Implementation of Segregated Solvers for Incompressible and Compressible Flow in Collocated Grids

Francisco de Lemos Cabral Granadeiro Martins

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. José Carlos Fernandes Pereira
Dr. Duarte Manuel Salvador Freire Silva de Albuquerque

Examination Committee

Chairperson: Prof. Carlos Frederico Neves Bettencourt da Silva
Supervisor: Dr. Duarte Manuel Salvador Freire Silva de Albuquerque
Member of the Committee: Prof. João Carlos de Campos Henriques

June 2018

Acknowledgments

Em primeiro lugar gostaria de agradecer à minha família, em especial aos meus pais e à minha irmã, por tudo o que fizeram por mim, tudo o que me ensinaram e me transmitiram até hoje, que, de certa forma, culmina neste trabalho.

Um grande obrigado ao Dr. Duarte Albuquerque, orientador desta tese, pela inigualável disponibilidade, apoio e conselhos dados desde o primeiro ao último momento este trabalho.

O Manuel Costa e o Rui Parente também merecem os meus agradecimentos, por umas vezes me ouvirem falar de coisas que não compreendiam, e mesmo assim tentarem ajudar, e, outras vezes saberem do que falava e ainda darem mais ideias. Para além disso, também gostaria de lhes agradecer o companheirismo, pois as horas de almoço são sempre muito mais animadas com eles por perto.

O Pedro Oliveira, o Rodrigo Trindade e o Gonçalo Henriques. Muito obrigado por tornarem as viagens de comboio e os dias no técnico infinitamente melhores.

Todos os que fizeram parte do projecto PSEM também merecem um grande agradecimento, pelo seu companheirismo, amizade e por tudo o que aprendi com eles.

Por último, também gostaria de agradecer ao André Veiga, à Beatriz Viúla e todos os outros com quem mantenho amizades duradouras.

Resumo

O algoritmo SIMPLE foi extendido, por forma a permitir a simulação de escoamentos compressíveis. A implementação deste algoritmo é explicada passo a passo, e as suas diferenças em relação ao algoritmo SIMPLE incompressível são estudadas. Foi também dada ênfase à discretização dos termos viscosos das equações de energia e de momento, em casos de escoamento compressível, com viscosidade, quer constante, quer variável.

Nesta adaptação do algoritmo SIMPLE foi proposto um método de estabilização equação de correcção de pressão, que é obtida através do cálculo de um factor de relaxação necessário para garantir que a diagonal principal da matriz seja dominante.

Para testar o algoritmo, foram realizadas simulações de vários casos clássicos, presentes na bibliografia, e que incluem exemplos de escoamentos para vários números de Mach.

A implementação foi feita em Matlab, onde foram desenvolvidos dois programas: um para a simulação de uma tubeira convergente divergente, e um para a simulação de vários casos 2D, tanto invíscidos, como viscosos. Neste último programa, foi estudado o impacto dos esquemas convectivos upwind, linear e NVD Gamma, na solução.

Foi também incluida no algoritmo a capacidade de simular escoamentos não estacionários, recorrendo ao esquema de Euler implícito, tendo sido testado no caso do escoamento invíscido sobre um degrau a $Mach = 3$, com e sem correcção dos erros numéricos gerados junto do canto do degrau que é proposta na literatura para este caso.

Palavras-chave: Método de Volume Finito, Escoamento Incompressível, Escoamento Compressível, Algoritmos Acoplamento Pressão-Velocidade, Mecânica de Fluidos Computacional

Abstract

An extension to the SIMPLE algorithm was developed, with the objective of solving compressible flow problems. The implementation of the algorithm was explained in detail, and its main differences to the original SIMPLE for incompressible flows are emphasized. Special attention was given to the discretization, in cartesian and uniform grids, of the viscous terms in the energy and momentum equations, in compressible flow, with constant and variable viscosity.

To test the algorithm, several benchmark cases available in the literature were solved, in compressible flows over a wide range of Mach numbers.

A pressure correction stabilization procedure is included in the algorithm, where the main diagonal dominance of the matrix is guaranteed by an under relaxation factor calculated automatically.

Two programs featuring the algorithm were developed in Matlab: one to solve the quasi 1-D convergent-divergent nozzle problem, and another one to solve 2D test cases for both viscous and inviscid fluids. In the latter one, the upwind, linear and Gamma NVD schemes were implemented, and their performance compared.

The ability to solve unsteady flows was also included in the algorithm, wherein the implicit Euler scheme was used, and was tested on the Mach 3 inviscid flow over a forward facing step. The impact of the numerical errors generated on the step corner and the corrections suggested by other studies were also analysed.

Keywords: Finite Volume Method, Incompressible Flow, Compressible Flow, Pressure-Velocity Coupling Algorithms, Computational Fluid Dynamics

Contents

List of Tables

List of Figures

Nomenclature

It is noted that some variables may have a different meaning along the text.

Acronyms

CDS	Central Differencing Scheme
CV	Control Volume
FV	Finite Volume
HOS	Higher Order Scheme
UDS	Upwind Differencing Scheme
NVA	Normalize Variable Approach
NVD	Normalize Variable Diagram
$SIMPLE$	Semi-Implicit Method for Pressure Linked Equations
SOR	Successive Over Relaxation
TVD	Total Variation Diminishing

Roman Symbols

c_p	Heat capacity at constant pressure
f	Face
$\mathbf{f}_b$	Force exerted on a body vector
$\mathbf{g}$	Gravity acceleration vector
h_t	Total enthalpy
k	Thermal conductivity
$\mathbf{n}_f$	Normal vector of a face
p	Pressure
p'	Pressure correction
P	Cell or volume control (mesh)
q_ϕ	General source term
$\mathbf{u}$	Velocity vector
$u,\ v$	Velocity components - Cartesian coordinates
Ra	Rayleigh number
Re	Reynolds number
T	Temperature

Greek Symbols

α	Thermal diffusivity

Γ	General diffusive term
ϕ	General variable
Φ	Viscous stress component of the energy equation
$\tilde{\phi}$	General transported variable
μ	Dynamic Viscosity
ρ	Density
ν	Kinematic Viscosity
Ψ	Viscous stress component of the energy equation
$\boldsymbol{\tau}$	Viscous Stress Tensor

Operators

D	Material derivative
∂	Partial derivative
$\mathbf{I}$	identity matrix
$\nabla\phi$	gradient
$\nabla \cdot \mathbf{u}$	divergence
$\nabla^2\phi$	Laplacian
$\|\mathbf{u}\|$	Euclidean norm
$\sum$	Sum

Chapter 1

Introduction

1.1 Motivation

In the latest decades, computational fluid dynamics has become an essential tool in engineering. It has been increasingly used in many industries such as the automotive, aeronautic and naval ones as a way to understand how fluids behave in a wide range of circumstances and flow regimes. This knowledge can be used to quantify important physical quantities, like the drag force that a car is subjected to when racing, or the lift generated by the wings of an aeroplane, thus allowing the analysis of different engineering solutions in terms of their upsides and downsides.

It is important to remark that fluid flow can also be studied experimentally. When an innovative engineering solution is being designed, the ideal way to do it is to compare the numerical results to the experimental ones, to try to emulate the real situation as well as possible. Yet, sometimes, the problem isn't reproducible on a smaller scale, or in a way that allows for an experimental study, furthering the importance of CFD, since the designers must rely only on the numerical analysis of the problem.

