j24) -  (1.0-alpha)*deltat*(b(1,j)
      1                 *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~

   ak(j12,j21)=ak(j12,j21) -  (1.0-alpha)*deltat*(visc/roden)
      1                                            *  (b(1,i) *b(2,j)*da)
   2              -  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat
      3                                            *
velsound*velsound*(b(2,i)*b(1,j)*da)


   ak(j12,j22)=ak(j12,j22) +  p(i)*p(j)*da
      1               -  ((1.0-alpha)*deltat*2.0*(visc/roden)
      2        +  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat*velsound*velsound)
      3                                            *  (b(2,i)*b(2,j)*da)
   3               -  (1.0-alpha)*deltat*(visc/roden)
   4               * ((b(1,i)*b(1,j)+b(3,i)*b(3,j))*da)


   ak(j12,j23)=ak(j12,j23) -  (1.0-alpha)*deltat*(visc/roden)
      1                                            *  (b(3,i)*b(2,j)*da)
      2      -  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat*velsound
      3                                            *  velsound*(b(2,i)*b(3,j)*da)
```

```
ak(j13,j21)=ak(j13,j21) -  (1.0-alpha)*deltat*(visc/roden)
        1                                            *  (b(1,i) *b(3,j)*da)
    2                 -  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat
        3                                            *
velsound*velsound*(b(3,i)*b(1,j)*da)

    ak(j13,j22)=ak(j13,j22) -  (1.0-alpha)*deltat*(visc/roden)
        1                                            *  (b(2,i) *b(3,j)*da)
    2                 -  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat
        3                                            *
velsound*velsound*(b(3,i)*b(2,j)*da)



    ak(j13,j23)=ak(j13,j23) +  p(i)*p(j)*da
        1                  -  ((1.0-alpha)*deltat*2.0*(visc/roden)
        2 +  0.5*lambda*alpha*(1.0-alpha)*deltat*deltat*velsound*velsound)
        3                                            *  (b(3,i)*b(3,j)*da)
    3                   -  (1.0-alpha)*deltat*(visc/roden)
    4                   * ((b(1,i)*b(1,j)+b(2,i)*b(2,j))*da)


    ak(j13,j24)=ak(j13,j24) -  (1.0-alpha)*deltat*(b(3,j)
        1                    *p(i)*da)

c~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~

    ak(j14,j21)=ak(j14,j21) -  lambda*(1.0-alpha)*deltat*velsound
        1                  *velsound * (p(i)*b(1,j)*da)



    ak(j14,j22)=ak(j14,j22) -  lambda*(1.0-alpha)*deltat*velsound
        1                  *  velsound * (p(i)*b(2,j)*da)


    ak(j14,j23)=ak(j14,j23) -  lambda*(1.0-alpha)*deltat*velsound
        1                  *  velsound * (p(i)*b(3,j)*da)


    ak(j14,j24)=ak(j13,j24) +  p(i)*p(j)*da
```

```
c
c    Body Force Effect (for Elemental Load Vector Calculation)
c    ============================================================c
c
         C(j11) =C(j11) + (1.0-alpha)*deltat*p(j)*grav1*da

     C(j12) =C(j12) + (1.0-alpha)*deltat*p(j)*grav2*da

     C(j13) =C(j13) + (1.0-alpha)*deltat*p(j)*grav3*da

     C(j14) =C(j14) + 0



6070  continue

6020  continue


c    Writing the Stiffness Matrix for Element Number 14

       If (iel==14) then

         write (idv4,3045)
         write (idv4,3050) iter
     write (idv4,3055) ((aa(i,j),j=1,ntrix),i=1,ntrix)

3045  format(///,' ',10('*'),' element stiffness matrix for element
    114',10('*'),///)
3050  format(///,' ','Iteration number =',i5,///)
3055  format(27(E15.8,3x))

       end if



c    For Transient State (Cartesian Co-ordinate System)
c    ====================================================
c    Term one on RHS is evaluated
c    ============================
                j12= i + ncn
                     j13= i + 2*ncn
                     j14= i + 3*ncn

      do 6080 j=1,ncn
                j21= j
                j22= j + ncn
                     j23= j + 2*ncn
                     j24= j + 3*ncn

      nn=iabs(node(iel,j))


           akf(j11)=akf(j11) + ak(j11,j21)*vel(nn,1) +
     1                 ak(j11,j22)*vel(nn,2) +
     2                 ak(j11,j23)*vel(nn,3) +
     3                 ak(j11,j24)*press(nn)


           akf(j12)=akf(j12) + ak(j12,j21)*vel(nn,1) +
     1                  ak(j12,j22)*vel(nn,2) +
     2                 ak(j12,j23)*vel(nn,3) +
     3                 ak(j12,j24)*press(nn)


           akf(j13)=akf(j13) + ak(j13,j21)*vel(nn,1) +
     1                  ak(j13,j22)*vel(nn,2) +
     2                 ak(j13,j23)*vel(nn,3) +
     3                 ak(j13,j24)*press(nn)


           akf(j14)=akf(j14) + ak(j14,j21)*vel(nn,1) +
     1                  ak(j14,j22)*vel(nn,2) +
     2                  ak(j14,j23)*vel(nn,3) +
     3                 ak(j14,j24)*press(nn)

 6080  continue

c
c    Evaluation of Elemental Load Vector
c    ===================================

      do 6085 i=1,ncn
```

```
c     For Transient State  (Cartesian Co-ordinate System) hnt

          rr(j11)= rr(j11) + akf(j11) + C(j11)
      rr(j12)= rr(j12) + akf(j12) + C(j12)
          rr(j13)= rr(j13) + akf(j13) + C(j13)
          rr(j14)= rr(j14) + akf(j14) + C(j14)

6085  continue


c
      maxte=maxdf
      call front
     1(aa   ,rr   ,iel  ,node ,maxel,maxst,ldest,kdest,nk   ,maxfr
     2,eq   ,lhed ,khed ,kpiv ,lpiv ,jmod ,qq   ,pvkol,vel  ,r1
     3,ncod ,bc   ,nopp ,mdf  ,ndn  ,maxdf,nel  ,maxte,ntov ,lcol
     4,nell ,ntra, press,icho )
c
      return
      end


c
=============================================================
===
c
      subroutine concn
     1    (node ,cord ,pmat ,nopp ,mdf  ,ndn  ,ncod ,bc   ,conc ,vel
     2    ,r1   ,xg   ,ndim ,da   ,ldest,kdest,nk   ,eq   ,lhed
     3    ,khed ,kpiv ,lpiv ,jmod ,qq   ,pvkol,iter ,nel  ,ncn  ,ntov
     4    ,icord,ngaus, p    ,del  ,b    ,ntrix,maxel,maxnp
     5    ,maxst,maxfr,maxdf,num  ,ijmo ,ae
     6    ,re   ,cg   ,iel, del1  )
c
      implicit double precision(a-h,o-z)
c
      dimension node (maxel,maxst),cord (maxnp, ndim)
      dimension ncod (maxdf      ),bc   (maxdf      )
      dimension conc (maxnp      ),r1   (maxdf      ),vel  (maxnp,    3)
      dimension ae   (maxst,maxst),re   (maxst      )
      dimension p    (         20),del  (   3,    20),b    (   3,    20)
      dimension x    (          3),v    (          3)
      dimension bicn (          3),bjcn (          3)
      dimension khed (maxfr      ),kpiv (maxfr      ),lpiv (maxfr      )
      dimension jmod (maxfr      ),qq   (maxfr      ),pvkol(maxfr      )
      dimension mdf  (maxdf      ),ndn  (maxdf      )
          dimension ppp  (20    ,     20),pp   (20           )
          dimension kae  (maxst,maxst),kaef (maxst       )
          dimension NQ   (4    ,    3),NP   (3    ,    4)
          dimension C    (maxst      )
      dimension DEL1 (3           )
c
c
                                          rvisc = pmat(iel,1)
                                          rbulk = pmat(iel,2)
                                          power = pmat(iel,3)
                                          rtem  = pmat(iel,4)
                                          tco   = pmat(iel,5)
                                          roden = pmat(iel,6)
                                                                                          dispc
= pmat(iel,7)
c
c     Basic element loop
c     ==================
                              do 6010 itrix  = 1 ,maxst
                              re(itrix)      = 0.0
                              do 6010 jtrix  = 1 ,maxst
                              ae(itrix,jtrix)= 0.0
6010                            continue
c
c     Numerical integration
c     =====================
c
      call gaussp(ngaus,xg,cg)
c
                lg=0

      do 6020 ig=1,ngaus
                                  g = xg(ig)
      do 6020 jg=1,ngaus
                                  h = xg(jg)
      do 6020 kg=1,ngaus
                                  f = xg(kg)
         lg=lg+1
                                     if(iter.eq.1) then
```

```
    end if
c
c    Read shape functions and their cartesian derivatives data from
c    a work file

        read (15)   iiel,iig,jjg,kkg,p,del,b,da

    ijmo=ijmo+1
c
c    Coefficients evaluated at integration point
c
                    do 6040 idf = 1 , 3
                    x(idf)      = 0.0
                    v(idf)      = 0.0
6040                   continue
       do 6050 icn = 1 ,ncn
             jcn = iabs(node(iel,icn))
       do 6050 idf = 1 , 2
          x(idf) = x(idf) + p(icn)*cord(jcn,idf)
          v(idf) = v(idf) + p(icn)*vel (jcn,idf)
6050     continue
c
    if(icord.eq.1) then
c
c   Modify da for axisymmetric computations.
c   =========================================
    da = da * x(1)

    endif

    do 6060 icn = 1 ,ncn

c    Row index
c    =========
                              ir = icn
    do 6060 jcn = 1 ,ncn

c    Column index
c    ============
                          ic = jcn

6060  continue
6020  continue
```

```
    do 6080 icn = 1 , ncn

                          ir = icn

    do 6080 jcn = 1 , ncn

                                  ic = jcn


        KAEF(ir)=KAEF(ir) + kae(ir,ic)*conc(ic)

6080  continue

c     Term two on RHS i.e. Calculation of Line Integrals
c     ===================================================

    NQ(1,3)=2
    NQ(1,2)=5
    NQ(1,1)=1
    NQ(2,3)=3
    NQ(2,2)=6
    NQ(2,1)=2
    NQ(3,3)=4
    NQ(3,2)=7
    NQ(3,1)=3
    NQ(4,3)=1
    NQ(4,2)=8
    NQ(4,1)=4

        do I=1,4
          do J=1,3
            NP(4-J,1)=node(iel,NQ(I,4-J))
          end do
    end do

        do 6096 L=1,4
                call gaussp(ngaus,xg,cg)

                    do 6096 ig=1,ngaus
                                      g = xg(ig)
                                      if(iter.eq.1) then
```

```
              end if

                        dsix=0.0
                        dsiy=0.0


                        do I=1,ngaus
                          dsix=dsix+p(I)*cord(NP(I,L),1)
                               dsiy=dsiy+p(I)*cord(NP(I,L),2)
                        end do

                              concx=0.0
               concy=0.0

                   do I=1,ngaus

               concx   = concx  + DEL1(I)*conc(NP(I,L))
           concy   = concy  + DEL1(I)*conc(NP(I,L))
           concz   = concz  + DEL1(I)*conc(NP(I,L))
                              u11              = u11    + DEL1(I)*vel (NP(I,L),1)
           u12                = u12    + DEL1(I)*vel (NP(I,L),1)
           u21                = u21    + DEL1(I)*vel (NP(I,L),2)
           u22                = u22    + DEL1(I)*vel (NP(I,L),2)
                              velx    = velx   + DEL1(I)*vel (NP(I,L),1)
                              vely    = vely   + DEL1(I)*vel (NP(I,L),2)

                        end do


c       ellgth = sqrt(dsix**2 + dsiy**2+dsiz**2)

               dcell    =       dsix/ellgth
               dcelm  =        dsiy/ellgth
c              dceln   =      dsiz/ellgth
               dnx      s        =        dcelm
          dny            =        -dcell
    dnz       =     dceln
          ellgth  =   2.0*ellgth
               djacob =   ellgth/2.0


        do 6087 icn=1,ncn

                         ir = icn
```

```
c     Calculation of the Elemental Load Vector
c     ==========================================

      do 6085 icn =1 , ncn

                          ir = icn

      re(ir)  = re(ir) + kaef(ir) + C(ir)

 6085 continue


c     Solve equations
c     ===============

      maxte=maxnp
      call front
     1(ae   ,re   ,iel   ,node ,maxel,maxst,ldest,kdest,nk   ,maxfr
     2,eq   ,lhed ,khed  ,kpiv ,lpiv ,jmod ,qq   ,pvkol, conc ,r1
     3,ncod ,bc   ,nopp  ,mdf  ,ndn  ,maxdf,nel  ,maxte,ntov ,lcol
     4,nell ,ntra ,press,icho,c,akf,ak)
c
c     End of basic element loop
c     =========================

      return
      end
c


c
=================================================================
===
      subroutine stress
     1     (nel,nnp,ncn ,node ,p , b , da ,vel  ,maxnp, maxel,  maxst ,
     2      maxdf, stres, press, rvisc ,clump ,ngaus  )
c
      implicit double precision(a-h,o-z)
c
c function
c --------
c     calculates stress components at integration points,
      dimension clump(maxnp      )
c
                                        rewind 15
c
                                 do 4990 inp   =1,maxnp
                                 do 4990 icp   =1, 6
                                 stres(inp,icp)= 0.0
 4990                               continue
c

      do 5000 iel = 1 ,nel
c
      do 6010 ig=1,ngaus

      do 6010 jg=1,ngaus

      do 6010 kg=1,ngaus

       if(.not. EOF(15))read(15) iiel,iig,jjg,kkg,p ,del ,b , da

                                u11 = 0.0
            u12 = 0.0
            u13 = 0.0
            u21 = 0.0
            u22 = 0.0
            u23 = 0.0
            u31 = 0.0
            u32 = 0.0
            u33 = 0.0
               pres1 = 0.0

        do 6020 icn = 1 ,ncn
              jcn = iabs(node(iel,icn))
            u11 = u11 + b(1,icn)*vel(jcn,1)
            u12 = u12 + b(2,icn)*vel(jcn,1)
            u13 = u13 + b(3,icn)*vel(jcn,1)
            u21 = u21 + b(1,icn)*vel(jcn,2)
            u22 = u22 + b(2,icn)*vel(jcn,2)
            u23 = u23 + b(3,icn)*vel(jcn,2)
            u31 = u31 + b(1,icn)*vel(jcn,3)
            u32 = u32 + b(2,icn)*vel(jcn,3)
            u33 = u33 + b(3,icn)*vel(jcn,3)
                        pres1 = pres1 + p(icn)*press(jcn)
```

```
c    cartesian components of the stress tensor
c    ==========================================
c    Shear Stress (Tau)
c    ==========================================

     sd11 = 2.0   *rvisc *  u11
     sd22 = 2.0   *rvisc *  u22
          sd33 = 2.0   *rvisc *  u33
     sd12 = rvisc * (u12 +  u21)
     sd13 = rvisc * (u13 +  u31)
     sd23 = rvisc * (u23 +  u32)

c    ==========================================
c    Normal Stress (Pi)
c    ==========================================

     s11  =-pres1 + sd11
     s22  =-pres1 + sd22
     s33  =-pres1 + sd33
     s12  = sd12
     s13  = sd13
     s23  = sd23

c
=================================================================
=
c *** calculate stress at nodal points
c
=================================================================
=


       do 6500 icn = 1 ,ncn
                      jcn = iabs(node(iel,icn))

              stres(jcn,1)= stres(jcn,1)
     1                  + p(icn) *s11  *da
              stres(jcn,2)= stres(jcn,2)
     1                  + p(icn) *s22  *da
              stres(jcn,3)= stres(jcn,3)
     1                  + p(icn) *s33  *da
              stres(jcn,4)= stres(jcn,4)
     1                  + p(icn) *s12  *da
              stres(jcn,5)= stres(jcn,5)
```

```
 6500              continue

c
 6010 continue

 5000 continue

     return
c
     end
c
c
==============================================================
===

     subroutine lumpm
    1    (clump,nnp  ,maxnp,nel  ,ngaus,p   ,del  ,b   ,maxst,
    2     node ,maxel,ncn  )
c
     implicit double precision(a-h,o-z)
     dimension b    (    3,   20) ,del  (   3,   20) ,p   (  20)
     dimension clump(maxnp      )
     dimension node (maxel,maxst)
     dimension pp   (ncn  ,ncn  )
c
                              do 5000 inp = 1 ,nnp
                              clump  (inp)= 0.0
 5000                             continue
c
            rewind 15
c
     do 5010 iel = 1 ,nel

           do 5020           ig  = 1 ,ngaus
           do 5020           jg  = 1 ,ngaus
           do 5020           kg  = 1 ,ngaus


          if(.not. EOF(15)) read (15) jel ,iig ,jjg ,kkg ,p ,del ,b ,da

                  do 5030    icn = 1 ,ncn
                            ww = 0.0
                  do 5040    jcn = 1 ,ncn
```

```
5030            continue

 5020       continue
 5010       continue

      return
      end
c
c
=================================================================
===

      subroutine getnod (nnp  ,cord ,idv1 ,idv2 ,maxnp,ndim,icord)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     nnp    total number of nodal points in the mesh
c     cord   array for nodal coordinates
c     idv1   input device id.
c     idv2   output device id.
c     ndim    see below
c
      dimension cord(maxnp, ndim)
c
      if (.NOT. EOF(51)) read (idv1,1010)
           1(jnp ,(cord(jnp,idf),idf=1,3),jnp=1,nnp)
      if(icord.eq.0) write(idv2,3010)
      if(icord.eq.1) write(idv2,3020)
      write(idv2,3030) (jnp ,(cord(jnp,idf),idf=1,3) ,jnp=1,nnp)
c
      return
c
1010  format(i8,e20.12,e20.12,e20.12)
3010  format(' ',///' ',20('*'),' nodal coordinates ',20('*'),//
     1' ',(7x,'id.',13x,'x-coord',13x,'y-coord',13x,'z-coord',13x)/)
3020  format(' ',///' ',20('*'),' nodal coordinates ',20('*'),//
     1' ',2(7x,'id/',7x,'r-coord',7x,'z-coord',20x)/)
3030  format(' ',i10,10x,f10.6,10x,f10.6,10x,f10.6)
c
      end
c
c============================================================c
```