To simulate a flow numerically, a solver must be used to determine the solution of the equations that govern fluid motion in that particular problem, therefore, solvers are the very core of CFD software, be it commercial, home made, or, in this case, research-oriented. Many solvers are specifically designed to solve flows in a specific regime, e.g. incompressible or compressible flow (i.e. a flow where density variations can't be neglected), low-Mach flow or isothermal flow, any of which already includes copious amounts of applications. The issue arises when a particular problem features fluid flow in several regimes, either seriously hindering the usage of these more specific solvers, or thwarting it altogether, and justifying the development of all-Mach solvers.

This work is the first step before implementing an all-Mach solver in the SOL software. This software cannot solve flows where density varies, since only incompressible flow solvers are implemented. Hence, the sheer amount of additional cases which could be studied, would, in itself, justify the implementation of one of these solvers in SOL.

From a more personal point of view, implementing an all-Mach solver grants the coder with a detailed perception on how CFD software is developed and how the different parts of that software are linked, something that is invaluable for anyone interested in this area (even when using commercial software). Another motivation, with a more personal nature, is the desire to learn more about compressible flows, since they are important in many real-life applications.

1.2 Literature Review

In order to design a solver to simulate fluid flow problems, there is an abundance of strategies that can be followed, but two often stand out for the case of finite volume or difference methods: density based solvers and pressure based solvers.

1.2.1 Density Based Solvers

Historically, density based solvers were developed to simulate high-speed compressible flows [5] . These solvers use density as the main variable, a strategy that works well in compressible regime. Albeit, density variations become negligible in the incompressible flow, depleting its coupling with pressure and the efficiency of these solvers. Despite this drawback, there have been successful attempts to adapt these methods to incompressible flow (like adding artificial compressibility [1]), wherefore, with these modifications, they can be classified as all-Mach solvers.

1.2.2 Pressure Based Solvers-SIMPLE

Pressure based solvers were originally designed to solve incompressible flows, by using pressure as the variable which ensures that continuity is respected. One of the most famous, and perhaps oldest, pressure based methods is the SIMPLE algorithm [6]. It is a guesser-corrector algorithm for pressure, that was proposed to be used in staggered grids (preventing the pressure checker board problem) and introduced the concept of pressure correction to guarantee continuity

Since then, many different adaptations of this method have been proposed: SIMPLER [7], the revised form of the original algorithm, which includes both an equation for pressure and one for presure correction. The pressure values are updated by solving the pressure equation, and the pressure correction equation only is used to correct the velocity values, thus ensuring continuity. SIMPLEC [8], which features a different pressure correction equation than SIMPLE, due to the different assumptions made when deriving it. To overcome the pressure checker board issue in colocated grids, [11] proposed what is known as the Rhie and Chow interpolation, in which the face velocities are calculated by taking into account the pressure gradients not only in the face, but also in the control volumes that share it. In 1985 another variant of the SIMPLE-like method was proposed by [9], where two consecutive corrector steps are used, i.e. the corrected pressure field and face fluxes resultant from the first corrector step are used in the second one to correct the cell centered ones. It would become known as the PISO algorithm. These algorithms were compared by [10] for steady flows, who concluded that there is no overall best algorithm, for their performance depends on the type of flow to be simulated.

1.2.3 Adaptation of the SIMPLE Algorithm to All-Mach Flows

One of the first pressure based solvers that was able to handle both supersonic and subsonic flows was proposed by [12] to predict the supersonic viscous flow near walls, both in laminar and turbulent regimes. Here, the density field was recommended to be recalculated at the end of each time step via the ideal gas equation, after the energy equation was solved. The pressure correction yielded the velocity correction values to guarantee the continuity constraint with the density value from the last iteration.

An all-Mach solver was proposed by [13]. In this article, an extension to the incompressible SIMPLE-like algorithms in staggered grids was introduced. This breakthrough was accomplished by writing the compressible form of the continuity equation and transforming it into a pressure correction one, in which both density and velocity are corrected as function of that pressure correction. Most pressure based all-Mach solvers since then follow this formulation. The article also compared the performance of several SIMPLE-typed methods when adapted to be all-Mach methods.

In 1988, [14] presented an all-Mach version of the SIMPLER algorithm in arbitrary configurations using staggered grids. The pressure and density fields are recommended to be corrected by solving the pressure equation and the velocity and mass flow rates by solving the pressure correction equation. In the same year [15] devised an all-Mach pressure based inviscid flow solver with adaptive grids in order to enhance the accuracy near shocks.

The SIMPLES: ”SIMPLE-Supersonic” was introduced by [16]. An all-Mach solver for collocated non-orthogonal meshes, which included the Rhie and Chow interpolation. The results for several schemes, including TVD and second order upwind differencing were compared in this article, all of which were implemented using the deferred correction approach to overcome the convergence difficulties faced by all-Mach solvers when higher-order schemes are employed. A multigrid technique for staggered grids was proposed in the same year by [17] to accelerate the simulations, something that had also been proposed earlier by [18] for the compressible version of the PISO algorithm.

Another boundary fitted method was presented in [19], using the Rhie and Chow interpolation and the upwind scheme. Here, the results were compared with the ones obtained when using a staggered grid.

The algorithm devised by [20] had an ameliorated accuracy by using a deferred correction blending between the upwind and central differencing schemes. This work also provides additional details on the compressible flow boundary condition implementation, and on the stability issues of the pressure correction equation in all-Mach solvers. These issues will be further addressed in section 3.

An all-Mach pressure based solver was used to simulate turbulent flow by [21]. The concept of density retardation was introduced to improve stability by securing the positiveness of the density and pressure values. This concept was also used by [22] who used LU decomposition to solve the equations of transport and the $k-\epsilon$ turbulence model to simulate the flow in turbomachinery applications.

A high-resolution scheme was implemented by [23], by taking advantage of the deferred correction approach. The discretized governing equations were also written in an alternative notation.

Another extension to the algorithm, allowing it to solve turbulent cavitating flows, was proposed in [24]. A mass transfer method is used to simulate the phase change of the fluid and the $k-\epsilon$ is used to model turbulence.

1.2.4 Coupled All-Mach Solvers

More recently, the attention has been shifted towards coupled solvers, both due to the computational time required when employing segregated algorithms and because of the large increase in available computational power and memory in the latest decade.

An incompressible pressure based solver on structured grids was developed by [25], who later developed two versions of all-Mach solvers: [26] for inviscid flows in unstructured grids, and [27] used to solve turbulent flows in OPENFOAM employing a rotating frame of reference. Both of these solvers couple a pressure equation with the momentum equations, which are solved simultaneously, but leave the energy equation to be computed afterwards, in a segregated way.

An algorithm for unstructured grids where the momentum, pressure and energy equations are all coupled was proposed by [28], where the only equation left to be solved afterwards is the ideal gas equation. The results presented include the solution of a 3D flow over a backwards facing step in laminar flow with constant viscosity and an omnipresent comparison between the computational effort (CPU time) with a classic segregated solver.

An alternative approach (also for unstructured grids) was proposed in [29]: solving the coupled momentum and pressure equations in a constant temperature cycle until the desired tolerance is attained, followed by the computation of the energy equation. The density is updated via the ideal gas equation in every inner iteration of the cycle. This algorithm has been heavily tested, with results for many different flows presented.

1.3 Objectives

The objectives that were determined at the start of this work and laid down its main path were:

- Alter a previously written code in Matlab to allow a better grasp on how the SIMPLE algorithm is implemented for incompressible flow;
- Implement the Boussinesq approximation in an incompressible flow solver to solve convection driven flows;
- Develop and delineate clearly the key steps of an all-Mach solver;
- Test and develop in a quasi one dimensional code in Matlab with variable cross-section;
- Implement the previously developed all-Mach solver in Matlab to solve 2D inviscid flows with cartesian and uniform grids;
- Further test the solver by adding the ability to simulate viscous flow;
- Implement the all-Mach solver in SOL by expanding it to unstructured meshes;

1.4 Present Contributions

The objectives for this thesis stated above are a compilation of the tasks and general pointers that were initially planned. This section serves the purpose of clarifying what was achieved during this work.

Two previously existing incompressible flows codes were adapted. This was a good way to earn some experience in the implementation of the SIMPLE algorithm. The first of the two codes simulated the viscous flow in a duct and was modified to allow the simulation of the flow over a baffle (whose position and existence can be defined by the user) inside the duct.

The second code that was altered simulates the lid-driven cavity. It was modified to allow the simulation of a convection-driven cavity via the Boussinesq approximation. The main changes to the code were the addition of the energy equation as well as some minor changes to the momentum equations.

A quasi-1D flow in a convergent-divergent nozzle simulator, already featuring the all-Mach solver, was programmed in Matlab. This permitted a good feel of how the 2D version was had be implemented, mainly on how to avoid the stability related issues in the pressure correction equation, and in the mathematical conditions that should be used in which situations and how to implement them. The results of this test are recorded in section 4.1.

A 2D version of the algorithm was implemented in Matlab for and underwent several tests, whose results are detailed in section 4.2,and 4.3. The several types of flows with which this code was tested include: viscous incompressible flows, isothermal and non isothermal, with constant and variable viscosity (a feature that can be seen as a primordial step towards the accommodation of turbulence models), supersonic inviscid flows and supersonic viscous flows (in which the energy dissipation through viscosity is taken into account), all of which can be simulated in transient or steady state. Several discretization schemes have been implemented (Upwind, Linear Interpolation and the Gamma NVD[4]), as well as the implicit Euler scheme for unsteady problems.

The program allows the user to easily choose from any of the previously installed test cases, or select the geometry, boundary conditions and mesh to try out custom tests cases, like the one if figure 1.1.

Two variants of the energy equation have also been included in this program, one that is solved in function of temperature and another of the enthalpy. The reason for this choice to be included is explained in section 2.1, but is related to the discretization of the viscous stress tensor in the energy equation (an important characteristic of viscous supersonic flows).

Both programs in which the all-Mach solver is implemented include numerous additional options to aid the debug and simulation, as: plots of that show the simulation's progress, a fail-safe option to guarantee diagonal dominance in the algebraic matrices and the ability to resume previously saved simulations, as well as other options.

1.5 Thesis Outline

In this section, the content of each chapter will be outlined:

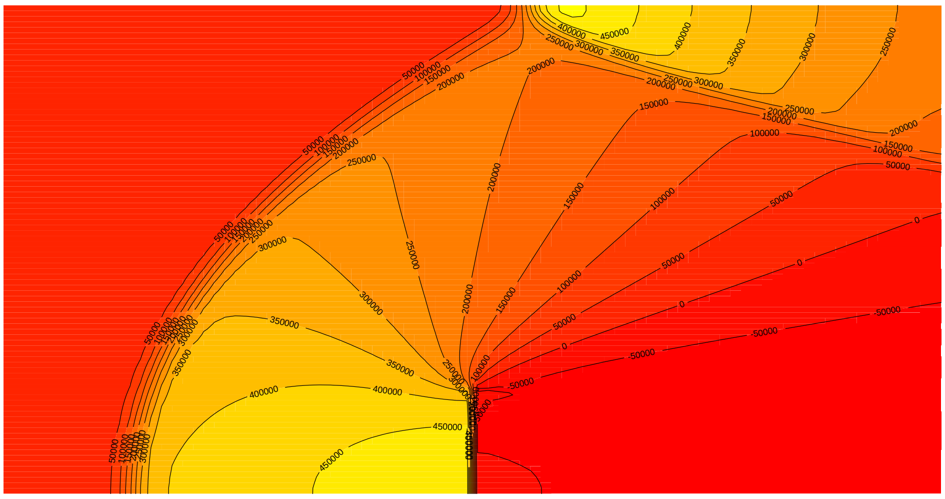


Figure 1.1: Pressure contours of a custom test case: Mach 2 inviscid flow over a baffle.

Beginning with chapter 2, a detailed explanation on the governing flow equations' terms for the incompressible, and compressible flow regimes ,and how they can be simplified in in each of them. Afterwards, the general transport equation for a scalar is presented, and discretized to exemplify the proceedings of the finite volume method. This section includes the integration, spatial and temporal discretizations, their respective schemes, boundary conditions implementation, and, in the end, a section dedicated to the discretization and implementation of each of the governing equations.

In chapter 3 the structure of the SIMPLE algorithm, and its implementation are detailed. In the same chapter, the all-Mach algorithm that was developed during this work is described, with special emphases on its stability, boundary condition implementation and on how it differs from the original SIMPLE algorithm.

The results obtained are shown in chapter 4 . The test cases simulated in the quasi 1D convergent-divergent nozzle and the 2D codes are included in this chapter. Several notes on difficulties found when solving the problems, and information on relevant parameters, e.g. under relaxation factors, have been incorporated in the respective subsections whenever possible.

The last chapter (5) contains the conclusions taken from this work, and possible future work approaches.

Chapter 2

Finite Volume Method

In this chapter, the finite volume method will be described. Initially, the governing equations of fluid flow and the scalar transport are stated, followed by an example on how the finite volume discretization is applied to these equations.

2.1 Governing Equations

In this section the governing equations will be detailed, along with the several simplifications that are allowed for the different regimes of fluid flow at study.

Compressible Flow

The set of equations that governs compressible flow of newtonian fluids is comprised by the continuity equation, the Navier-Stokes equations and the energy equation, along with an equation of state (as the ideal gas equation):

$$D\left(\rho\mathbf{u}\right) = \frac{\partial \rho}{\partial t} + \nabla\cdot\left(\rho\mathbf{u}\right) = 0, \tag{2.1}$$

$$\frac{\partial \rho\mathbf{u}}{\partial t} + \nabla\cdot\left(\rho\mathbf{u}\otimes\mathbf{u}\right) = -\nabla p + \nabla\cdot\left(\boldsymbol{\tau}\right) + \rho\mathbf{g}, \tag{2.2}$$

$$\frac{\partial \rho h_t}{\partial t} + \nabla\cdot\left(\rho h_t\mathbf{u}\right) = \nabla\cdot\left(\boldsymbol{\tau}\cdot\mathbf{u}\right) - \nabla\cdot\left(k\nabla T\right) + \frac{\partial p}{\partial t} + \mathbf{f}_b\cdot\mathbf{u} + q_v, \tag{2.3}$$

$$p = \rho R T, \tag{2.4}$$

here $\boldsymbol{\tau} = \left(\mu\nabla\boldsymbol{u} + \mu\nabla\boldsymbol{u}^T\right) - \frac{2}{3}\mu\left(\left(\nabla\cdot\boldsymbol{u}\right)\boldsymbol{I}\right)$ is the viscous stress tensor. It can be simplified for flows in which the viscosity is considered to be approximately constant, leading to: $\nabla\cdot\boldsymbol{\tau} = \mu\left(\nabla^2\boldsymbol{u} + \frac{1}{3}\nabla\left(\nabla\cdot\boldsymbol{u}\right)\right)$.