```
      subroutine getelm (nel  ,ncn  ,node ,idv1 ,idv2 ,maxel)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     ncn    number of nodes per element
c     node   array for element connectivity data
c     idv1   input device id.
c     idv2   output device id.
c     maxel  see below
c
      dimension node (maxel, ncn)
c
            do 6010 iel = 1 ,nel
6010  if (.not. eof(51))read (idv1,1010) iel ,(node(iel,icn),icn=1,ncn)
      print*, "nodal connectivity array read"
      write(idv2,3010)
             do 6020 jel = 1 ,nel
6020  write(idv2,3020) jel ,(node(jel,icn),icn=1,ncn)
c
      return
c

1010  format(21i7)
3010  format(' ',///,' ',20('*'),' element connectivity ',20('*'),//
     1' ',4x,'id.',7x,'n o d a l - p o i n t  e n t r i e s',/)
3020  format(21i7)
c
      end
c
c
==============================================================
===

      subroutine getbcd (nbc  ,ibc  ,jbc  ,vbc
     1                 ,idv1 ,idv2 ,maxbc)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     nbc    number of nodal constraint data
c     ibc    array for constrained nodal points
```

```
c     maxbc  see below
c
      dimension ibc  (maxbc) ,jbc  (maxbc),vbc  (maxbc)
c
      if (.not. eof(51))read (idv1,1010) (ibc(ind) ,jbc(ind) ,vbc(ind)
           1    ,ind=1,nbc)
           print*, "boundary conditions array read"
      write(idv2,3010)
      write(idv2,3020) (ibc(ind) ,jbc(ind) ,vbc(ind) ,ind=1,nbc)
c
      return
c
1010  format(2i5,f10.4)
3010  format(' ',//  /,' ',20('*'),' nodal constraint ',20('*'),//
     1' ',(8x,'id.',7x,'dof',10x,'value',10x)/)
3020  format(5x,i5,5x,i5,f17.4)
c
      end
c
c
=================================================================
==

      subroutine putbcv
     1 (nnp  ,nbc ,ibc ,jbc ,vbc ,ncod ,bc ,maxbc,maxdf,maxel,maxst,
     2  node)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     ncod    array for constraint switch defined for every d.o.f.
c     bc      array for storing contraint value
c     maxbc   see below
c     maxdf   see below
c
      dimension ibc  (maxbc) ,jbc  (maxbc) ,vbc  (maxbc)
      dimension ncod (maxdf) ,bc   (maxdf) ,node (maxel,maxst)
c
      do 6010 ind = 1 ,nbc
      if(jbc(ind)>4) goto 6010
      jnd         = ibc(ind)+(jbc(ind)-1)*nnp
      bc    (jnd) = vbc(ind)
```

```
c         iel=16
c         inp=24
c         kc=iabs(node(iel,inp))
c
      return
      end
c


      subroutine putbcc
     1    (nbc  ,ibc  ,jbc  ,vbc  ,ncod ,bc   ,maxbc,maxdf)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     arguments same as subroutine putbcv
c
      dimension ibc  (maxbc) ,jbc  (maxbc) ,vbc  (maxbc)
      dimension ncod (maxdf) ,bc   (maxdf)
c
      do 6010 ind = 1 ,nbc
      if(jbc(ind).eq.5) then
      jnd         = ibc(ind)
      bc     (jnd)= vbc(ind)
      ncod   (jnd)= 1
      endif
6010  continue
c
      return
      end
c
c
=================================================================
===

      subroutine putbcs
     1    (nnp  ,nbc  ,ibc  ,jbc  ,vbc  ,ncod ,bc   ,maxbc,maxdf)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     arguments same as subroutine putbcv
```

```
      do 6010 ind = 1 ,nbc
      if(jbc(ind).eq.6) then
      jnd        = ibc(ind)
      bc     (jnd)= vbc(ind)
      ncod   (jnd)= 1
      endif
6010  continue
c
      return
      end
c
c
=================================================================
===
      subroutine clean
     1    (ncn  ,nel  ,ndf  ,node ,r1    ,maxel,maxst,maxdf,
     2     bc   ,ncod ,icho )
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     all arguments are defined elsewhere.
c
      dimension r1    (maxdf) ,node(maxel,maxst)
      dimension bc    (maxdf) ,ncod(maxdf      )
c
c     function
c     ========
c     cleans the used arrays and makes them ready for  solution
c
          do  6010 i   = 1,maxdf
                r1(i) = 0.0
                bc(i) = 0.0
              ncod(i) = 0
6010               continue
              ntrix   = ndf *ncn
          do  6020 iel = 1,nel
          do  6020 inp = 1,ntrix
          node(iel,inp) = iabs(node(iel,inp))
6020               continue
          if(icho.ne.1)then
          do 6030 iel = 1,nel
```

```
         endif
c
     return
     end
c
c
===============================================================
===
     subroutine setprm
    1    (nnp  ,nel  ,ncn  ,node ,ndf  ,maxel,maxst,ndn  ,ntrix,
    2     maxdf,ntov ,mdf  ,nopp )
c
     implicit double precision(a-h,o-z)
c
c    arguments
c    =========
c    all arguments are defined elsewhere.
c
     dimension node (maxel,maxst), ndn  (maxdf)
     dimension mdf  (maxdf      ), nopp (maxdf)
c
c    function
c    ========
c    Sets the location data for nodal degrees of freedom
c
     do 6010 iel = 1 ,nel
       ndn(iel) = ntrix
     do 6010 icn = 1 ,ncn
           kcn = node(iel,icn)
                        jacn= icn+(ndf-3)*ncn
           lacn= kcn+(ndf-3)*nnp

                        jbcn= icn+(ndf-2)*ncn
           lbcn= kcn+(ndf-2)*nnp

                        jccn= icn+(ndf-1)*ncn
           lccn= kcn+(ndf-1)*nnp

     node(iel,jacn) = lacn
     node(iel,jbcn) = lbcn
     node(iel,jccn) = lccn

6010  continue
```

```
6020  continue
c
      return
      end
c
c
=================================================================
===
      subroutine getmat (nel ,nmat,pmat, idv1, idv2,maxel,rtem, rpef)
c
      implicit double precision(a-h,o-z)
c
c     arguments
c     =========
c     nmat  number of materials
c     pmat  array for material constants for each element
c     idv1 input device id.
c     idv2  output devide id.
c     maxel see below
c
      dimension pmat (maxel,   9)
c
      write(idv2,3010)
c
      do 6010 imat = 1 ,nmat
      if (.NOT. EOF(51)) read(idv1,1010) rvisc, power, tref, tbco, taco,
          1                              dispc, pref, roden, gamad
      print*, "material properties read"
CCCC              ifrom = 1
CCCC              ito   = nel

                              if(rtem .eq.0.) rtem  = 0.001
              if(rpef .eq.0.) rpef  = 0.001

          do 6020 iel  = 1 ,nel
        pmat(iel,1) = rvisc
        pmat(iel,2) = pref
        pmat(iel,3) = power
        pmat(iel,4) = tref
        pmat(iel,5) = tbco
        pmat(iel,6) = taco
        pmat(iel,7) = dispc
            pmat(iel,8) = roden
        pmat(iel,9) = gamad
```

```
c     rvisc    mu nought; consistency coefficient
c     pref     reference pressure
c     power    power law index
c     tref     reference temperature
c     tbco     coefficient b in the power law model
c     taco     coefficient a in the power law model
c     dispc    dispersion coefficient
c     gamad    shear rate
c
6020  continue
c
      write(idv2,3020) imat ,ifrom ,ito ,rvisc ,power
      write(idv2,3030)
      write(idv2,3040) tref ,tbco, pref, taco
      write(idv2,3050)
      write(idv2,3060) dispc , roden , gamad

6010  continue
c
      return
c
1010  format(9d10.5)
c
3010  format(' ',//' ',35('*'),' material properties ',35('*'),//
     1     ' ',2x,'id.',5x,'eid.(from-to)',3x,'consistency co-efficient'
     2,5x,'power law index',/)
3020  format(' ',i3,i12,i4,5x,g15.5,15x,g15.5)
3030  format(/x,' reference temperature   coefficient b
     1   reference pressure  coefficient a '/)
3040  format(f16.3,f22.4,6x,g10.3,9x,g10.3)
3050  format(/x,
     1'Dispersion Coefficient     Density          Shear rate'/)
3060  format(g13.3,15x,g7.1,6x,g16.5)
c
      end
c
c
==================================================================
===
c
      subroutine contol
     1(vel  ,conc ,iter ,ntov ,nnp  ,maxnp,maxdf,errov,errop,vet ,cet,
     2 pet, press)
```

```
      dimension vel  (maxdf),conc (maxnp), press(maxnp)
      dimension vet  (maxdf),cet  (maxnp), pet  (maxnp)

          errv  = 0.0
      torv  = 0.0
      errc  = 0.0
      torc  = 0.0
          errp  = 0.0
          torp  = 0.0
c
c    calculate difference between velocities in consecutive iterations
c
=====================================================================
==
c
                do 6010 icheck = 1,ntov
      if(iter.eq.1) vet(icheck) = 0.0
C061008                    errv = errv +
C061008     1          (vel(icheck)-vet(icheck)) * (vel(icheck)-vet(icheck))

       errv = errv +
     1       vel(icheck)*vel(icheck)- 2*vel(icheck)*vet(icheck)
     2      +vet(icheck)*vet(icheck)
       torv = torv + vel(icheck)*vel(icheck)
c
      vet(icheck) = vel(icheck)
c
6010              continue
                errov= errv/torv
c
c    calculate difference between concentrations in consecutive iterations
c
=====================================================================
======
                do 6020 icheck = 1,nnp
      if(iter.eq.1)  cet(icheck) = 0.0
       errc = errc +
     1      (conc(icheck)*conc(icheck))-2*conc(icheck)*cet(icheck)
     2      + cet(icheck)*cet(icheck)
                torc = torc + conc(icheck)*conc(icheck)

      cet(icheck) = conc(icheck)
```

```
                do 6030 icheck = 1,nnp
     if(iter.eq.1)  pet(icheck) = 0.0

c230908           if(iter.eq.1)  press(icheck) = 0.0
      errp = errp +
     1       press(icheck)*press(icheck)-2*press(icheck)*pet(icheck)
     2     + pet(icheck)*pet(icheck)
      torp = torp + press(icheck)*press(icheck)
c
           pet(icheck) = press(icheck)
c
6030               continue
                errop= errp/torp

c
     return
     end

c
============================================================
===

     subroutine output
     1     (nnp  ,vel  ,press, maxdf,maxnp,icord, stres)
c
     implicit double precision(a-h,o-z)
c
c    arguments are already defined
c    ============================
     dimension vel(maxdf), press(maxnp), conc(maxnp)
          dimension stres(maxnp,    6)

          write(60,3010)

     if(icord.eq.0) write(60,3020)
     if(icord.eq.1) write(60,3030)

          do 6010 inp = 1,nnp
                    jnp = inp + nnp
                                                knp = inp + (2*nnp)
     press(inp)=press(inp)
     write(60,3040)inp,vel(inp),vel(jnp),vel(knp),press(inp),
          1stres(inp,1)
      call minimax
          1( cmax  , pmax  , vel  , conc  ,  press ,  maxnp,  nnp  ,    nc,
          2  np    , nm    , ncm  , nvxm  ,  nvym  ,  nvzm ,
     3  nvxl  , nvyl  , nvzl , pmin  ,  cmin  ,
     4  vxmax , vxmin , vymax, vymin ,  vzmax ,  vzmin,  ndim , maxdf )

          write(60,3045)
          write(60,3050)nvxm,vel(nvxm),nvxl,vel(nvxl)

          write(60,3055)
          write(60,3060)nvym,vel(nnp+nvym),nvyl,vel(nnp+nvyl)

          write(60,3065)
          write(60,3070)nvzm,vel(2*nnp+nvzm),nvzl,vel(2*nnp+nvzl)

          write(60,3075)
          write(60,3080)np,press(np),nm,press(nm)


3010  format(/' nodal velocities and  pressures  '/)
3020  format('  id.     ux        uy         uz     press  stress'/)
3030  format('  id.     ur        uz         uz            press'/)
3040  format(i5,3e13.4,e22.8,g15.5)

3045  format('node no.        max ux    node no.        min ux')
3050  format(i5,e22.8,i5,e22.8,/)

3055  format('node no.        max uy    node no.        min uy')
3060  format(i5,e22.8,i5,e22.8,/)

3065  format('node no.        max uz    node no.        min uz')
3070  format(i5,e22.8,i5,e22.8,/)

3075  format('node no.        max p     node no.        min p')
3080  format(i5,e22.8,i5,e22.8,/)

      return
      end
c
c
==============================================================
===
c    The subroutine slip introduces the slip wall boundary conditions
```

```
        subroutine slip (ldsc)

      implicit double precision(a-h,o-z)

      dimension ldsc  (22        )


C
c     Channel depth data for slip wall b.c. & friction c.
c     ====================================================

      read(50,1010)rfrct
1010  format(f10.0)
c
c     This loop is used for identifying upper slip layer
c     ===================================================

      i1=0
      do 6030 ids=1,nel-13,14
        do 6030  j=1,2
        i1=i1+1
        ldsc(i1)=ids+j-1
6030  continue
c
c     This loop is used for identifying lower slip layers
c     ====================================================

      i1=0
      do 6020 ids=13,nel-2,14
        do 6020 j=1,2
        i1=i1+1
        ldsc(i1)=ids+j-1
6020  continue
c
c
      return
      end

c
==============================================================
===

      subroutine hgstvl ( cmax, pmax , conc  ,  press , maxnp,
      cmax= conc(1)
      pmax= press (1)
      pmin= press (1)
      nc=1
      np=1
      nm=1
      do i=2,nnp
        cm= conc(i)
        pm= press (i)
        pi= press (i)
        if ( cm.gt.cmax ) then
          cmax=cm
          nc  =i
        endif
        if ( pm.gt.pmax ) then
          pmax=pm
          np  =i
        endif
        if ( pi.lt.pmin ) then
          pmin = pi
          nm   = i
        endif
      enddo
      return
      end
c
c
=================================================================
===
c     write nodal outputs                          .
c
      subroutine output2 ( nnp , vel  , conc , press , maxdf, maxnp,
     1                  time , nwr  , iter , errov , erroc, errop)

      implicit real*8 (a-h,o-z)
      dimension vel   ( maxdf) , conc  ( maxnp) , press ( maxnp)
      dimension nwr (10) ,  vr  (10) , pv(10) , cr (10)

          call hgstvl ( cmax, pmax , conc  ,  press , maxnp,
     1                  nnp , nc   , np   , nm   , pmin     )

      write ( 2 , 5111) iter,errov,erroc,errop

      write ( 2 , 5115 ) time
      do 24  i=1,nnp
        vres= dsqrt (vel(i+i-1)**2+vel(i+i)**2)
        psee= press(i)

        write (2,5130) i,vel(i+i-1),vel(i+i),vres,conc(i),press(i)

 24   continue

      call stress
     1     (nel,nnp,ncn ,node ,p , b , da ,vel  ,maxnp, maxel,  maxst ,
     2      maxdf, stres, press, rvisc ,clump ,ngaus  )
c
c     write the stress components
c     ==========================

      write ( 2 , 5133 )
      write ( 2 , 5130 ) i,   sd11,  sd12,  sd22

c     writing of output results
c     ========================

        do k=1,3
          i      = nwr(k)
          vr (k) = dsqrt (vel(i+i-1)**2+vel(i+i)**2)
          pv(k)  = press(i)
          cr (k) = conc (i)
        enddo
        write ( 4 , 5125 ) time , (vr(i),pv(i),cr(i),i=1,3)
c.....................................................................
 5111 format (1x,/,'-  solution after',i5,'  iteration(s) -',
     1        1x,/,'-  error oval ( velocity ) =',f20.9,
     2        1x,/,'-  error oval ( concentration ) =',f20.9,
     3        1x,/,'-  error oval ( pressure ) =',f20.9)
 5112 format (1x,'  maximum concentration = ',g20.5,'  at node =',i5)
 5113 format (1x,'  maximum pressure    = ',g20.5,'  at node =',i5)
 5114 format (1x,'  minimum pressure    = ',g20.5,'  at node =',i5)
 5115 format (1x,'solution at time = ',g20.5,/)
 5120 format (1x,//,' result ( node no. ,vx, vy, |v|, concentration,
     1 pressure)',/)
 5125 format (1x,e11.6,3(' |',3e12.4))
 5130 format (1x,i4,2x,5(d11.5,2x))
 5133 format (1x,//,1x,' sd11    , sd12   , sd22 ',/)
      return
      end
```

```
c     This subroutine calculate the viscosity using the power law model
c
      subroutine visca
     1         (rvisc,power,visc,stemp,rtem,tbco,spress,rpef,taco
     2         ,gamad )

      implicit double precision(a-h,o-z)

c230908        visc   = rvisc*(4.0*gamad**((power-1.0)*0.5))
c230908        1            *exp(-tbco*(stemp-rtem))

c110909        visc = rvisc*(4.0*gamad**((power-1.0)*0.5))

            visc = rvisc*(gamad**((power-1.0)))

      return
      end
c
c
====================================================================
===

      subroutine lagsh1 ( xi , shape1d , del1 )

      implicit real*8 (a-h,o-z)
      dimension shape1d(3), del1(3)
        shape1d(1)   = -0.5*xi*(1.0-xi)
        shape1d(2)   = (1.0+xi)*(1.0-xi)
        shape1d(3)   =  0.5*g*(1.0+xi)
        del1(1)  = -0.5+xi
        del1(2)  = -2.0*xi
        del1(3)  =  0.5+xi
      return
      end


c
====================================================================
===                          .

      subroutine minimax
          1( cmax  , pmax  , vel  , conc  ,  press ,  maxnp,  nnp  ,    nc,
          2  np    , nm    , ncm  , nvxm  ,  nvym  ,  nvzm ,
      dimension conc  ( maxnp )  , vel  (maxdf)
      dimension press ( maxnp )

c
      vxmax      =        vel(1)
          vxmin  =        vel(1)
          vymax =         vel(nnp+1)
          vymin  =        vel(nnp+1)
          vzmax =         vel(2*nnp+1)
          vzmin  =        vel(2*nnp+1)

      pmax       =        press (1)
      pmin       =        press (1)


          nc              =       1
          ncm             =       1
      np         =        1
      nm         =        1
          nvxm            =       1
          nvym            =       1
          nvzm            =       1
          nvxl            =       1
          nvyl            =       1
          nvzl            =       1

          do 6020 i=2,nnp

      pm         =        press (i)
      pi         =    press (i)
          vxmx    =   vel(i)
          vxmn    =   vel(i)
          vymx    =   vel(nnp+i)
          vymn    =   vel(nnp+i)
          vzmx    =   vel(2*nnp+i)
          vzmn    =   vel(2*nnp+i)

        if (   pm.gt.pmax  ) then
          pmax=pm
          np  =i
        endif
        if (   pi.lt.pmin  ) then
          pmin = pi
          nvxm  = i
        endif
        if ( vymx.gt.vymax ) then
          vymax= vymx
          nvym  = i
        endif
        if ( vzmx.gt.vzmax ) then
          vzmax= vzmx
          nvzm  = i
        endif

            if ( vxmn.lt.vxmin ) then
          vxmin= vxmn
          nvxl  = i
        endif
        if ( vymn.lt.vymin ) then
          vymin= vymn
          nvyl  = i
        endif
        if ( vzmn.lt.vzmin ) then
          vzmin= vzmn
          nvzl  = i
        endif
 6020 continue

          return
      end

c
c
=============================================================
c
      subroutine secinv
     1   (nel  ,nnp  ,ncn  ,ngaus,node ,sinv ,cord ,p  ,b ,
     2    del  ,da   ,vel  ,maxnp,maxel,maxst,ndim ,icord,
     3    maxdf,num)
c
      implicit double precision(a-h,o-z)
c
c function
c --------
c     calculates the second invariant of rate of deformation
c     tensor at integration points.
c
```

```
      dimension b    (    3,   20)
c
                                  rewind 15

      do 5000 iel= 1 , nel
             lg = 0
      do 5010 ig = 1 ,ngaus
      do 5010 jg = 1 ,ngaus
           do 5010 kg = 1 ,ngaus

             lg = lg+1
c
          read (15) iiel,iig,jjg,p,del,b,da
c
               u11 = 0.0
               u12 = 0.0
                                        u13 = 0.0
               u21 = 0.0
               u22 = 0.0
                    u23 = 0.0
                    u31 = 0.0
                    u32 = 0.0
                    u33 = 0.0

         do 5020 icn = 1 ,ncn
                             jcn = iabs(node(iel,icn))
c                              mcn = jcn + nnp
c                                                                  kcn = jcn + (2*nnp)

c *** components of the rate of deformation tensor

               u11 = u11 + b(1,icn)*vel(jcn,1)
               u12 = u12 + b(2,icn)*vel(jcn,1)
                                        u13 = u13 + b(3,icn)*vel(jcn,1)
               u21 = u21 + b(1,icn)*vel(jcn,2)
               u22 = u22 + b(2,icn)*vel(jcn,2)
                                        u23 = u23 + b(3,icn)*vel(jcn,2)
                                        u31 = u31 + b(1,icn)*vel(jcn,3)
               u32 = u32 + b(2,icn)*vel(jcn,3)
                                        u33 = u33 + b(3,icn)*vel(jcn,3)

 5020     continue
c
```

```
        2                               (u13+u31)*(u13+u31)+
    3           (u21+u12)*(u21+u12)+
    4           (u22+u22)*(u22+u22)+
        5                               (u23+u32)*(u23+u32)+
        6                               (u31+u13)*(u31+u13)+
    7           (u32+u23)*(u32+u23)+
        8                               (u33+u33)*(u33+u33))

 5010 continue
 5000 continue

      return
      end

c=========================================================c


          subroutine gausspt(ngaus,xg,cg,ncn)
c
      implicit double precision(a-h,o-z)
c
c      x(g)   specifies the coordinates of the Gauss points
c      c(g)   specifies the Gauss weights
c
      dimension xg(5,5),cg(3)
          Real:: al, be
          al=0.58541020
          be=0.13819660

          xg=0.0
          cg=0.0

          if (ngaus==1) then
          xg(1,1)= 1.0/4.0
          xg(1,2)= xg(1,1)
          xg(1,3)= xg(1,1)
          xg(1,4)= xg(1,1)
          cg(1)=1
          else if (ngauss==2)then
          xg(1,1)=al
          xg(1,2)=be
          xg(1,3)=be
          xg(1,4)=be
          xg(3,1)=be
          xg(3,2)=be
          xg(3,3)=al
          xg(3,4)=be
          xg(4,1)=be
          xg(4,2)=be
          xg(4,3)=be
          xg(4,4)=al
          cg(1)=1.0/4.0
          cg(2)=cg(1)
          else if (ngauss==3) then
          xg(1,1)=1.0/4.0
          xg(1,2)=1.0/4.0
          xg(1,3)=1.0/4.0
          xg(1,4)=1.0/4.0

          xg(2,1)=1.0/2.0
          xg(2,2)=1.0/6.0
          xg(2,3)=1.0/6.0
          xg(2,4)=1.0/6.0

          xg(3,1)=1.0/6.0
          xg(3,2)=1.0/2.0
          xg(3,3)=1.0/6.0
          xg(3,4)=1.0/6.0

          xg(4,1)=1.0/6.0
          xg(4,2)=1.0/6.0
          xg(4,3)=1.0/2.0
          xg(4,4)=1.0/6.0

          xg(5,1)=1.0/6.0
          xg(5,2)=1.0/6.0
          xg(5,3)=1.0/6.0
          xg(5,4)=1.0/2.0

          end if
          cg(1)=-4.0/5.0
          cg(2)=9.0/20.0
      return
      end

c==========================================================c
```

```
SUBROUTINE GFMFEM

        PARAMETER (maxel  = 250000   )
        PARAMETER (maxnp  = 50000    )

        PARAMETER (maxbc  = 25000    )
        PARAMETER (maxdf  = maxnp*4  )
        PARAMETER (maxst  = 80       )
        PARAMETER (maxfr  = 2000     )
       PARAMETER (ndim   = 3        )

    IMPLICIT double PRECISION(a-h,o-z)


    CHARACTER
CH(150)*1,SF*4,CC*2,FNAME*30,VL*2,VM*2,VN*2,CD*4,CE*5
    CHARACTER C1*2,C2*2,CW*4
        CHARACTER filnam (80)
    DIMENSION NOD(27)
    DIMENSION cord(maxnp, ndim),node (maxel, 27)

    WRITE (*,130 )
130   FORMAT(1X,'Enter GFORM file name    ',$)

    READ (*,135) FNAME
135   FORMAT (A30)

    OPEN (UNIT=1,FILE=FNAME,FORM='FORMATTED')
    OPEN (UNIT=2,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=3,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=4,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=5,FILE='MESH.FEM',FORM='FORMATTED')
    OPEN (UNIT=6,FILE='input.dat',FORM='FORMATTED')
    OPEN (UNIT=7,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=8,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=9,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=10,STATUS='SCRATCH',FORM='FORMATTED')
    OPEN (UNIT=11,STATUS='SCRATCH',FORM='FORMATTED')
```

```
c=============================================================
=====
c     TO SCALE THE GEOMETRY
c=============================================================
=====

1111  PRINT *," Enter the scale factor: any number between 0.5 and 2 "
      READ*, scale
      IF(scale < 0.5 .OR. scale > 2) THEN
      PRINT*, " Invalid scale factor, try again."

      GOTO 1111

      END IF

C============================================================
=======
C    DYNAMIC MEMORY ALLOCATION
C============================================================
=======

    DO WHILE ( IOS.EQ.0 )
      READ (1,'(150A)',ERR=300,END=300,IOSTAT=IOS) (CH(J),J=1,150)
      CC=CH(1)//CH(2)
      CD=CH(1)//CH(2)//CH(3)
      CE=CH(1)//CH(2)//CH(3)//CH(4)//CH(5)

          IF (CC.EQ.'ND')THEN
         WRITE (2,'(100A)') (CH(K),K=4,100)
         WRITE (8,'(100A)') (CH(K),K=4,100)
         NND=NND+1
      ENDIF

          IF (CC.EQ.'EL')THEN
         WRITE (2,'(100A)') (CH(K),K=4,100)
         WRITE (8,'(100A)') (CH(K),K=4,100)
         NEM=NEM+1
      ENDIF


      IF (CD.EQ.'VND') THEN
         WRITE (3,'(50A)') (CH(K),K=5,30)
         WRITE (10,'(150A)') (CH(K),K=7,85)
         NNP=NNP+1
      ENDIF
      IF (CW.EQ.'ND')THEN
         WRITE (7,'(100A)') (CH(K),K=4,100)
         WRITE (11,'(100A)') (CH(K),K=4,100)
         NND=NND+1
      ENDIF
      IF (CW.EQ.'VX')THEN
         WRITE (7,'(100A)') (CH(K),K=4,100)
         WRITE (11,'(100A)') (CH(K),K=4,100)
         NND=NND+1
      ENDIF
      IF (CW.EQ.'VY')THEN
         WRITE (7,'(100A)') (CH(K),K=4,100)
         WRITE (11,'(100A)') (CH(K),K=4,100)
         NND=NND+1
      ENDIF
      IF (CW.EQ.'VZ')THEN
         WRITE (7,'(100A)') (CH(K),K=4,100)
         WRITE (11,'(100A)') (CH(K),K=4,100)
         NND=NND+1
      ENDIF

    ENDDO

300   REWIND 2
    REWIND 3
    REWIND 4
    REWIND 7
    REWIND 8
    REWIND 9
    REWIND 10
    REWIND 11


C==============================================================
====
C      NODAL COORDINATES
C==============================================================
====

    DO I=1,NND
      READ (2,*) N,X,Y,Z
    ENDDO

C010908=========================================================
===

     DO I=1,NND
         READ (7,*,ERR=333,END=333,IOSTAT=IOS) K,VX,VY,VZ,X,Y,Z
         WRITE (11) VX,VY,VZ,X,Y,Z
     ENDDO
333   REWIND(7)

C==========================================================
====
C      ELEMENT CONNECTIVITY
C==========================================================
====

    DO I=1,NEM
      READ (2,*) NNEE,SF,NSF,NPE,(NOD(K),K=1,NPE)


           WRITE (5,120) I,NOD(2),NOD(6),NOD(5),NOD(1),NOD(3),
         1                 NOD(7),NOD(8),NOD(4)

120   FORMAT (9I8)
    ENDDO

C==========================================================
====
C      VELOCITY BOUNDARY CONDITION
C==========================================================
====

    DO J= 1,NNV
      READ (3,1601) ND,VL,CU,V,NT,NP,VM,VN
1601    FORMAT (I7,A2,A1,G20.8,2I5,A2,A3)

C-------------------------------------------------
      IF (NP.EQ.0) THEN
!-----VND,ND,VX,VAL
         IF (VL.EQ.'VX') THEN
               WRITE (5,'(2I5,F10.4)') ND,NCODE1,V
!-----VND,ND,VY,VAL
```

```
   END IF
                 NVN=NVN+1
                 NBC=NBC+1
          ELSE IF (NP.EQ.1) THEN
         IF (VL.EQ.'VX') THEN
!------VND,ND,VX,VAL,ND,NPE,VY,VZ
                 IF((VM.EQ.'VY').AND.(VN.EQ.'VZ'))THEN
                    WRITE (5,'(2I5,F10.4)') ND,NCODE1,V
                    WRITE (5,'(2I5,F10.4)') ND,NCODE2,V
                    WRITE (5,'(2I5,F10.4)') ND,NCODE3,V
                    NVN = NVN + 3
                    NBC = NBC + 3
!------VND,ND,VX,VAL,ND,NPE,VY
                 ELSE IF (VM.EQ.'VY') THEN
                    WRITE (5,'(2I5,F10.4)') ND,NCODE1,V
                    WRITE (5,'(2I5,F10.4)') ND,NCODE2,V
                    NVN = NVN + 2
                    NBC = NBC + 2
!------VND,ND,VX,VAL,ND,NPE,VZ
                 ELSE IF(VM.EQ.'VZ') THEN
                    WRITE (5,'(2I5,F10.4)') ND,NCODE1,V
                    WRITE (5,'(2I5,F10.4)') ND,NCODE3,V
                    NVN = NVN + 2
                    NBC = NBC + 2

        ElSE IF(VL.EQ.'VX')Then
!-----VND,ND,VX,VAL,NPE
              WRITE (5,'(2I5,F10.4)') ND,NCODE1,V
              NVN=NVN+1
              NBC=NBC+1
CCC           END IF
        END IF
!-----VND,VY,VAL,ND,NPE,VZ
      ELSE IF(VL.EQ.'VY'.AND.(VM.EQ.'VZ'))THEN
              WRITE (5,'(2I5,F10.4)') ND,NCODE2,V
              WRITE (5,'(2I5,F10.4)') ND,NCODE3,V
                 NVN = NVN + 2
                 NBC = NBC + 2
          ELSE IF(VL.EQ.'VY')then
              WRITE (5,'(2I5,F10.4)') ND,NCODE2,V
            NVN=NVN+1
                 NBC=NBC+1
      ELSE IF(VL.EQ.'VZ') THEN
            WRITE (5,'(2I5,F10.4)') ND,NCODE3,V
```

```
    END DO

C============================================================
====
C      PRESSURE BOUNDARY CONDITION
C============================================================
====

      DO I=1,NNP
         READ (4,'(I5,G20.8)') PC,VP
         WRITE (5,'(2I5,F10.4)' ) PC,NCODEP,VP
         NBC=NBC+1
         NAT=NAT+1
      END DO

C============================================================
====
C    Data file preparation
C============================================================
====

    ncn=NPE
         ngauss = 3
         nmat=1
         ntep=1
         icord=0
    grav1=0.0
         grav2=0.0
         grav3=0.0
         tolv= 1e-05
         tolp= 1e-05
         tolc= 1e-05
         rvisc = 80.0
         power = 1.23
         tref= 293.0
         tbco = 0.014
         taco = 0.2
         dispc = 0.2
         pref =1.01325e5
         roden = 1000.0

c============================================================
===
c    writing the data file
```

```
        write (6,'(2i5)') ncn,ngauss
        write (6,'(4i5)') NND,NEM,NBC,nmat
        write (6,'(2i5)') ntep,icord
        write (6,'(3f10.3)') grav1, grav2, grav3
        write (6,'(3f10.5)') tolv,tolp,tolc
        write (6,'(9d10.5)') rvisc, power, tref, tbco, taco,
        1            dispc, pref, roden, gamad

C=================================================================
====
C     NODAL COORDINATES
C=================================================================
====

      DO I=1,NND
        READ (8,*) N,X,Y,Z
            X=X/scale
            Y=Y/scale
            Z=Z/scale
            cord(I,1)=X
            cord(I,2)=Y
            cord(I,3)=Z
            WRITE (6,'(I8,3e20.12)') N,X,Y,Z
      ENDDO

C=================================================================
====
C     ELEMENT CONNECTIVITY
C=================================================================
====

      DO I=1,NEM
        READ (8,*) NNEE,SF,NSF,NPE,(node(I,K),K=1,NPE)

        if (NPE==4) then
           WRITE (6,'(5I8)')I,node(I,3),node(I,2),
      1            node(I,1),node(I,4)
        else if (npe==8) then
             WRITE (6,'(21I7)') I,node(I,2),node(I,6),
        2            node(I,5),node(I,1),node(I,3),
      3            node(I,7),node(I,8),node(I,4)
            else if (NPE==20) then
                WRITE (6,'(10I8)') I,NOD(2),NOD(6),NOD(5),
C=================================================================
====
C      VELOCITY BOUNDARY CONDITION
C=================================================================
====

        DO J= 1,NNV

      READ (9,1602) ND,VL,CU,V,NT,NP,VM,VN
1602    FORMAT (I7,A2,A1,G20.8,2I5,A2,A3)