In the energy equation, $h_t = c_p T + \frac{1}{2}||\mathbf{u}||^2$ is the total enthalpy and $\nabla\cdot\left(\boldsymbol{\tau}\cdot\mathbf{u}\right)$ is the energy lost due to viscous effects. This dissipation of energy is usually neglected for subsonic flows.

According to [45], by assuming the fluid can be modelled as an ideal gas, the energy equation 2.3

can be written as a function of temperature:

$$\frac{\partial \rho c_p T}{\partial t} + \nabla \cdot (\rho c_p T \mathbf{u}) = \nabla \cdot (k \nabla T) + Dp + \lambda \Psi + \mu \Phi + \dot{q}_v \tag{2.5}$$

where $\lambda\Psi + \mu\Phi$ are the viscous terms, and Dp is the material derivative of pressure. This equation is useful, a it allows the direct computation of temperature instead of the total enthalpy. However, the viscous terms feature several 2^{nd} order partial derivatives, rendering them a more challenging discretization than the ones that appear in equation 2.3, whose 2^{nd} order derivatives vanish when integrated in a finite volume, by using the divergence theorem.

All in all, equation 2.5 was mostly useful to solve inviscid flow problems and to help the verification of the implementation of equation 2.3.

The previous equations can be simplified for steady-state flow, leading to:

$$\nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.6}$$

$$\nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \nabla \cdot (\boldsymbol{\tau}) + \rho \mathbf{g}, \tag{2.7}$$

$$\nabla \cdot (\rho h_t \mathbf{u}) = \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{u}) - \nabla \cdot (k \nabla T) + \mathbf{f}_b \cdot \mathbf{u} + q_v, \tag{2.8}$$

and the energy equation written as a function of temperature to:

$$\nabla \cdot (\rho c_p T \mathbf{u}) = \nabla \cdot (k \nabla T) + \mathbf{u} \cdot \nabla p + \lambda \Psi + \mu \Phi + \dot{q}_v \tag{2.9}$$

Incompressible Flow

For incompressible flow, density drops from the momentum and continuity equation, allowing further simplification of the governing equations:

$$\nabla \cdot (\mathbf{u}) = 0, \tag{2.10}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u}, \tag{2.11}$$

$$\frac{\partial T}{\partial t} + \nabla \cdot (c_p T \mathbf{u}) = \nabla \cdot (\alpha \nabla T) + q_T, \tag{2.12}$$

here $\nu \nabla^2 \mathbf{u}$ represents the incompressible flow viscous stresses and $\alpha = \frac{k}{\rho c_p}$ is the thermal diffusivity.

Equation 2.12 is the transport equation for temperature. This equation is useful for solving problems in which temperature gradients can influence the flow, but don't affect density significantly, such as problems where the Boussinesq approximation is valid. This approximation also uses an additional vertical source term $\beta g \Delta T$ in the momentum equations to account for the influence of density changes driven by the temperature gradients. With this extra term, the momentum equations have the following form:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \beta \mathbf{g} \Delta T, \tag{2.13}$$

where $\mathbf{g} = (0, 1)$.

Scalar Transport Equation

A general transport/conservation equation, valid in both compressible and incompressible flow regimes, can be written for generic scalar:

$$\frac{\partial \rho\phi}{\partial t} + \nabla \cdot (\rho\phi\mathbf{u}) = \nabla \cdot (\Gamma \nabla \phi) + q_\phi, \tag{2.14}$$

The several terms symbolize the convective $\nabla \cdot (\rho\phi\mathbf{u})$ and diffusive $\nabla \cdot (\Gamma\nabla\phi)$ transport, the temporal fluctuations $\frac{\partial \rho\phi}{\partial t}$, and the possible sources of ϕ in an infinitesimal domain.

It is noteworthy that all the equations presented in sections 2.1 and 2.1 are conservation equations. Hence, they can be looked at as being specific employments of the general transport equation, a fact, which, in itself, explains this equation's significance.

Since equation 2.14 has been written for a generic scalar, it will be called into play to exemplify the general discretization procedure of the finite volume method in section 2.2.

2.2 Finite Volume Discretization

To solve the equations presented in section 2.1 numerically, they must be treated in some way. This treatment is a discretization, which turns these continuous equations into discrete ones that can be handled by a computer.

The discretization procedure can be obtained by several means: using finite differences, the finite element method, the finite volume method, or other numerical methods.

This work focuses only on the finite volume method, which is made up of several steps:

1. Discretization of the domain into several finite volumes (mesh generation).
2. Discretization of the equations in a finite volume.
3. Imposition of boundary conditions in the boundary CVs.
4. System of equations assembly in function of the computational values.

In this section all the steps of the finite volume method will be looked into, by exemplifying them on the general transport equation.

2.2.1 Finite Volume Integration

By integrating equation 2.14 in volume:

$$\int_V \frac{\partial \rho\phi}{\partial t} \mathrm{d}V + \int_V \nabla \cdot (\rho\phi\mathbf{u}) \,\mathrm{d}V = \int_V \nabla \cdot (\Gamma\nabla\phi) \,\mathrm{d}V + \int_V \dot{q}_\phi \mathrm{d}V, \tag{2.15}$$

afterwards, by employing the divergence theorem, the convective and the diffusive term become surface integrals and the divergence is dropped, leaving behind the inner product of the velocity with the face normal $\mathbf{n}_f$. The temporal and the source terms,though, must stay as volume integrals, since there is no way to simplify them any further. This leads to:

$$\underbrace{\int_V \frac{\partial \rho\phi}{\partial t} \mathrm{d}V}_{\text{temporal term}} + \underbrace{\int_S \rho\phi \left(\mathbf{u} \cdot \mathbf{n}_f\right) \mathrm{d}S}_{\text{convective term}} = \underbrace{\int_S \Gamma \left(\nabla\phi \cdot \mathbf{n}_f\right) \mathrm{d}S}_{\text{diffusive term}} + \underbrace{\int_V \dot{q}_\phi \mathrm{d}V}_{\text{source term}} \tag{2.16}$$

and now, this equation is ready to be discretized in a generic finite volume or computational cell.

2.2.2 Domain Discretization

To solve equation 2.16 numerically, it has to be discretized both in space and time. The domain discretization in this work was achieved by using cartesian and uniform grids, which are composed by rectangular or square elements. This choice of grid was made to facilitate the implementation in Matlab, since, if unstructured grids were to be implemented, special schemes would have had to be implemented, something that is beyond the scope of this work.

As mentioned, the major advantage of employing uniform grids is the ease with which they can be generated, needing only (in this case) the geometry's dimensions and the number of control volumes in the x and y directions (no connectivity matrix is required). Another plus in using these grids is that many of schemes can be simplified, like the linear differencing scheme which becomes an average.