      IF (NP.EQ.0) THEN
        IF (VL.EQ.'VX') THEN
             WRITE (6,'(2I5,F10.4)') ND,NCODE1,V
           ELSE IF (VL.EQ.'VY') THEN
              WRITE (6,'(2I5,F10.4)') ND,NCODE2,V
           ELSE IF (VL.EQ.'VZ') THEN
              WRITE (6,'(2I5,F10.4)') ND,NCODE3,V
           END IF
         ELSE IF (NP.EQ.1) THEN
        IF (VL.EQ.'VX') THEN
              IF((VM.EQ.'VY').AND.(VN.EQ.'VZ'))THEN
                 WRITE (6,'(2I5,F10.4)') ND,NCODE1,V
                 WRITE (6,'(2I5,F10.4)') ND,NCODE2,V
                 WRITE (6,'(2I5,F10.4)') ND,NCODE3,V
              ELSE    IF (VM.EQ.'VY') THEN
                 WRITE (6,'(2I5,F10.4)') ND,NCODE1,V
                 WRITE (6,'(2I5,F10.4)') ND,NCODE2,V
              ELSE IF(VM.EQ.'VZ') THEN
                 WRITE (6,'(2I5,F10.4)') ND,NCODE1,V
                 WRITE (6,'(2I5,F10.4)') ND,NCODE3,V
          ElSE IF(VL.EQ.'VX')Then
                 WRITE (6,'(2I5,F10.4)') ND,NCODE1,V
              END IF
        ELSE IF(VL.EQ.'VY'.AND.(VM.EQ.'VZ'))THEN
              WRITE (6,'(2I5,F10.4)') ND,NCODE2,V
              WRITE (6,'(2I5,F10.4)') ND,NCODE3,V
           ELSE IF(VL.EQ.'VY')then
              WRITE (6,'(2I5,F10.4)') ND,NCODE2,V
        ELSE IF(VL.EQ.'VZ') THEN
           WRITE (6,'(2I5,F10.4)') ND,NCODE3,V
        END IF
```

```
C==============================================================C

       DO I=1,NNP
          READ (10,'(I5,G20.8)') PC,VP
             WRITE (6,'(2I5,F10.4)' ) PC,NCODEP,VP
       END DO

C==============================================================C
C      OUTPUT OF RESULTS
C==============================================================C

        PRINT *,' '
        PRINT *,"The geometry is discretized into a finite element
     1 mesh of: "
        PRINT *,' '
        PRINT *,NND,
     1   "Nodes"
        PRINT *,' '
        PRINT *,NEM,
     2   "Elements"
            PRINT *,"With"
        PRINT *,NNP,
     4   'Applied nodal pressure boundary conditions'
            PRINT *,"And"
        PRINT *, NVN,
     5   "Applied nodal velocity boundary conditions"
        PRINT *,"Giving a"
        PRINT *,NBC,
     3   'total number of applied boundary conditions'

c         PRINT*,' '
c         PRINT*, ' THE SHEAR RATE = ',GAMAD


      CLOSE (1)
      CLOSE (2)
      CLOSE (3)
      CLOSE (4)
      CLOSE (5)
          CLOSE (6)
      CLOSE (11)
          print *,'       '
          print *,'       '
```

```
c==============================================================c
c    This subroutine prepares output data to use for visualization
c    using Cosmos GeoStar software.
c
C
c==============================================================c
      subroutine cosmos
     1      (nnp  ,  vel  ,  press   ,  maxdf  ,  maxnp  , icord   ,
    2       pmat ,  maxel,  actpress ,nel                        )
c
      implicit double precision(a-h,o-z)
c
c    arguments are already defined
c    ============================
      dimension vel (maxdf    ), press   (maxnp)
          dimension pmat(maxel,  9), actpress(maxnp)
          dimension vm      (nnp  )


          open(unit=610   ,   file='cosmGraph',   access='sequential',
     1form='formatted',   status="unknown" ,   iostat=ios        )

          j=0
          k=5

          write(610,3010) nnp, j, k

c     do i = 1, nel
c       roden=pmat(1,8)
c     end do

      DO inp=1,nel
      roden=pmat(inp,8)
      END DO

c     do i=1,nnp
c         j=i
c         vm(j)=sqrt((vel(j,1)**2)+(vel(j,2)**2)+(vel(j,3)**2))
c         end do
```

```
                  knp = inp + (2*nnp)

c         actpress(inp)=roden*press(inp)*-1

      actpress(inp)=roden*press(inp)
      vm(inp)=sqrt((vel(inp)**2)+(vel(jnp)**2)+(vel(knp)**2))
      write(610,3020)inp,vel(inp),vel(jnp),vel(knp),
     1          vm(inp),actpress(inp)

6010  continue

      close (610)

3010  format(3i5)
3020  format(i5,4e13.4,e22.8)
6000  FORMAT(8X,'U',8X,'V',5X,'W',5X, 'M'
     1     8X,'PRESSURE',/)

      return
      end
c=============================================================c
c    This subroutine prepares output data to use for visualization      c
c    using tecplot software.                                             c
c=============================================================c
      subroutine tecplot
          1      (nnp  ,  vel  ,  press   ,  maxdf  ,  maxnp  , icord   ,
     2       pmat ,  maxel,  actpress,  cord   ,  ncn    , nel     ,
     3       node ,  ndim  )
c
      implicit double precision(a-h,o-z)
c
c    arguments are already defined
c    ==============================
      dimension vel     (nnp,3), press (maxnp)     , pmat(maxel ,  9)
          dimension actpress(maxnp), cord  (maxnp,ndim), node(maxel, ncn)
          dimension vm      (nnp  )

          open(unit=614   ,   file='tecpGraph.dat',   access='sequential',
     1form='formatted',   status="unknown" ,   iostat=ios        )

c    Compute the Magnitude of the resultant velocity
c    ================================================
c     Write the Techplot file for post-processing
c     ==========================================
c      nnel=4*nel

      write (614,1000)
           write (614,2000) nnp, nel

      roden=pmat(1,8)

      do i=1,nnp
c          j=i
c          actpress(i)=roden*press(i)* -1
      actpress(i)=roden*press(i)

           write (614,5000) cord(i,1), cord(i,2), cord(i,3),
           1               vel(i,1), vel(i,2), vel(i,3),
     2              vm(i), actpress(i)
      end do


c230908      do i=1,nel
c230908         j=i
c230908         write (614,6000) abs(node(j,1)), abs(node(j,2)), abs(node(j,9))
c230908         1              ,abs(node(j,8))
c230908      write (614,6000) abs(node(j,2)), abs(node(j,3)), abs(node(j,4))
c230908         1              ,abs(node(j,9))
c230908         write (614,6000) abs(node(j,9)), abs(node(j,4)), abs(node(j,5))
c230908         1              ,abs(node(j,6))
c230908         write (614,6000) abs(node(j,8)), abs(node(j,9)), abs(node(j,6))
c230908         1              ,abs(node(j,7))
c230908      end do


c     Elemental connectivity for techplot files
c-----------------------------------------------

      do  i=1,nel
c          j=i
      if (ncn==8)then
           write (614,'(8i8)') abs(node(i,1)), abs(node(i,2)), abs(node(i,6))
           1               ,abs(node(i,5)), abs(node(i,4)), abs(node(i,3))
     2              ,abs(node(i,7)), abs(node(i,8))
```

```
      close (614)

1000  format(/'Variables = "X", "Y","Z","U","V","W","M","P"'/)
c 2000  format(/'ZONE N=',i5,',E=',i5,',F=FEPOINT,ET=QUADRILATERAL'/)


2000  format(/'ZONE N=',i5,',E=',i5,',F=FEPOINT,ET=BRICK'/)

5000  format(3e20.12,3e13.4,e13.4,e22.8)
6000  format(4i8)

      return
      end

c

c    ===========================================================c
c   e  n  d  o  f  p  r  o  g  r  a  m                         c
c=========================================================== c
```

```
*************************************************************
*                                                           *       *
*
*                  A three dimensional finite element model of a              *
*                  Generalized-Newtonian isothermal flow using                *
*                  the UVWP or the modified UVWP method.                      *
*                                                                             *
*                                                                             *
*                                                                             *
*************************************************************

              -------------------------------------------------------------------
               Sample output File.
              -------------------------------------------------------------------

              [[[ element description data..........
                 no.of nodes per element            =       8
                 no.of integration points           =       3

   *** coordinate system is cartesian (planar) ***

              [[[ mesh description data ..........
                 no.of nodal points              =     9062
                 no.of elements                  =    7560
                 no.of nodal constraints on boundary    =     8390
                 no.of different materials          =       1

              [[[ uniform body force vector ..........
                 grav1                        =       0.0000
                 grav2                        =       0.0000
                 grav3                        =       0.0000

 ******************** material properties***********************

  id.    eid.(from-to)   consistency co-efficient     power law index

  1          17560          80.000                        1.0000

 reference temperature   coefficient b   reference pressure  coefficient a

       293.000              0.0140      0.101E+06         0.200
 Dispersion Coefficient    Density        Shear rate

   0.200                0.1E+04         0.20000

 ******************* nodal coordinates *******************

      id.          x-coord           y-coord           z-coord

        1          0.000000          0.000000          0.000000
        2          0.000000          0.008333          0.000000
        3          0.000000          0.016667          0.000000
        4          0.000000          0.025000          0.000000
        5          0.000000          0.033333          0.000000
        6          0.000000          0.041667          0.000000
        7          0.000000          0.050000          0.000000
        .
        .
        .
     9056            1.000000          0.058333          0.093750
     9057            1.000000          0.100000          0.100000
     9058            1.000000          0.091667          0.100000
     9059            1.000000          0.083333          0.100000
     9060            1.000000          0.075000          0.100000
     9061            1.000000          0.066667          0.100000
     9062            1.000000          0.058333          0.100000

******************* element connectivity *******************

    id.      n o d a l - p o i n t  e n t r i e s

     1    79     2     1    78    86     9     8    85
     2    80     3     2    79    87    10     9    86
     3    81     4     3    80    88    11    10    87
     4    82     5     4    81    89    12    11    88
     5    83     6     5    82    90    13    12    89
     6    84     7     6    83    91    14    13    90
     7    86     9     8    85    93    16    15    92
     .
     .
     .
  7554  8802  8774  9026  9050  8809  8781  9032  9056
  7555  9052  9028  9027  9051  9058  9034  9033  9057
  7556  9053  9029  9028  9052  9059  9035  9034  9058
  7557  9054  9030  9029  9053  9060  9036  9035  9059
  7559   9056   9032   9031   9055   9062   9038   9037   9061
  7560   8809   8781   9032   9056   8816   8788   9038   9062



********************* nodal constraint ***********************


       id.      dof        value

        1        1        0.1000
        1        2        0.0000
        1        3        0.0000
        2        1        0.1000
        2        2        0.0000
        2        3        0.0000
        3        1        0.1000
        3        2        0.0000
        3        3        0.0000
        4        1        0.1000
        4        2        0.0000
        4        3        0.0000
        5        1        0.1000
        5        2        0.0000
        5        3        0.0000
        6        1        0.1000
        6        2        0.0000
        6        3        0.0000
        7        1        0.1000
        7        2        0.0000
        7        3        0.0000
        8        1        0.0000
        8        2        0.0000
        8        3        0.0000
        9        1        0.0000
        9        2        0.0000
        9        3        0.0000
        .
        .
        .
     4639        4        0.0000
     4645        4        0.0000
     4651        4        0.0000
```

```
Total number of time steps  =    5


 Deltat                   =        0.0010



    iteration no.    5



 nodal velocities and  pressures

  id.   u            v            z                press

    1  0.1000E+00   0.0000E+00   0.0000E+00        0.37831296E+02
    2  0.1000E+00   0.0000E+00   0.0000E+00        0.37732291E+02
    3  0.1000E+00   0.0000E+00   0.0000E+00        0.37624267E+02
    4  0.1000E+00   0.0000E+00   0.0000E+00        0.37558306E+02
    5  0.1000E+00   0.0000E+00   0.0000E+00        0.37518140E+02
    6  0.1000E+00   0.0000E+00   0.0000E+00        0.37495612E+02
    7  0.1000E+00   0.0000E+00   0.0000E+00        0.37487137E+02
    .
    .
    .
 9056   0.0000E+00   0.0000E+00   0.0000E+00        0.48201533E+00
 9057   0.0000E+00   0.0000E+00   0.0000E+00        0.00000000E+00
 9058   0.0000E+00   0.0000E+00   0.0000E+00        0.99745670E-01
 9059   0.0000E+00   0.0000E+00   0.0000E+00        0.20045870E+00
 9060   0.0000E+00   0.0000E+00   0.0000E+00        0.29998994E+00
 9061   0.0000E+00   0.0000E+00   0.0000E+00        0.39553431E+00
 9062   0.0000E+00   0.0000E+00   0.0000E+00        0.48493800E+00

node no.         max ux    node no.         min ux
  231        0.23911295E+00 9003       -0.41400282E-02

node no.         max uy    node no.         min uy
 7849        0.17519545E+00 2349       -0.22368228E-01