To apply these grids to flows around objects, like the supersonic flow over a forward facing step, or the flow in a duct with a baffle (figure 1.1), a vector specifying which nodes and faces are included in the object is generated, as a way to take that information into account when building the equation matrices. Thus, the objects were included in the mesh, but the equations weren't solved in the control volumes that are part of that area, this is illustrated in figure 2.1, where the CVs lying inside the step are excluded from the calculations.

The major drawbacks of using uniform and cartesian grids are the limitation on the number of test cases than can be ran, and their accuracy-to-number-of-control volumes ratio. The reason for this is that non-uniform grids can have different sized control volumes, which grants them the ability to have an increased number of nodes in regions of particular interest.

Another grid related choice, was about the arrangement of variables. A staggered arrangement in itself would prevent the pressure checker board issue without needing the Rhie and Chow interpolation for face velocities, but, it would be more cumbersome to implement, since two grids are actually needed, one for velocity and one for pressure.

2.2.3 Finite Volume Discretization

By taking equation 2.16 and applying it to a finite volume, the surface integrals can be approximated

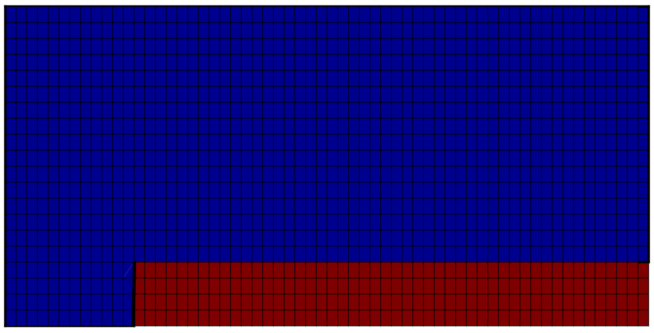


Figure 2.1: Example of application of a cartesian and uniform grid applied to a non-rectangular geometry case.

by a sum over all faces, while the volume integrals can be approximated by taking the product between the CV volume the integrated quantity's value in the CV center, becoming:

$$V\frac{\partial\rho\phi}{\partial t}+\sum_{f=1}^{F}\rho_f\phi_f\left(\mathbf{u}_f\cdot\mathbf{n}_f\right)A_f=\sum_{f=1}^{F}\Gamma_f\left(\nabla\phi\cdot\mathbf{n}_f\right)_f A_f+V\dot{q}_\phi \tag{2.17}$$

This equation written in a form that allows it to be applied to unstructured grids, but can be rewritten for cartesian uniform grids. This will be done progressively as each of the terms is discretized, and the several options of schemes are presented. Furthermore, the demonstration of how the discretization schemes are applied will only be exemplified on the east face of a CV (figure 2.2) , since it would be redundant to do it for every face.

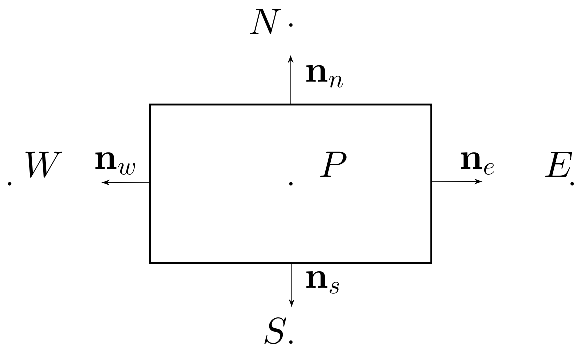


Figure 2.2: Control volume (CV) of a cartesian and uniform grid.

Convective Term

The convective term requires the value of ϕ at the face to be determined. Rewriting the convective term in cartesian coordinates:

$$\sum_{f=1}^{F} \rho_f \phi_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) A_f = F_e\phi_e + F_w\phi_w + F_s\phi_s + F_n\phi_n \tag{2.18}$$

where $F_f = \rho_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) Af$ is the mass flux at face f.

Upwind Differencing Scheme A very common way to discretize this term is by using the upwind differencing scheme (UDS), which approximates ϕ_f as being equal to ϕ from the adjacent upstream CV nodes:

$$F_e\phi_e = Fe \begin{cases} \phi_E : Fe < 0 \\ \phi_P : Fe > 0 \end{cases} \tag{2.19}$$

The UDS is first order accurate, and the major upside of using this scheme is that it can be implemented implicitly, due to the fact that it doesn't affect the dominance of the matrix's main diagonal. It is highly diffusive, meaning that when simulating flows that include discontinuities, large gradients, or peaks, these important features of the flow become smeared, thus diminishing the accuracy. This does not mean that this scheme shouldn't be used to simulate these flows, due to the importance of the added stability it brings to the simulations, allied to the fact that it prevents wiggles in the final solution cannot be emphasized enough. Yet, in flows where greater accuracy can be achieved without risking instabilities, other schemes might be better alternatives.

Another way to formulate the upwind scheme is by writing it as an extreme case of a blending scheme:

$$F_e\phi_e = Fe\left(\mu\phi_E + (1-\mu)\,\phi_P\right) \tag{2.20}$$

Where μ is chosen to be 0 or 1, according to the direction of the flow F_e:

$$\phi_e = \begin{cases} \phi_E : \mu = 1 \\ \phi_P : \mu = 0 \end{cases} \tag{2.21}$$

Linear Interpolation An alternative to the UDS is the linear interpolation scheme (LIN). In the same framework presented with 2.20, the linear interpolation scheme calculates the face value of ϕ by interpolating it from the adjacent nodes, which can be done by weighing the distance to each of the nodes. In the case of a uniform grid ($\mu = 0.5$), this means averaging their values:

$$F_e\phi_e = Fe\left(\frac{\phi_E + \phi_P}{2}\right) \tag{2.22}$$

Being a second order scheme, its accuracy is superior to the UDS. However, this improvement comes at the expense of stability to an extent that impedes its implicit implementation, and, in flows where discontinuities appear, oscillations in the final solution are commonplace.

Gamma NVD Scheme A third alternative, the Gamma scheme presented by [4], allows the upwind scheme to be used in regions where gradients are substantial and LIN wherever this is not the case. Essentially, the NVD (Normalized Variable Diagram) family of schemes is similar to the TVD (Total Variation Diminishing) family in concept, but has a different formulation: the Normalized Variable Approach (NVA), which, does not guarantee convergence, as TVD schemes do, but guarantees boundedness.

To implement this scheme:

- Find the face flow direction to identify the centered (C), upwind (U) and downwind nodes (D);
- Calculate the normalized value of the variable:$\tilde{\phi}_C = \frac{\phi_C - \phi_U}{\phi_D - \phi_U}$;
- According to the value of $\tilde{\phi}_C$ select the discretization scheme:
  - If $\tilde{\phi}_C \leq 0$ or $\tilde{\phi}_C \geq 1$, use the upwind scheme: $\phi_f = \phi_C$;
  - If $\beta_m \leq \phi$ or $\tilde{\phi}_C \leq 1$, use LIN: $\phi_f = \frac{\phi_C + \phi_D}{2}$;
  - If $0 \leq \tilde{\phi}_C$ or $\tilde{\phi}_C \leq \beta_m$, blend the two schemes: $\phi_f = \left(1 - \frac{\tilde{\phi}_C}{\beta_m}\right)\phi_C + \frac{\tilde{\phi}_C}{\beta_m}\phi_D$;

The major upside of this scheme, when compared to other NVDs is that there is no single limit value for when the CDS or UDS are chosen. Instead, there is range of values of $\tilde{\phi}_C$, inside which a blending between the UDS and CDS is made, consequently decreasing the perturbations generated by the shifting between schemes. The value of β controls how wide the blending range is, so a higher value of β increases the range of values where blending occurs, lessening the range where central differences are used.

Deferred Correction

To implement convective schemes other than the UDS, an alternative strategy must be followed, for the diagonal dominance of the matrices cannot be violated if stability is to be expected. This special proceeding is the implementation a deferred correction, which is based on the idea of maintaining stability by building the matrices with the upwind scheme, while transferring the influence of the the higher order scheme to the source term. This can be done in the following way:

$$C_{UDS}\phi^{n+1} = \left(C_{UDS} - C_{HOS}\right)\phi^n \tag{2.23}$$

Where the superscript represents the iteration in which the value of ϕ is being considered, and C is the contribution of either the higher order scheme (HOS) or the upwind scheme. Upon reaching convergence, the values of ϕ in consecutive iterations are very close ($\phi^{n+1} \approx \phi^n$), making it so that the two UDS contributions cancel each other out, becoming overshadowed the contribution of the higher order scheme as the only one.

Diffusive Term

In the diffusive term, the face derivative of ϕ is computed during the discretization procedure. Rewriting the diffusive term of equation 2.17 for cartesian grids:

$$\sum_{f=1}^{F} \Gamma_f \left(\nabla\phi \cdot \mathbf{n}_f\right)_f A_f = \Gamma_e \left(\frac{\partial\phi}{\partial x}\right)_e A_e - \Gamma_w \left(\frac{\partial\phi}{\partial x}\right)_w A_w + \Gamma_n \left(\frac{\partial\phi}{\partial y}\right)_n A_n - \Gamma_s \left(\frac{\partial\phi}{\partial y}\right)_s A_s \tag{2.24}$$

The negative coefficients of faces w and s come from the fact that $\mathbf{n}_s = (0, -1)$ and $\mathbf{n}_w = (-1, 0)$.

The most common way to discretize the diffusive terms is via the central differencing scheme, where the derivatives are equated by subtracting the value of ϕ in those nodes over the distance between them dx:

$$\Gamma_e \left(\frac{\partial\phi}{\partial x}\right)_e A_e = \Gamma_e \frac{\phi_E - \phi_P}{dx} A_e \tag{2.25}$$

This scheme is second order accurate and is very commonly used. It can be implemented implicitly, because the coefficient that is added to the matrix's diagonal and to the non diagonal entries are equal: $\Gamma_f \frac{1}{dx}$. The fact that only cartesian uniform grids were used permits $\mathbf{n}_f$ and $\left(\nabla\phi \cdot \mathbf{n}_f\right)_f$ to be aligned, because the nodes of the CVs that share the face in consideration are aligned as well, thus revoking the need for gradient reconstruction techniques.

Temporal Term

In this work, the temporal discretization was made utilizing the implicit Euler scheme:

$$V\frac{\partial\rho\phi}{\partial t} + R = V\frac{(\rho\phi)^{n+1} - (\rho\phi)^n}{\Delta t} + R^{n+1} \tag{2.26}$$

In this equation R represents the remaining terms, n the present time step and $n+1$ the future time (to which the equation is being solved to). This scheme adds an extra contribution to the diagonal of the matrices, increasing stability and sometimes allowing the incrementation of the values of the under relaxation factors. Between time steps constant time iterations are usually required to decrease the value of the residuals.

An alternative scheme is the explicit Euler scheme, which only differs from the implicit Euler scheme in the way the remaining terms are treated, which is explicitly:

$$V\frac{\partial\rho\phi}{\partial t} + R = V\frac{(\rho\phi)^{n+1} - (\rho\phi)^n}{\Delta t} + R^{n} \tag{2.27}$$

This scheme only requires one iteration per time step, but is proner to instabilities, if the time step is too big. This can be controlled by the Courant number $C = \frac{u\Delta t}{\Delta x}$, which is usually kept below at least 1.

Other Terms

In the momentum, energy and continuity equations there are additional terms that don't fit the description of the ones present in the scalar transport equation, thus upholding the need for this extra section. All of these terms are treated as explicit, otherwise, they would turn the process of solving the equations much more cumbersome, or even impossible. In this section **u** is assumed to be a vector quantity with components (u_x, u_y).

CV Centered Gradient The first of the extra terms whose calculation method will be discussed is the CV centered gradient $(\nabla\mathbf{u})_p$. Integrating it over the control volume and putting Gauss's theorem to use:

$$\int_V \nabla\mathbf{u}\,\mathrm{d}V = \int_S \mathbf{u}_f\,\mathrm{d}S = \sum_{f=1}^{F} A_f\mathbf{u}_f \tag{2.28}$$

On the other hand, if the gradient is integrated in volume without using this theorem:

$$\int_V \nabla\mathbf{u}\,\mathrm{d}V = V\nabla\mathbf{u} \tag{2.29}$$

Noting that these equations must be equivalent:

$$\nabla\mathbf{u} = \frac{1}{V}\sum_{f=1}^{F} A_f\mathbf{u}_f \tag{2.30}$$

In uniform grids, this is the same as calculating the CV centered gradient by considering the CV centered values of the neighbouring CVs:

$$\frac{\partial u_x}{\partial x} = \frac{u_E - u_W}{2dx} \tag{2.31}$$

$$\frac{\partial u_y}{\partial y} = \frac{u_N - u_S}{2dy} \tag{2.32}$$

where dx and dy are the dimensions of the CV in the x and y directions.

The CV centered gradient is also often used to aid in the computation of other extra terms.

CV Centered Divergence Calculating the cell centered divergence $(\nabla\cdot\mathbf{u})_p$ is possible by calculating the trace of the CV centered gradient's matrix: $\begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_x}{\partial y} \\ \frac{\partial u_y}{\partial x} & \frac{\partial u_y}{\partial y} \end{bmatrix}$

This becomes evident when writing the definition of the divergence $(\nabla\cdot\mathbf{u})_p = \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right)_p$. The CV centered divergence can be used to calculate yet another extra term: the face centered divergence.

Face Centered Divergence In order to calculate this value in uniform grids, the average of the CV centered divergences of two CVs that share the face in question must be taken:

$$(\nabla\cdot\mathbf{u})_f = \frac{(\nabla\cdot\mathbf{u})_E + (\nabla\cdot\mathbf{u})_P}{2} \tag{2.33}$$

Where the procedure to calculate the cell centered divergences has been detailed above.

Derivatives Parallel to Faces The computation of this value can be done in several ways, such as averaging the values of the 4 CVs surrounding the two vertices that limit the face in question, yet in this work, the antidiagonal values from the CV centered gradient were be taken and averaged, just like for the face centered divergence:

$$\left(\frac{\partial u_x}{\partial y}\right)_e = \frac{\left(\frac{\partial u_x}{\partial y}\right)_E + \left(\frac{\partial u_x}{\partial y}\right)_P}{2} \tag{2.34}$$

2.2.4 Boundary Conditions

To solve the governing equations of fluid flow, boundary conditions must be specified. This section will only feature the mathematical boundary conditions, as their correspondent physical boundary conditions will be further explained in section 3. Each of the boundary conditions will be described, followed by an exemplification on the east face of a control volume for the discretized transport equation. Assuming steady flow and a null source term (these assumptions do not affect the way in which the implementation is done):

$$Fe\phi_e - \Gamma_e Ae\left(\nabla\phi\right)_e = Fe\begin{cases}\phi_E : Fe < 0\\ \phi_P : Fe > 0\end{cases} - \Gamma_e\frac{\phi_E - \phi_P}{dx}A_e = 0 \tag{2.35}$$

It is also assumed that these terms will be discretized via upwind scheme and central differences.