node no.         max uz    node no.         min uz
   77        0.13617839E+00 1020       -0.16184282E-01
```

## APPENDIX 4
## PROGRAM MANUAL

The Fortran software is incorporated in the Visual Studio 2005, it can be found by clicking on Start\Programs\Microsoft Visual 2005\ Microsoft Visual 2005. Once the software opened, click on Open Project followed by double clicking on the desired project name in order to open it.

If the program is to be run for the first time, then it is a good practise to rebuild it by clicking on Build\Rebuild Flowsolution09. Bear in mind that Flowsolution09 is used in this manual simply because it was a name given to the program. Once the Building process successfully done, then the program can be run by following the steps described below

- ► Step 1: Click on Debug
- ► Step 2: Select Start Without Debugging
- ► Step 3: When prompted to enter the GFORM file name then type in the Name of the gfm file created using GeoStart followed by the **.**gfm extension, and hit the Enter key from the keyboard.
- ► Step 4: Enter the desired scale factor (1 is usually preferred) press the Enter key again.
- ► Step 5: The steps described above will create the data file and display the basic discretizations variables (nodes, elements, and boundary conditions) on the screen.
- ► Step 6: At this stage, the user will be prompted to enter the date file name created in step 5. Type in the file name with the **.**dat extension and press the Enter key. In this program, the data file name is set to input.dat.
- ► Step 7: Once step 6 completed, the user will be prompted to enter the Number of time steps desired. Press the Enter Key after entering the value.

► Step 8: Enter the value of delta t when prompted, then hit the Enter key.

► Step 9: Enter the value of alpha and press the Enter key once again.

► Step 10: The last step of the process consists of choosing the desired scheme, and this can be achieved by typing 1 for the UVWP scheme or 2 for the modified UVWP scheme. Then followed by a last hit on the Enter key.

The following tutorial is used to illustrate the 10 steps described above. In this tutorial, a gfm file will be created after the geometry definition, meshing, and the specification of the boundary conditions. The gfm file will be given the name model.gfm, which will be used at later stage to create the data file. The data file will be given the name input.dat.
The user must consul the "Help" section of GeoStar where he/she could get further information about the software. The user is strongly advised to try the GeoStar tutorials available from the "Help" section.

The domain in this tutorial consists of a simple rectangular box of 1m length, 0.1m width and 0.1m high and there is no obstruction to the flow as shown in figure A4.1. The computational domain is discretized using 8-noded hexahedral isoparametric elements into a mesh of 3751 nodes, and 3000 elements and the prescribed boundary conditions are as follow; the fluid enters the domain with a velocity of 0.1m/s perpendicular to the inlet; the other components of the velocity (v, and w) are zero (see figure A4.2 through A4.3). The only prescribed boundary condition at the outlet is a zero datum pressure, and the no-slip conditions are applied to the remaining sides of the rectangular box.

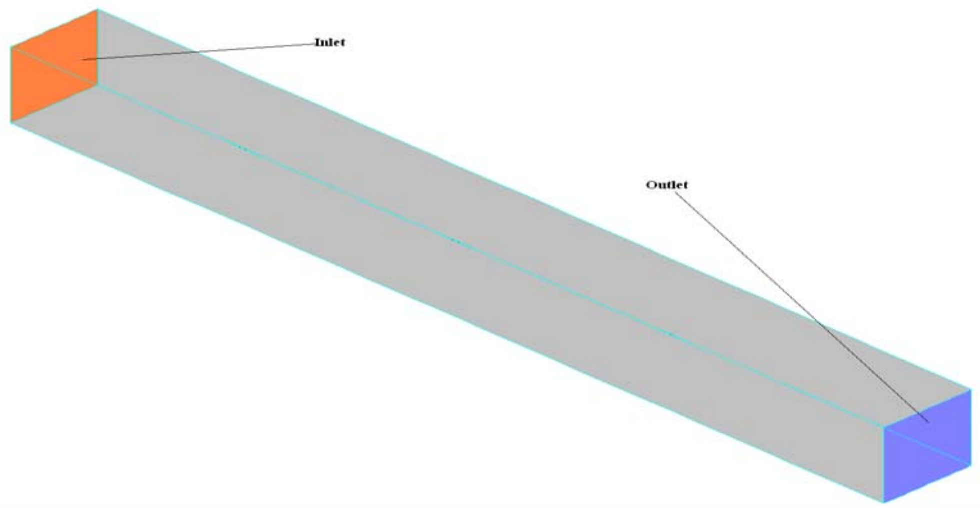


**Figure A4.1:** Geometry of example 1.

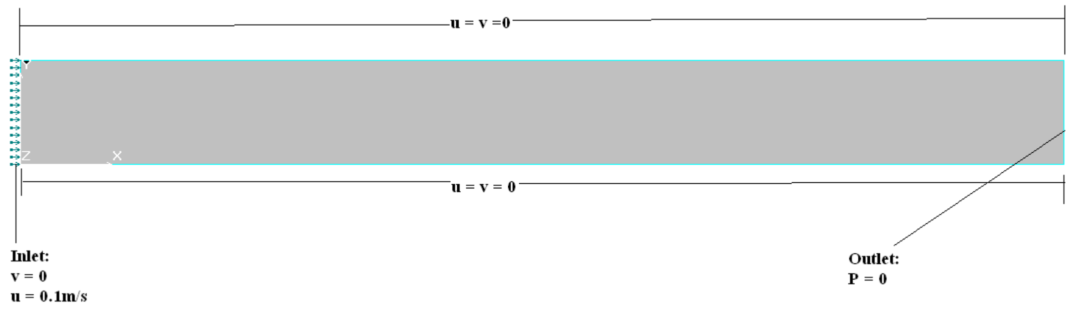


**Figure A4.2:** 2-D schematic representation of the boundary condition in the xy plane.

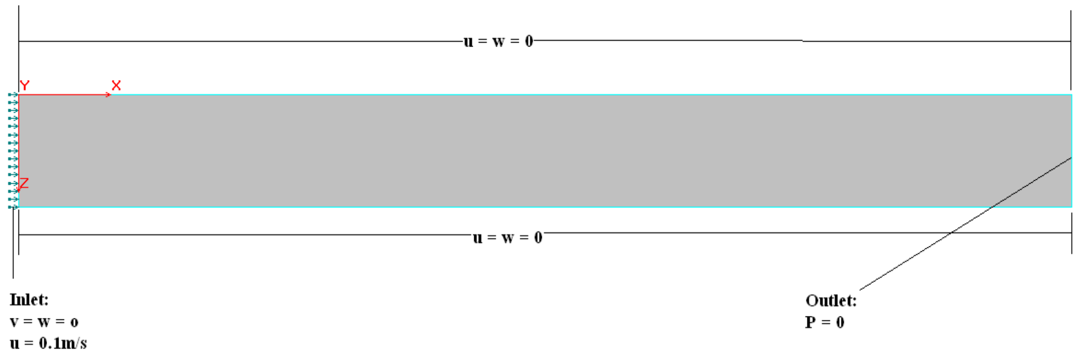


**Figure A4.3:** 2-D schematic representation of the boundary condition in the xz plane.

**Pre-processing steps**

► Begin the step by clicking on Start\Programs\Cosmos Applications\GeoStart 256.

► Select My Documents, and create a new folder (TutoExple in this case).

► Double click this newly created folder (TutoExple) and type in model in the file name dialog box then click on the "Open" button.

► Select "Yes" from the "Open Problem Files".