Dirichlet Boundary Condition

In the Dirichlet boundary condition, the value of the variable at the boundary is specified $\phi_e = \phi_b$. This transforms the convective term into a source term, and the diffusive term becomes divided into an explicit ϕ_e and implicit ϕ_P component:

$$-\Gamma_e Ae\left(\nabla\phi\right)_e = -\Gamma_e\frac{\phi_e - \phi_P}{\frac{dx}{2}}A_e \tag{2.36}$$

The final equation becomes:

$$2\Gamma_e\frac{\phi_P}{dx}A_e = 2\Gamma_e\frac{\phi_b}{dx}A_e - Fe\phi_b \tag{2.37}$$

Neumman Boundary Condition

The Neumann boundary condition indicates that the value of $\left(\nabla\phi\right)_e$ is specified (k will be assumed to be that value). Thus. the diffusive term becomes fully explicit, while the convective term can be calculated implicitly by:

$$\left(\nabla\phi\right)_e = k \Rightarrow \frac{\phi_e - \phi_P}{\frac{dx}{2}} = k \Rightarrow \phi_e = k\frac{dx}{2} + \phi_P \tag{2.38}$$

In the majority of uses, $k = 0$, which implies that the convective term can also be fully implicit: $\phi_e = \phi_P$.

Robin Boundary Condition

The Robin boundary condition dictates that the the sum of the boundary gradient and value are defined $Fe\phi_e - \Gamma_e \frac{\phi_E - \phi_P}{dx} A_e = k$. This value is included the source term, making the implicit contributions on the east face disappear, thus forcing the value of ϕ_P to be calculated from the interior nodes, after which, the value of ϕ_e can be computed from the specified value of k. In this case, the procedure will be exemplified using two faces (east and west):

$$Fe\phi_e - \Gamma_e \frac{\phi_E - \phi_P}{dx} A_e - Fw\phi_w + \Gamma_w \frac{\phi_W - \phi_P}{dx} A_w = 0 \tag{2.39}$$

Introducing the Robin boundary condition $a\left(\nabla\phi\right)_e + b\phi_e = k$:

$$-Fw \begin{cases} \phi_P : Fw < 0 \\ \phi_W : Fw > 0 \end{cases} + \Gamma_w \frac{\phi_W - \phi_P}{dx} A_w = -k \tag{2.40}$$

Regardless of the flow direction, the value of ϕ_P is always computable. Afterwards, the value of ϕ_e can be determined by inputting ϕ_P in:

$$Fe\phi_e - \Gamma_e \frac{\phi_E - \phi_P}{dx} A_e = k \tag{2.41}$$

2.2.5 Discretization of the Governing Equations for a Cartesian and Uniform Grid

In this section, the finite volume discretization of the governing equations will be demonstrated. The equations will be presented in a form achieved by integrating in a control volume, as discussed in sections 2.2.1 and 2.2.3.

Compressible Flow Momentum Equations

The steady state compressible flow momentum equation integrated over general control volume is:

$$\sum_{f=1}^{F} A_f \rho_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) u_f = -V\left(\nabla P\right) + \sum_{f=1}^{F} \left(\boldsymbol{\tau}_f \cdot \mathbf{n}_f\right) + V\rho\boldsymbol{g} \tag{2.42}$$

Exemplifying for the u velocity in a cartesian and uniform grid:

$$F_e u_e - F_w u_w + F_n u_n - F_s u_s = V\left(\nabla p\right)_P + A_e \tau_e - A_w \tau_w + A_n \tau_n - A_s \tau_s + V_P \rho g_x \tag{2.43}$$

The convective and pressure terms are ready to be discretized using the previously presented schemes, but not the stress term, which will now be further looked into.

Compressible Flow Viscous Stress The variable viscosity compressible flow viscous stress term is best analysed by writing it in a matrix form:

$$(\boldsymbol{\tau}_f \cdot \mathbf{n}_f) = \begin{bmatrix} 2\mu \overbrace{\frac{\partial u}{\partial x}}^{\text{implicit}} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) & \mu \overbrace{\frac{\partial u}{\partial y}}^{\text{implicit}} + \mu\frac{\partial v}{\partial x} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \\ \mu\frac{\partial u}{\partial y} + \mu \underbrace{\frac{\partial v}{\partial x}}_{\text{implicit}} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) & 2\mu \underbrace{\frac{\partial v}{\partial y}}_{\text{implicit}} - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \end{bmatrix}_f \begin{bmatrix} nx \\ ny \end{bmatrix} \tag{2.44}$$

where the terms that were treated implicitly in this work have been specified, and the rest of the terms were calculated explicitly as a source term. Thus, for the u momentum equation in the north face of a control volume:

$$\tau_n = \mu_n \underbrace{\left(\frac{\partial u}{\partial y}\right)_n}_{\text{implicit}} + \mu_n \frac{\partial v}{\partial x} - \frac{2}{3}\mu_n\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)_n \tag{2.45}$$

The implicit term is usually discretized using central differences, and the value of μ_n can be calculated by using an interpolation scheme. It is important to highlight that the x derivative is parallel to face n, and the divergence is face centered. The value of these terms can be computed as described in section2.2.3. It is important that the divergence is calculated fully explicitly, as treating one of the terms implicitly would mean that the accuracy with which it would be calculated would be different from the one with which the explicit term would be calculated, i.e. calculating $\frac{\partial v}{\partial x}$ with the CDS implicitly does not guarantee the same order of accuracy as treating $\frac{\partial u}{\partial y}$ explicitly with the scheme from section 2.2.3. This could lead to an imbalance, that, even in incompressible flow, could lead to wrong results, as it would not guarantee the divergence's value to be null.

If constant viscosity can be considered, the viscous stress tensor can be simplified, as seen in section 2.1:

$$(\boldsymbol{\tau}_f \cdot \mathbf{n}_f) = \begin{bmatrix} \mu \overbrace{\frac{\partial u}{\partial x}}^{\text{implicit}} + \frac{1}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) & \mu \overbrace{\frac{\partial u}{\partial y}}^{\text{implicit}} + \frac{1}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \\ \mu \underbrace{\frac{\partial v}{\partial x}}_{\text{implicit}} + \frac{1}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) & \mu \underbrace{\frac{\partial v}{\partial y}}_{\text{implicit}} + \frac{1}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) \end{bmatrix}_f \begin{bmatrix} nx \\ ny \end{bmatrix} \tag{2.46}$$

It is remarkable that there are no derivatives which are parallel to faces outside of the velocity divergences, whose calculation is part of the face divergence source term. Other than that, the discretization procedure is the same as for the variable viscosity case.

Unsteady Momentum Equation The unsteady version of equation 2.42 can be written by adding the unsteady term:

$$V\frac{\partial \rho u}{\partial t}+\sum_{f=1}^{F}A_f\rho_f\left(\mathbf{u}_f\cdot\mathbf{n}_f\right)u_f=-V\left(\nabla P\right)+\sum_{f=1}^{F}\left(\boldsymbol{\tau_f}\cdot\mathbf{n}_f\right)+V\rho\boldsymbol{g} \tag{2.47}$$

This term can be discretized using a temporal discretization scheme (2.2.3).