► Click on Status 1 from the Geo Panel (left hand side of the screen) and check the PT, CR, SF and VL under the "Labl" tag. Once this is done, click on the "Save" button.

► The next step consists on creating 8 points with the following coordinates:
Point 1 (0,0,0), point 2 (0,0.1,0), point 3 (1, 0.1, 0), point 4 (1, 0,0).
point 5 (0,0,0.1), point 6 (0,0.1,0.1), point 7 (1,0.1,0.1),and point 8 (1, 0, 0.1).
This can be achieved by clicking on Geometry\Points\Define from the Geo Panel.

► Click on "Auto" under the "Scale" tag from the Geo Panel in order to get a better view of the points created. The user is expected to get an image similar to one given by figure A4.4.

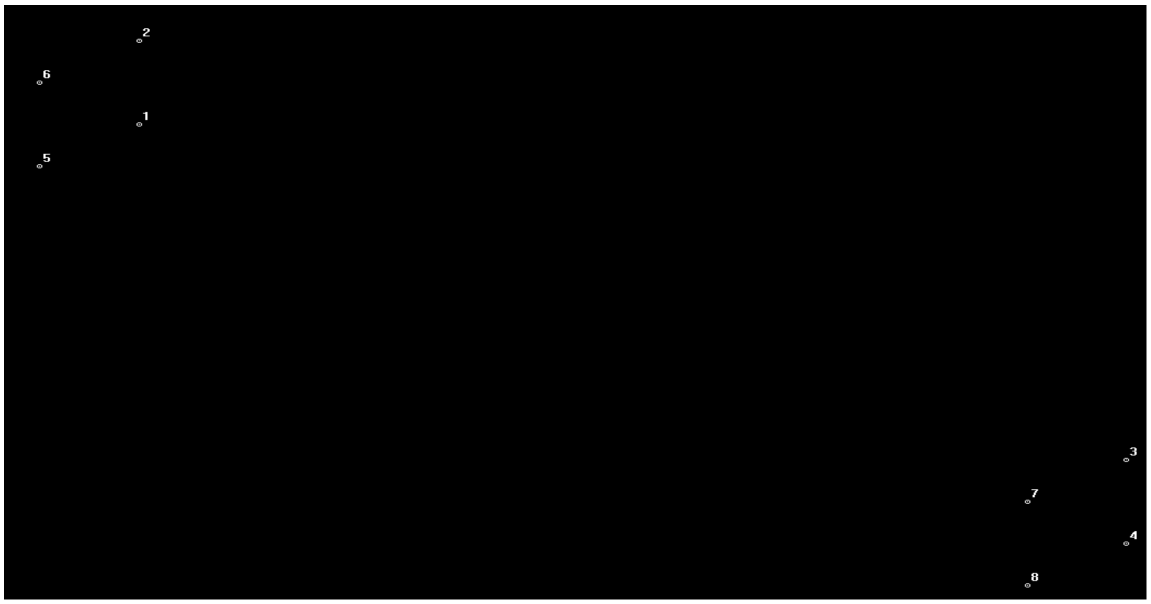


**Figure A4.4:** 8 Points representation.

► Since the domain in this example consists of a simple rectangular box, one can create the volume directly by linking the 8 points as follow:
Click on Geometry\Volumes\8 points from the Geo Panel, and then enter numbers 1 to 8 into the Vertex Keypoint dialog boxes. Click the "OK" button once this is done.
The user is expected to have an image similar to the one given by figure A4.5

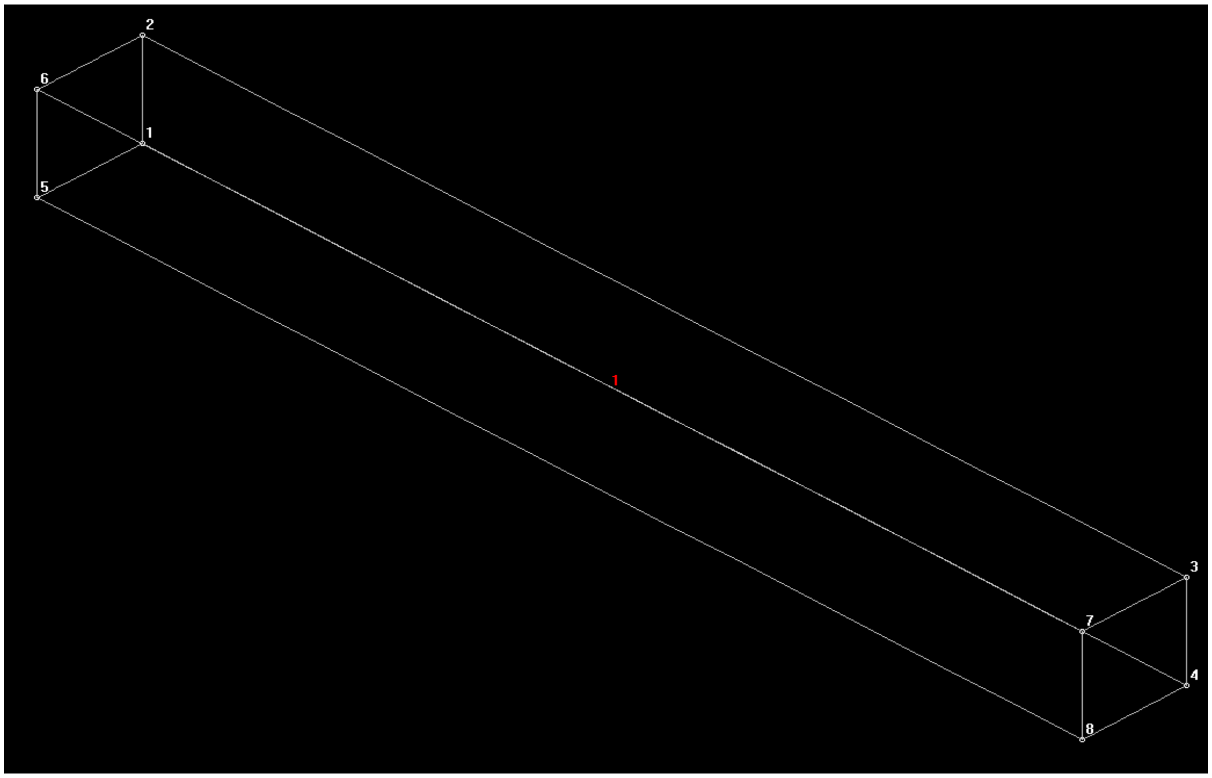


**Figure A4.5:** Volume representation.

► The next step is to proceed with the finite element discretization of the domain into elements and the associated nodes. This can be achieved by selecting Meshing\Parametric_Mesh\Volumes from the Geo Panel. Keep unchanged the default value of 8 (representing 3D brick element) from the number of nodes per element, but enter 10 into the number of elements on the first curve, 30 into the number of elements on the second curve, and 10 into the number of elements on the third curve dialog boxes. Leave the remaining values from the dialog boxes unchanged and click the "OK" button.

► The previous action discretized the domain into a finite element mesh of 3000 elements and 3751 nodes (see figure A4.6). This can be verified by clicking on the Status 1 button from the Geo  Panel.

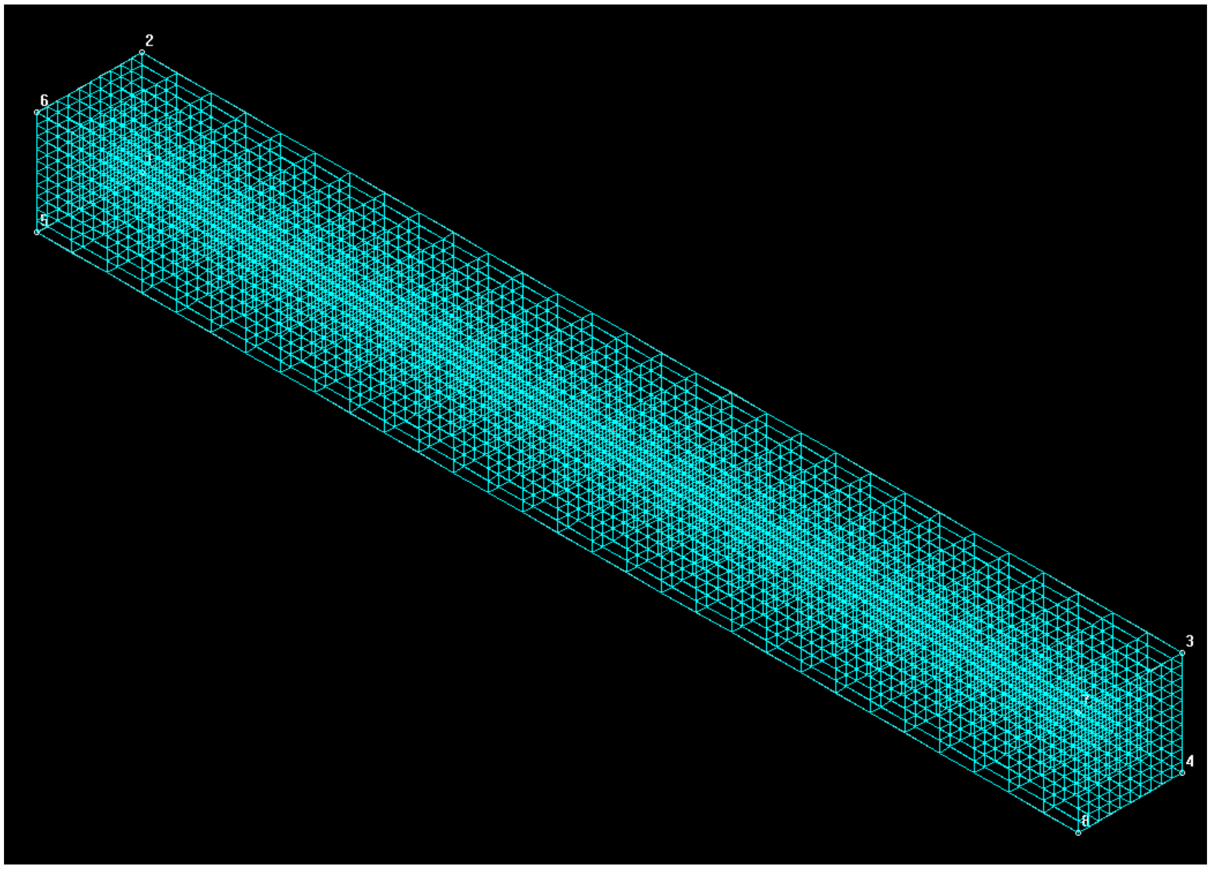


**Figure A4.6:** Finite element mesh

**Note:** For complex geometries, it is important to merge the nodes and elements after the discretization of the domain. This can be done as follow:

For nodes; click on Meshing\Nodes\Merge and when the NMERGE window appear keep all the values to their default values, then press the "OK" button. Once this is done, click on Meshing\Nodes\Compress from the Geo Panel. This will lead the user to the NCompress window where one needs

to click on the "OK" button to complete the task.
Similar actions must be performed for elements by clicking Meshing\Elements\Merge Elements and the "OK" button followed by Meshing\Elements\Compress.

► The next step is to assign boundary conditions to the discretized domain. To this end, click on Clear Screen (CLS) button (bottom left side) from the Geo Panel, then plot the Domain by clicking on Edit\Plot\Surfaces from the Geo Panel, this will give an image shown by figure A4.7.

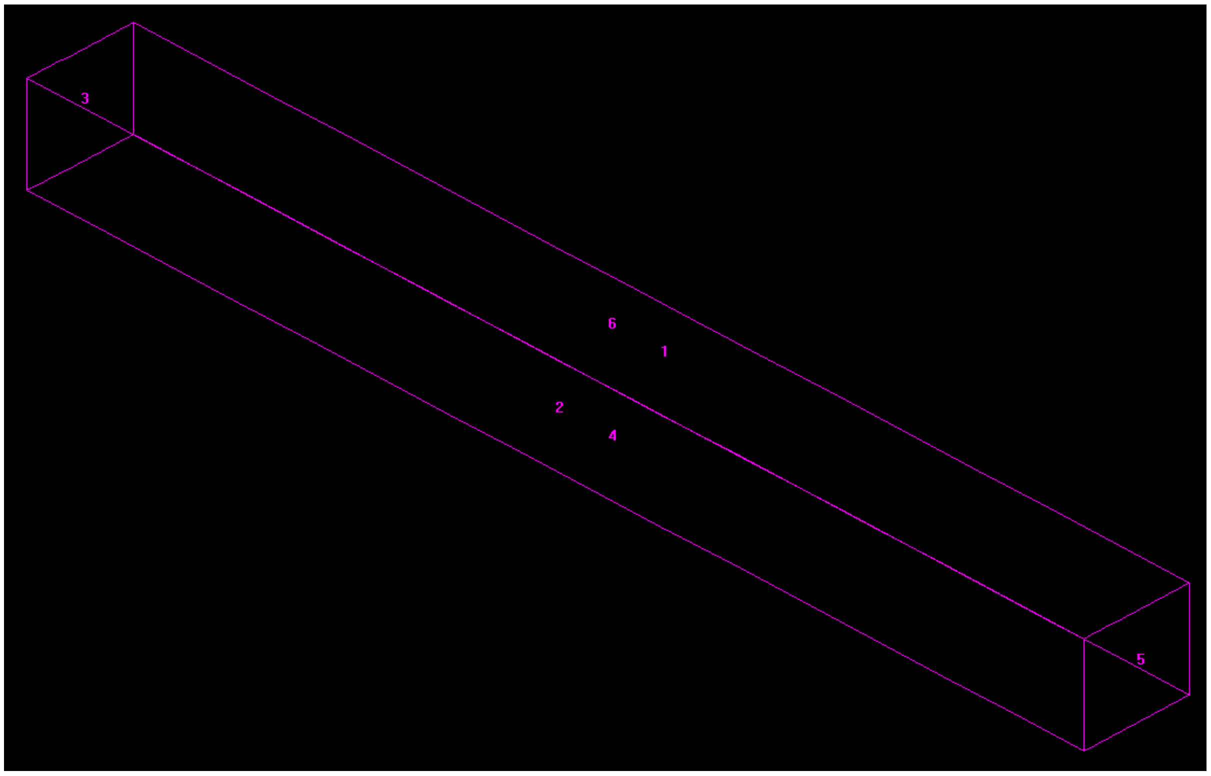


**Figure A4.7:** Surface plot

The inlet in figure A4.7 is represented by surface 3, the outlet by surface 4, while the solid walls are represented by surfaces 1, 2, 4, and 6.

► Click on LoadsBC\Fluid_Flow\Velocity\Define by Surfaces from the Geo Panel, and enter the following data from the "VSF" dialog box.

| | |
|---|---|
| Beginning Surface: | 3 |
| Velocity label: | VX; Velocity in X |
| Value: | 0.1 |
| Increment: | 1 |

Then click on "OK".
Repeat the same operation to specify the boundary conditions in the VY, and VZ direction with a value of 0 each.

At the end of these operations, the user is expected to have a figure similar to figu re A4.8.

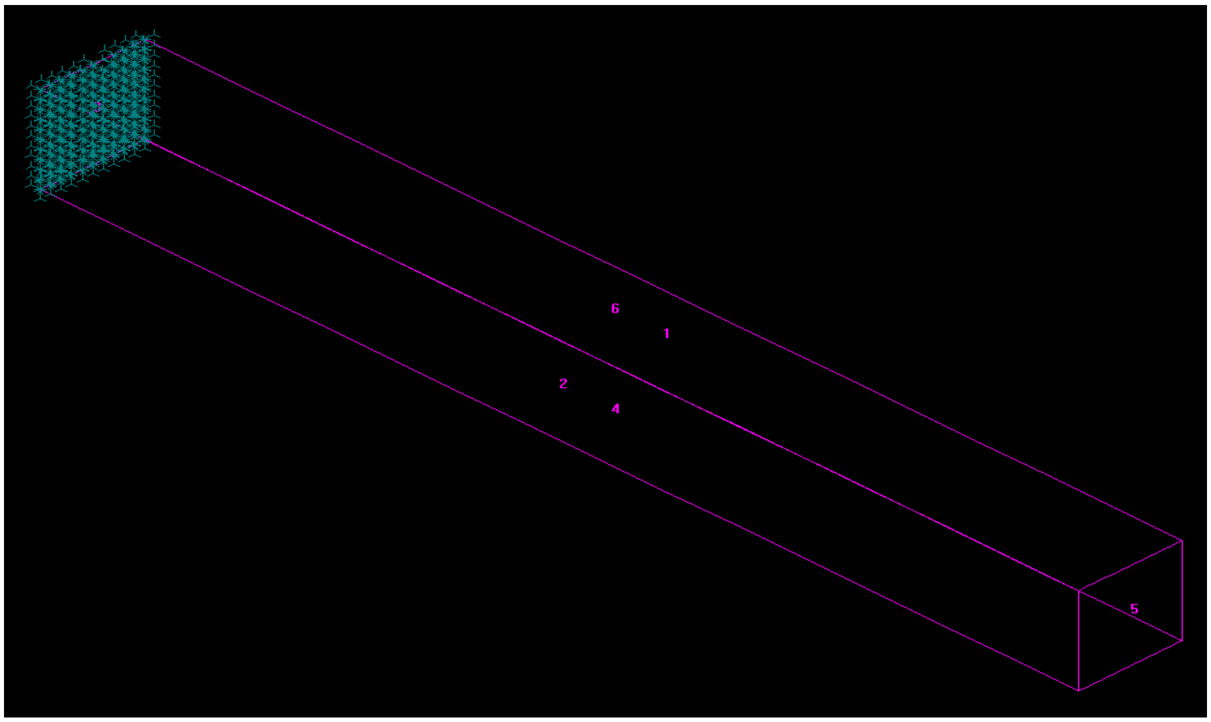


**Figure A4.8:** Inlet boundary conditions

The boundary conditions on the solid walls (surfaces 1, 2, 4, and 6) can be specified as for the inlet case but with the following differences:

Velocity label: Al; Velocity in all

Value: 0.

The boundary conditions at the exit (surface 5) can be specified as follow

► Click on LoadBC\Fluid_Flow\Pressure (Nodal)\Define by Surfaces, then from the "NPRSF" dialog box enter

Beginning surface: 5

Value: 0

Ending surface: 5

Increment: 1

And click "OK".

At the end of these operations, the user is expected to have a figure similar to figure A4.9

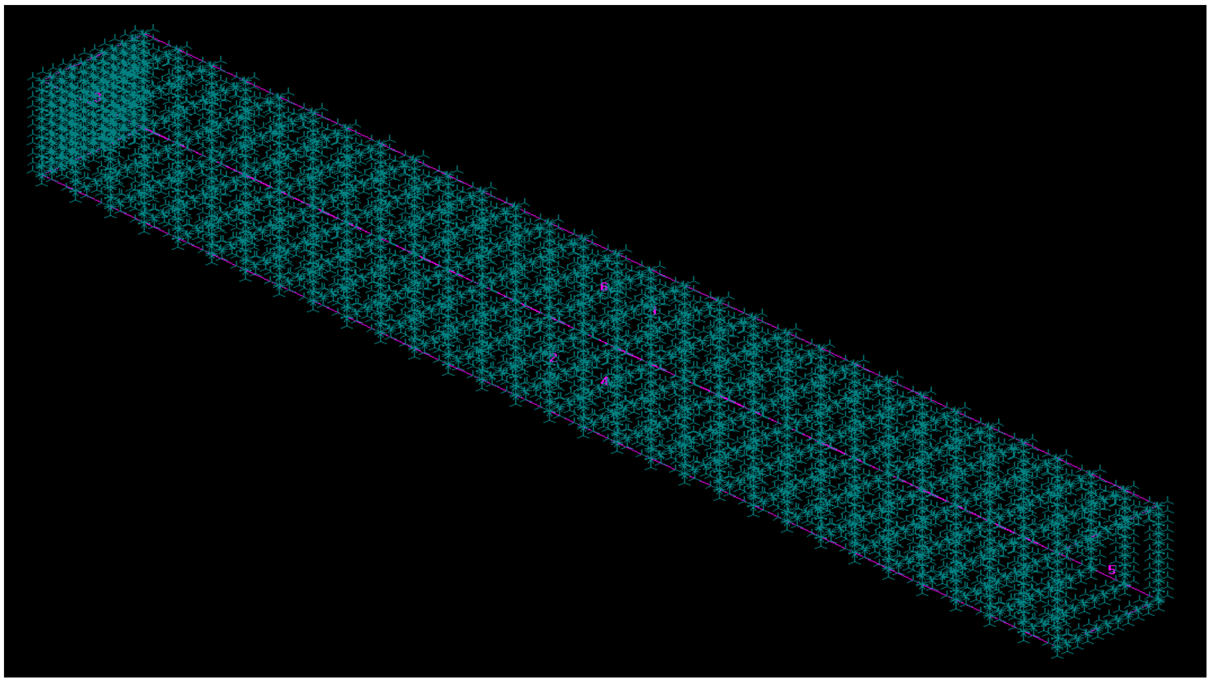


**Figure A4.9:** Domain with boundary conditions.

**Note:** To avoid viewing the mesh attached to the 3D contour plot during the post-processing, it is important at this stage to do the following operation:

From the Geo Panel, click on Display\Display_Option\Eval Element bound, then select "Yes" as values of the boundary face evaluation and boundary edge evaluation boxes and click "OK".

The last step of the pre-processing stage is the creation of the **.**gfm file. This can be achieved by the following operation.

► From the Geo Panel, click Control\Utility\Create GFM file, then click "Continue" following by a second click on the "OK" button.

This mark the end of the pre-processing steps.

**Note:** GeoStar will automatically give the geometry name plus the **.**gfm extension (model.gfm for this tutorial) to the GFM file created.

**Solver steps**

Before proceeding to the solution of the problem, the user must make sure that a copy of the GFM file (model.gfm for this tutorial) created is copied and pasted in the folder containing the FORTRAN program (My Documents\Visual Studio 2005\Projects\3FlowSoluFinal09C\FlowSolution09; for this tutorial). Once this done, Then the program can be run by performing the operations as explained in steps 1 through 10 which can be summarise here as:

From the FORTRAN software (Microsoft Visual Studio 2005)

► Click Debug\Start without Debugging, then enter the name of the GFM file (model.gfmfor this tutorial) when prompted, and press the Enter key.

►Type in the desired scale factor (1 is usually the best value) and hit the Enter key again.

These last two actions will create a data file with the name input.dat then display information about the finite element mesh (number of nodes, elements, boundary conditions…) on the screen.

► When prompted type in the data file name (input.dat) and press the Enter key.

► Enter the desired time steps and press the Enter key.

► Type in the value of delta t and hit the Enter key.

► Enter the value of alpha and press the Enter key.

► Select the desired numerical scheme (1 or 2) then press the Enter key to start the solution process.

Once the solution process terminated, the program will create two output files (CosmGraph and tecpGraph.dat) that the user can use to proceed with the post-processing.

CosmGraph can be used for post-processing analysis via Cosmos GeoStar software, while tecpGraph.dat can be used with Tecplot software.

**Note:** If Cosmos GeoStar is chosen as post-processing software, then the user must copy the CosmGraph file and paste it to the desktop location before visualizing the variables.

**Post-processing via Cosmos GeoStar software**

Once a copy of the CosmGraph file is saved on the desktop, the user can go the Geo Panel menu then click on Results\Plot\User Results in order to get the plot window (ACTUSRPLOT). Click on find, then select the CosmGraph file which is saved on the desktop and click open. Once this done, the user can now plot the variables (pressure and velocity) in term of contour and section plots.
The numbers in “Component number” dialog box represent the different components of the velocity (1 for U, 2 for V, 3 for W, and 4 for their magnitude) and pressure (5).

For instance the pressure contour plot can be obtained by selecting 5 as value of Component number and press the “Contour plot” button from the “ACTUSRPLOT” window. This will give a plot given by figure A4.10

The velocity magnitude contour plot can be obtained similarly but with the difference that 4 must be entered as the value of Component number, and the plot is as given by figure A4.11

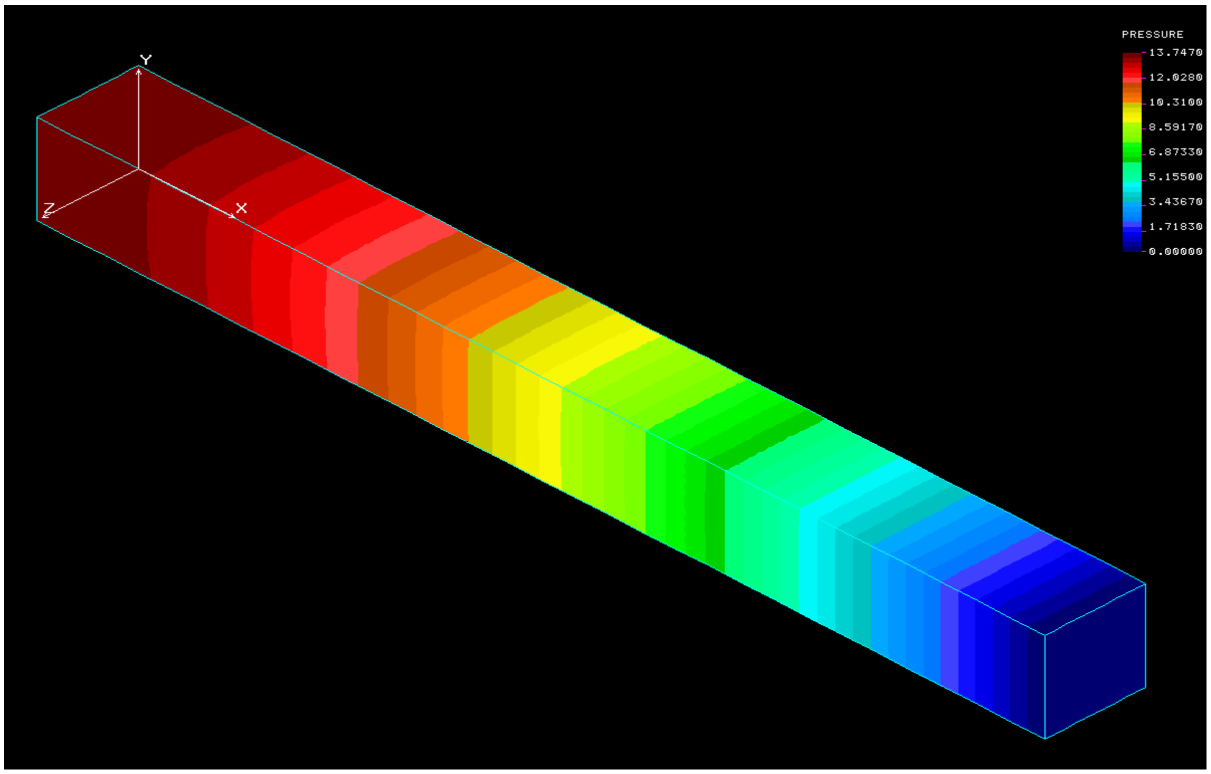


**Figure A4.10:** Pressure contour plot.

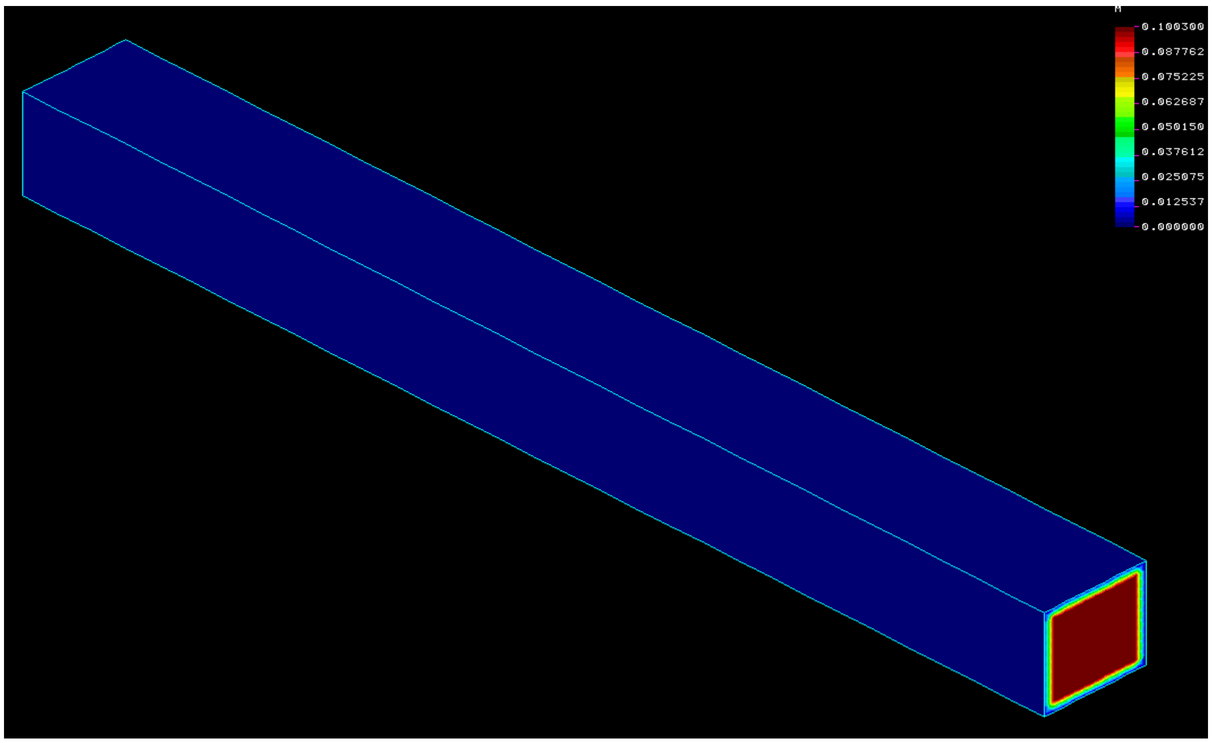


**Figure A4.11:** Velocity magnitude contour plot.

Section plots can be obtained similarly, but by pressing Section plot button instead of the Contour plot button. Once the Section plot button pressed, the user will be prompted to choose a desired plan in the "Orientation of section planes" dialog box from the "SECPLOT" window. Click the continue button after choosing the desired plan, then type in the number of plan needed and select 1: Yes (if the section plot (s) is or are to plotted at specific location(s)) otherwise leave it to its default value of 0: No. Once this done, click the continue button again followed by a last click on the OK button.

Some sample of section plots are given by figures A4.12 through figure A4.13

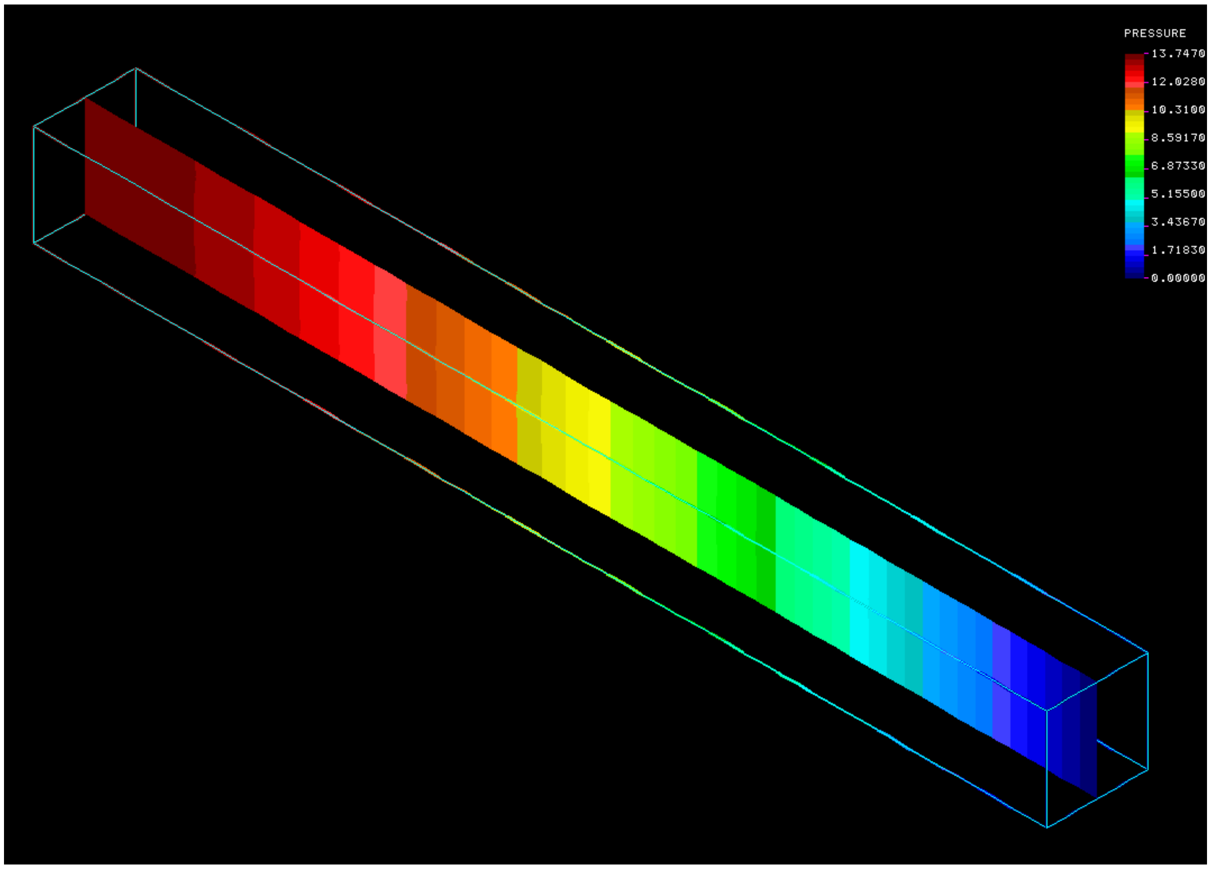


**Figure A4.12:** Pressure section plot in the Z plan.

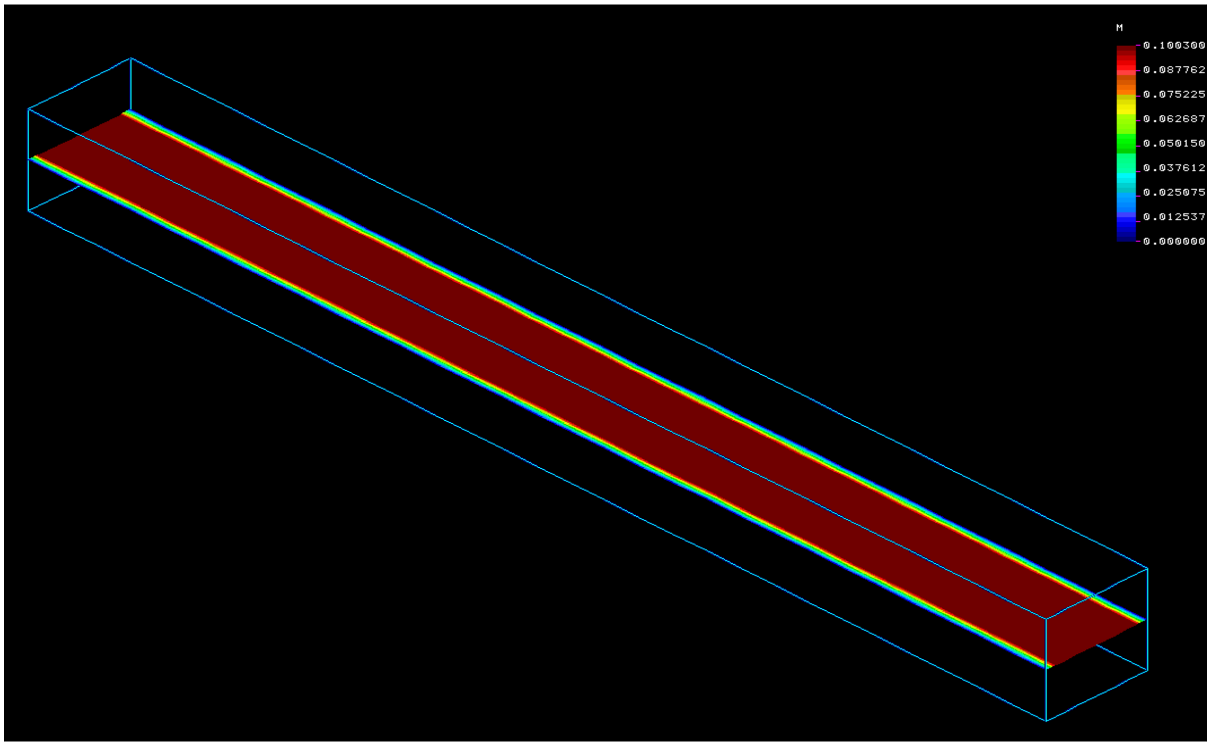


**Figure A4.12:** Velocity magnitude section plot in the Y plan.

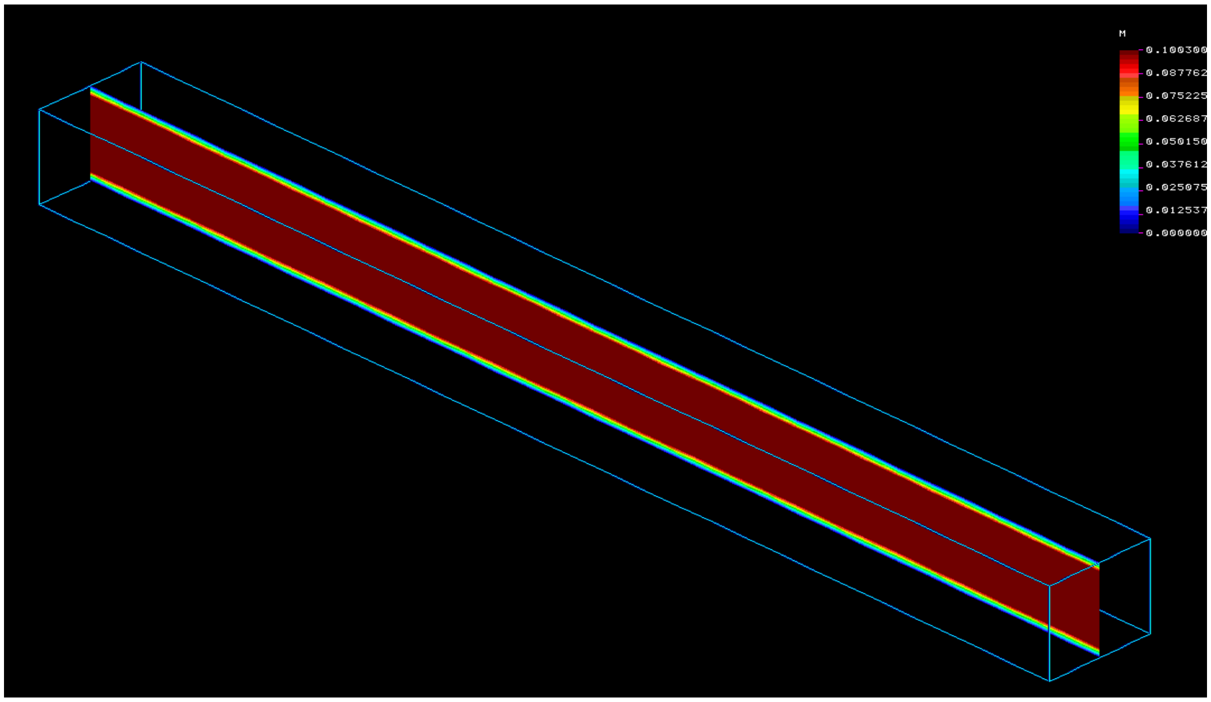


**Figure A4.12:** Velocity magnitude section plot in the Z plan.

## Post-processing via Tecplot software

Once the Tecplot software is turn on, then do the following

► Select File\Load Data File(s), then select Tecplot Data Loader from the "Select Import Format" window.

► Select the Tecplot output file (tecpGraph.dat) from its location and click OK from the "Select Initial Plot" window.

► If the orientation of the geometry is not as expected, then this can be corrected by clicking on X, Y, or Z button under "Option and Tools", then move the mouse over the geometry to change the orientation.

► Click on View\Fit to Full Size to have a good view of the object.

► Uncheck the Mesh dialog box from the "Zone Surface".

► Check the Contour dialog box to make contour plots. Click the "…" button opposi te to Contour and select the Legend tag to add a legend to the plots.
The "Contour & Multi-Colouring Details" window contains information about the variables. Pressure is represented by P, the components of the velocity vector by U, V, W, and M for their magnitude.

► Pressure contour can be plotted by selecting P from the "Contour & Multi-Colouring Details". This will give a plot as given by figure A4.13.

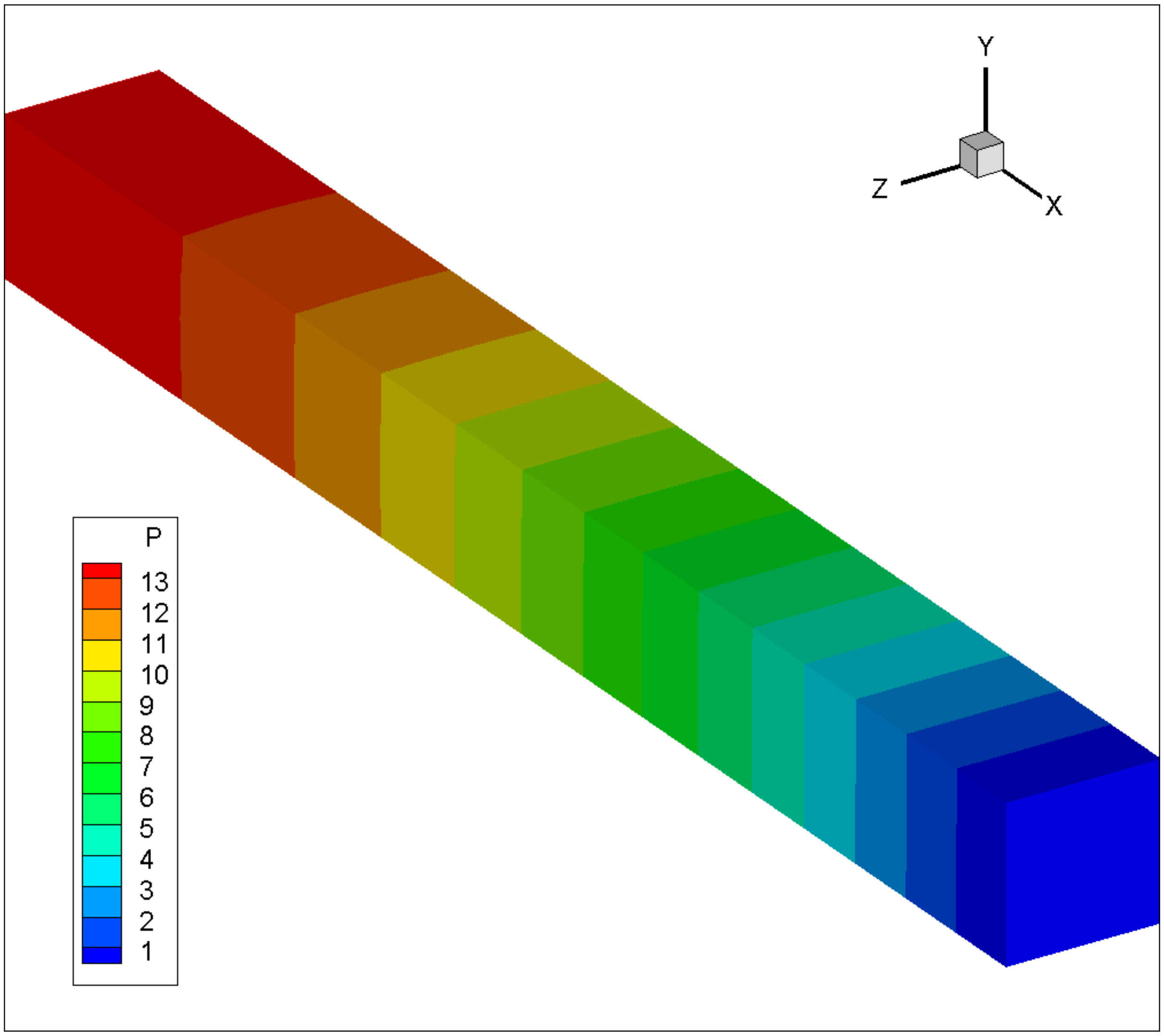


**Figure A4.13:** Pressure contour.

► Selecting M will plot the velocity magnitude shown by figure A4.14

► To plot section of plots, the user must uncheck the Contour from the "Zone Surfaces", then check the "Shade" and "Translucency" boxes, and checked the "Slice" box under "Derived Objects". Click the "**…**" button opposite to "Slice" to get the "Slice Details" window from which different plot planes can be obtained. Vectors plot can be obtained by clicking the "Vector" tag from the “Slice Details” window then checking the “Show Vectors” dialog box and pressing the OK button from the “Select Variable” window. The length of the vectors can be modified by clicking of “Plot” from the main menu (top screen) then select “Vector\Length\, and choose “Uniform” or one of the Relative options.

Some samples of the velocity magnitude and section plots are given by figure A4.14 through figure A4.19.

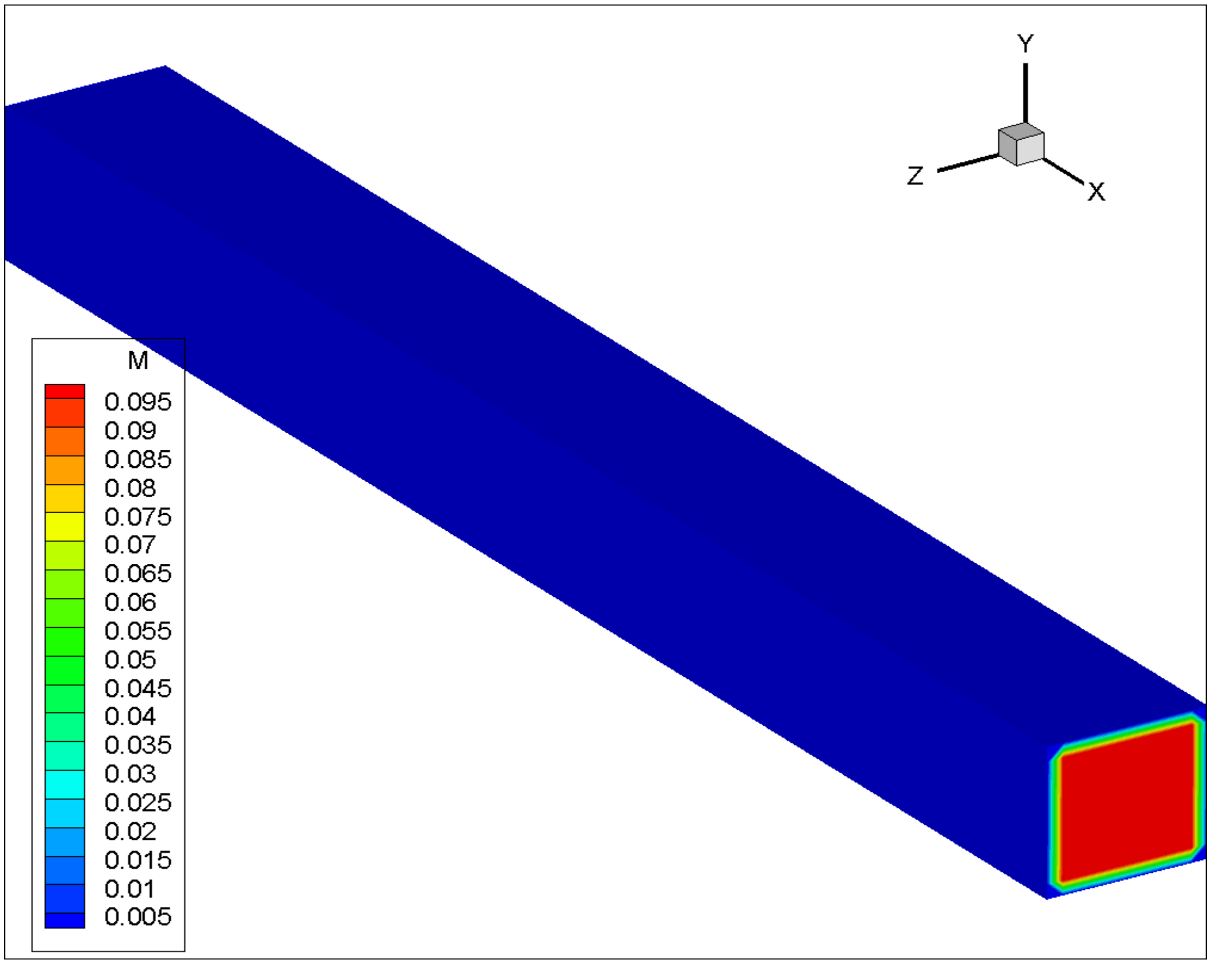


**Figure A4.14:** Velocity magnitude contour.

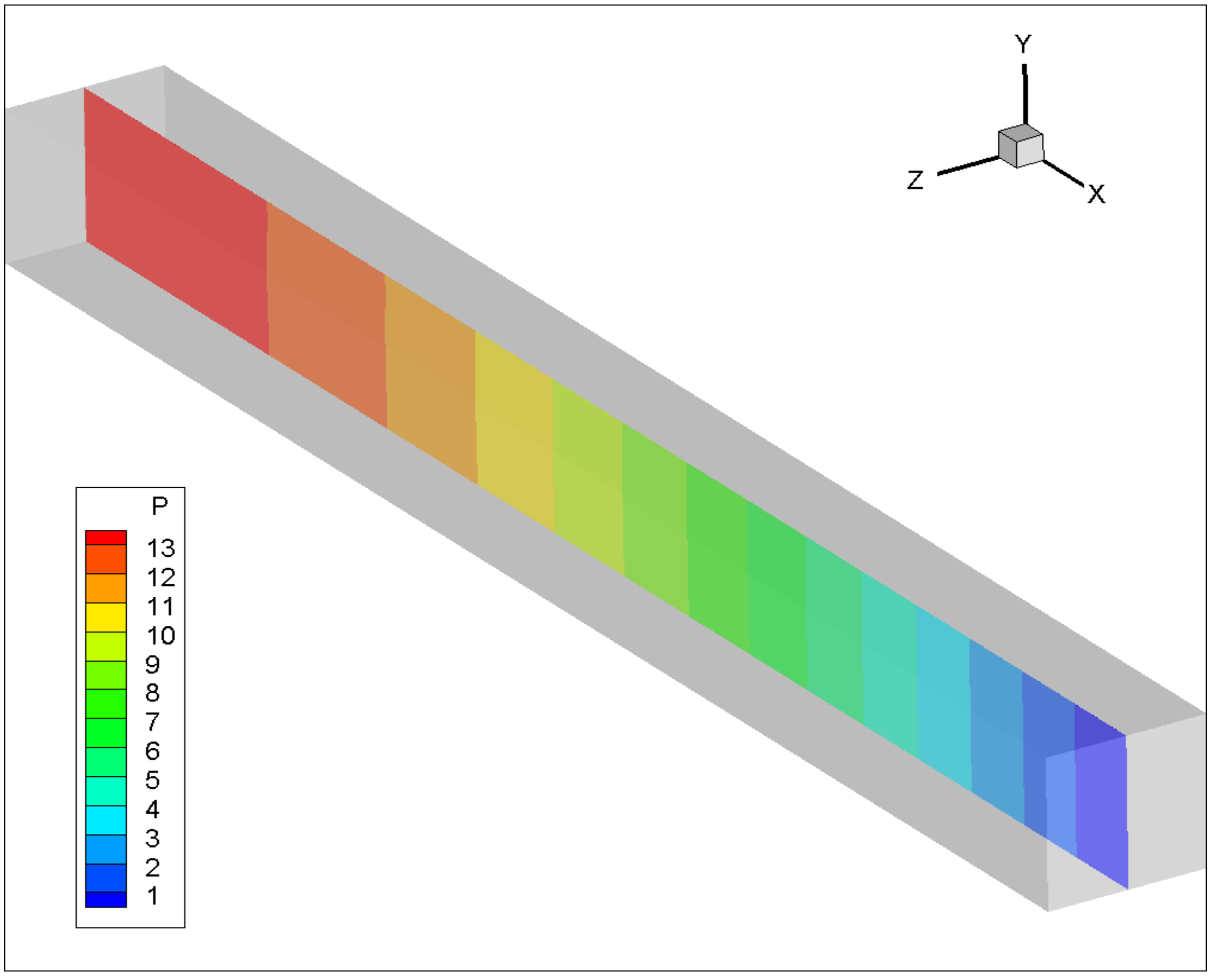


**Figure A4.15:** Pressure section plot in the Z plan.

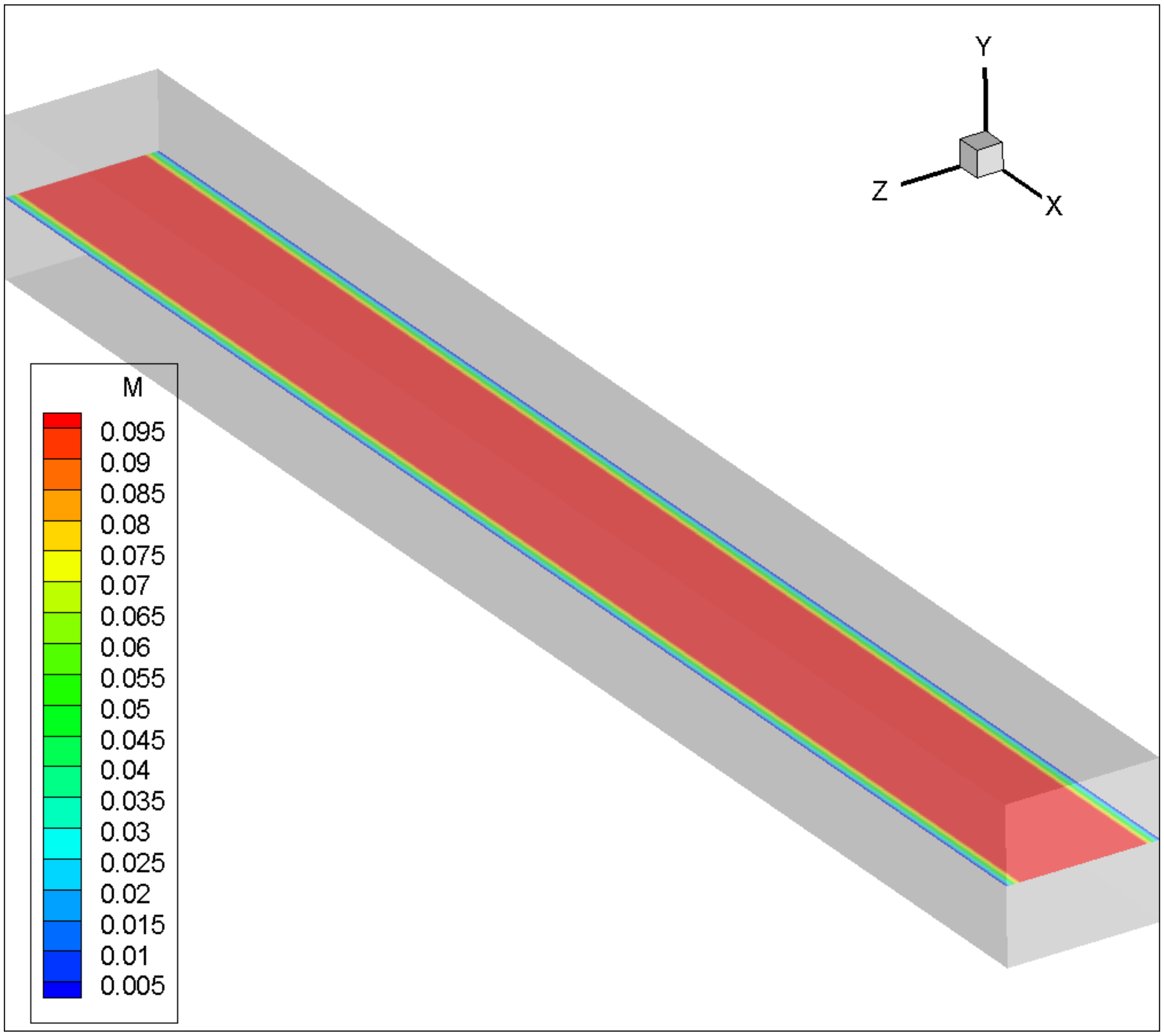


**Figure A4.16:** Velocity magnitude section plot in the Y plan.

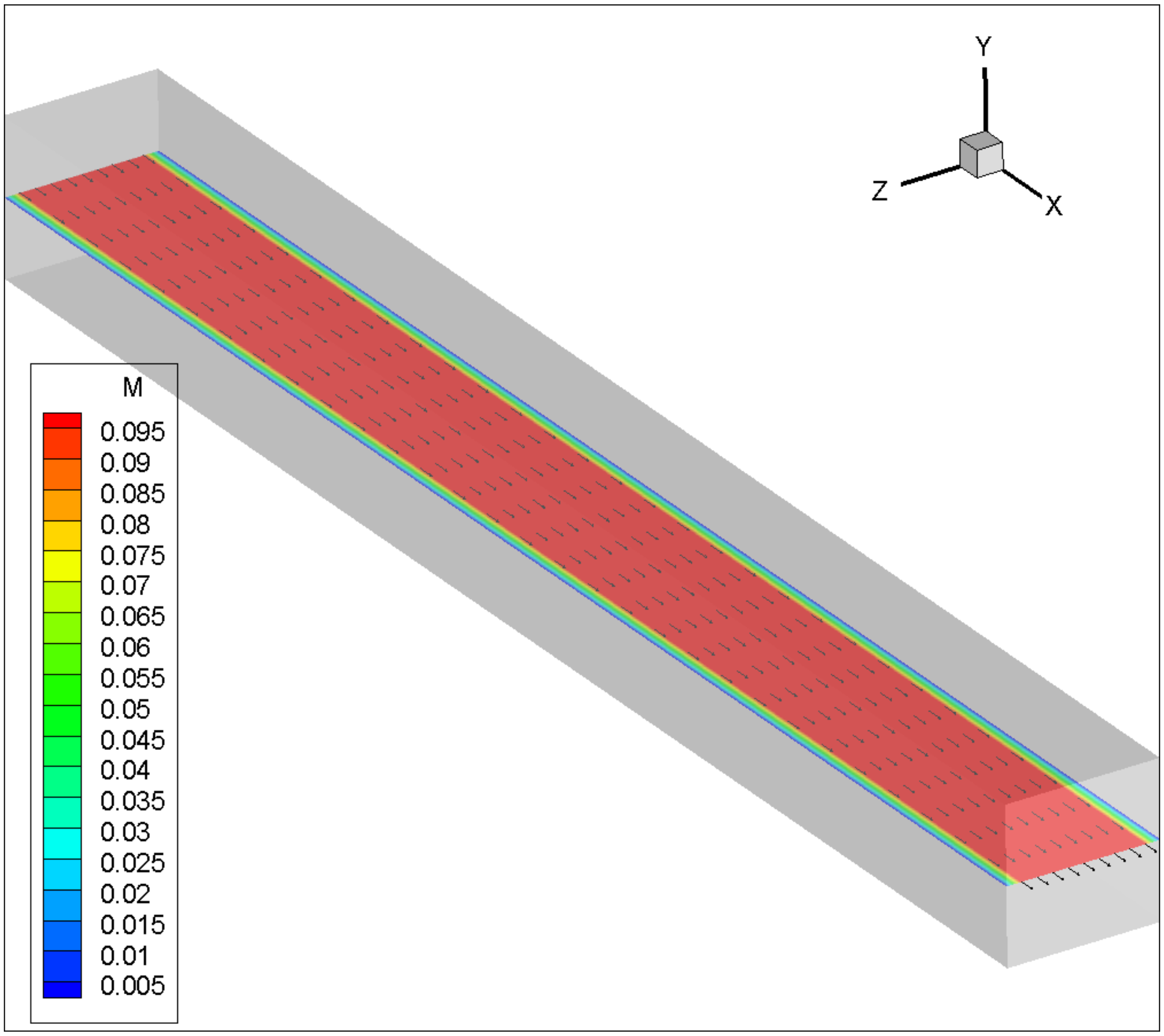


**Figure A4.17:** Vector section plot of the velocity magnitude in the Y plan.

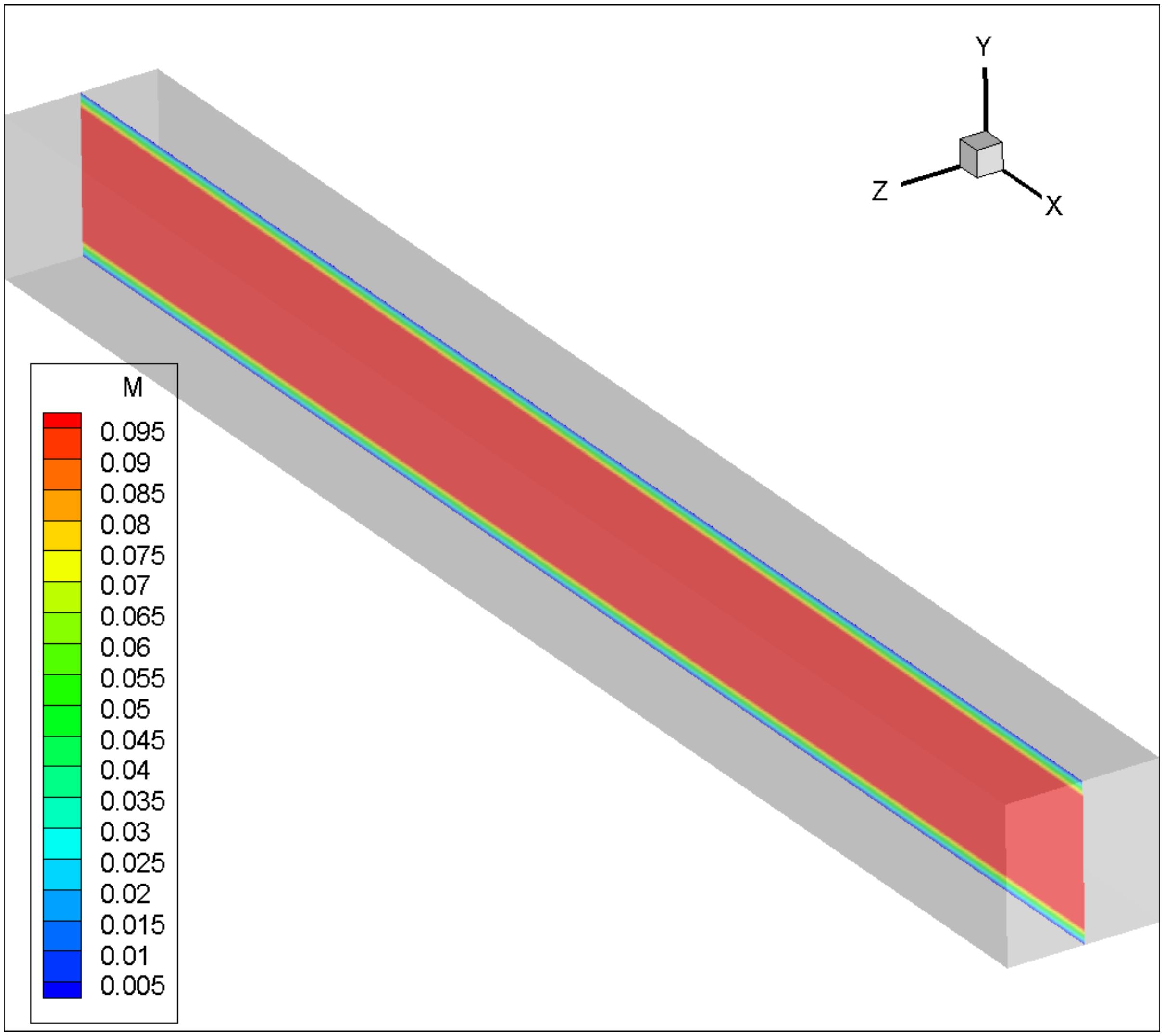


**Figure A4.18:** Velocity magnitude section plot in the Z plan.

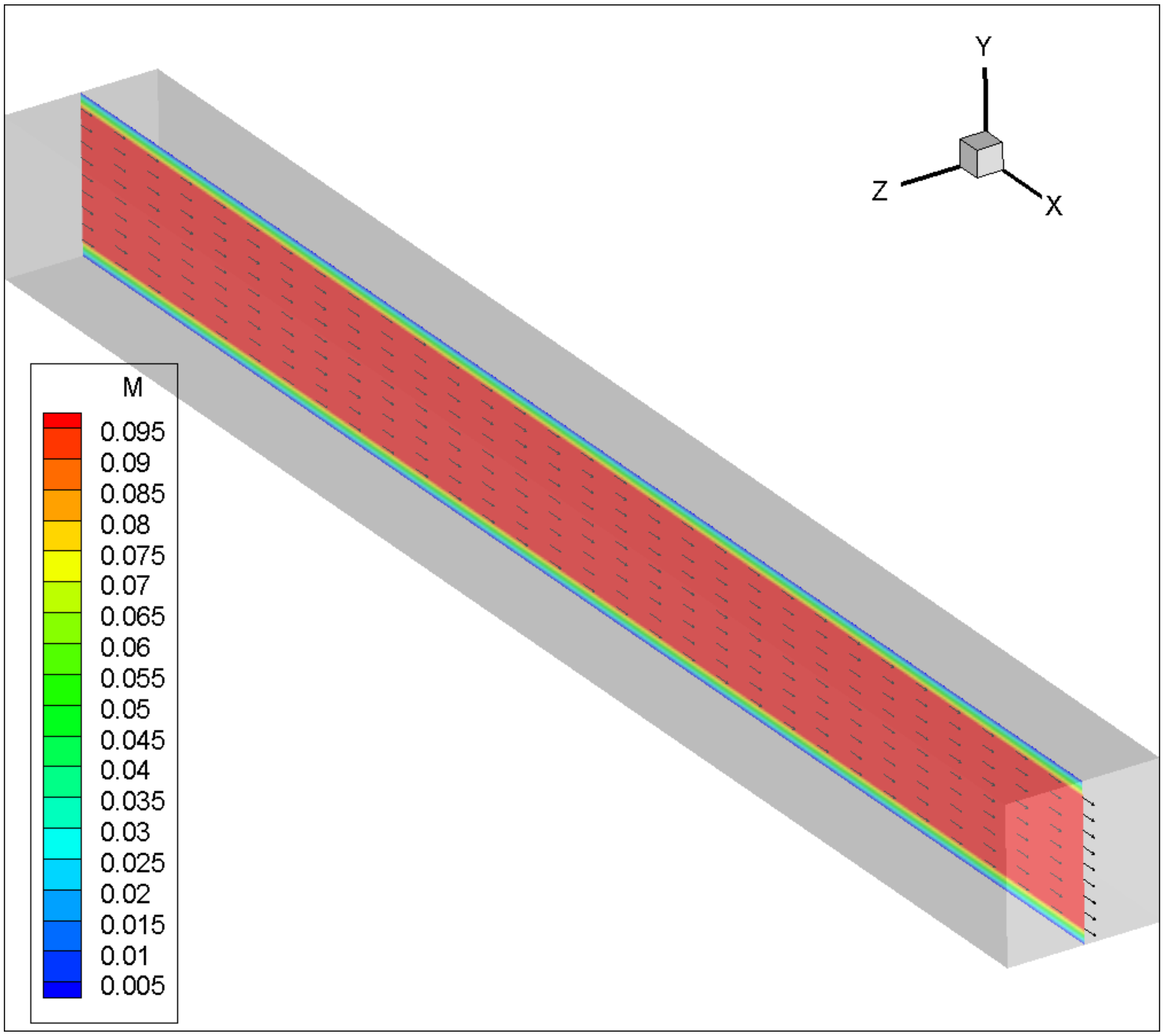


**Figure A4.19:** Vector section plot of the velocity magnitude in the Z plan.

## Closure

The aim of this tutorial is to initialise the user to the Cosmos GeoStar (pre-processing and post-processing parts), FORTRAN (solver), and Tecplot (post -processing) environments. The user is strongly advised to try the tutorials available from the help sections of both Cosmos GeoStar and Tecplot software for further information.