Incompressible Flow Momentum Equations

Integrating equation 2.11 over a control volume of a cartesian and uniform grid are:

$$F_eu_e-F_wu_w+F_nu_n-F_su_s=V\left(\nabla p\right)_P+A_e\tau_e-A_w\tau_w+A_n\tau_n-A_s\tau_s+V_PS_u \tag{2.48}$$

This time around, the face flux is the volumetric one $F_f=A_f\left(u_f\cdot\mathbf{n}_f\right)$, which can be discretized using the UDS. The viscous term is is majorly simplified, relative to the compressible viscous term ($\tau_f=\left(\nabla u\right)_f$), and can be discretized implicitly using the central differencing scheme. The pressure gradient can be computed using any of the CV centered gradient schemes.

Unsteady Momentum Equation Like the unsteady compressible equation, the only time-related term that needs to be added:

$$V\frac{\partial u}{\partial t} \tag{2.49}$$

Note that density is constant, so there is no need to calculate it.

Boussinesq Approximation If the Boussinesq approximation is to be assumed, the source term V_PS_u becomes $V\beta\boldsymbol{g}\Delta T$, where ΔT can be computed as T_n-T_s in the case of the v momentum equation, and as T_w-T_e in for the u momentum equation.

Total Enthalpy Energy Equation

By integrating the steady form equation 2.3 in a control volume, the following equation is obtained:

$$\sum_{f=1}^{F}A_f\rho_f\left(\mathbf{u}_f\cdot\mathbf{n}_f\right)h_{tf}=\sum_{f=1}^{F}A_f\left(\boldsymbol{\tau}\cdot\mathbf{u}\right)_f\cdot\mathbf{n}_f-\sum_{f=1}^{F}A_f\left(k\nabla T\right)_f\cdot\mathbf{n}_f+V\left(\mathbf{f}_b\cdot\mathbf{u}\right) \tag{2.50}$$

The enthalpy will now be divided in its two components $h_Tf=cpT_f+\frac{||\mathbf{u}||^2}{2}$, in order to be able to write the equation as a function of temperature. By doing this, the kinetic energy component of the enthalpy can be included in a source term S_{kin}:

$$\sum_{f=1}^{F} A_f \rho_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) c_p T_f = \sum_{f=1}^{F} A_f \left(\boldsymbol{\tau} \cdot \mathbf{u}\right)_f \cdot \mathbf{n}_f - \sum_{f=1}^{F} A_f \left(k \nabla T\right)_f \cdot \mathbf{n}_f + V \left(\mathbf{f}_b \cdot \mathbf{u}\right) - S_{kin} \tag{2.51}$$

$$S_{kin} = \sum_{f=1}^{F} F_f \frac{||\mathbf{u}||_f^2}{2} \tag{2.52}$$

In a cartesian and uniform grid, this equation can be written as:

$$F_e T_e - F_w T_w + F_n h_{tn} - F_s T_s = A_e \sigma_e - A_w \sigma_w + A_n \sigma_n - A_s \sigma_s - A_e \left(k \nabla T\right)_e + A_w \left(k \nabla T\right)_w - \tag{2.53}$$
$$- A_n \left(k \nabla T\right)_n + A_s \left(k \nabla T\right)_s + V_P \left(\mathbf{f}_b \cdot \mathbf{u}\right)_P - S_{kin}$$

$$S_{kin} = F_e \frac{||u||_e^2}{2} - F_w \frac{||u||_w^2}{2} + F_n \frac{||u||_n^2}{2} - F_s \frac{||u||_s^2}{2} \tag{2.54}$$

The steps followed in this work to discretize the viscous dissipation of energy $\sigma_f = \left(\boldsymbol{\tau} \cdot \mathbf{u}\right)_f$ will be detailed in the following section.

Viscous Dissipation in the Energy Equation It is useful to write the energetic viscous dissipation in its matrix form, as it was done for the momentum equations' viscous stress:

$$\sigma_f = \begin{bmatrix} nx & ny \end{bmatrix} \begin{bmatrix} 2\mu_f \left(\frac{\partial u}{\partial x}\right)_f - \frac{2}{3}\mu_f \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)_f & \mu_f \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)_f - \frac{2}{3}\mu_f \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)_f \\ \mu_f \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)_f - \frac{2}{3}\mu_f \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)_f & 2\mu_f \left(\frac{\partial v}{\partial y}\right)_f - \frac{2}{3}\mu_f \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)_f \end{bmatrix} \begin{bmatrix} u_f \\ v_f \end{bmatrix} \tag{2.55}$$

Which becomes a scalar, after the matrix multiplication:

$$\sigma_f = \left(2\mu \frac{\partial u}{\partial x} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)_f u_f n_x + \left(\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)_f v_f n_x + \tag{2.56}$$
$$+ \left(\mu \frac{\partial u}{\partial y} + \mu \frac{\partial v}{\partial x} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)_f u_f n_y + \left(2\mu \frac{\partial v}{\partial y} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)_f v_f n_y$$

Exemplifying for the east face $n_f = (1, 0)$:

$$\sigma_e = \left(2\mu \frac{\partial u}{\partial x} - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)_e u_e + \left(\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) - \frac{2}{3}\mu \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\right)_e v_e \tag{2.57}$$

It should be pointed out that all of the values are calculated explicitly and evaluated at the face locations, leading to the necessity to calculate velocities which are parallel to faces (and perpendicular to their normal vector, as seen in figure 2.3). To obtain these velocity values, the upwind scheme was used, while the face centered divergences and the remaining derivatives were calculated using the schemes from section 2.2.3 and central differences, respectively. It is restated that this term is only relevant in supersonic flow, otherwise the velocities aren't large enough to make it non-negligble.

Unsteady Total Enthalpy Equation There are two unsteady terms must be included in this equation to allow the resolution of time dependent problems: the pressure derivative $V\frac{\partial p}{\partial t}$ and the enthalpy

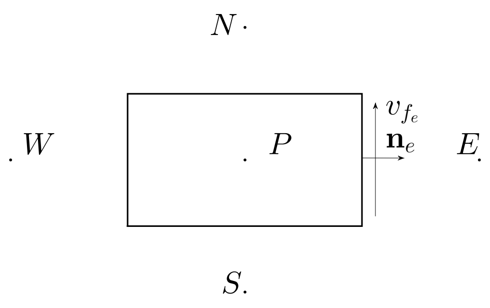


Figure 2.3: Face velocity perpendicular to its normal.

derivative $V\frac{\partial \rho h_t}{\partial t}$. The pressure term must be calculated explicitly, but part of the enthalpy term has to be calculated implicitly. To do this, once again, the enthalpy must be divided into its two components energetic components: $h_T f = cpT_f + \frac{||\mathbf{u}||^2}{2}$ and, by applying the schemes from section 2.2.3. The internal energy becomes divided into an explicit and an implicit component, while the kinetic energy is always computed explicitly:

$$V\frac{\partial\left(\rho c_p T\right)}{\partial t} + \sum_{f=1}^{F} A_f \rho_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) h_{tf} = \sum_{f=1}^{F} A_f \left(\boldsymbol{\tau}\cdot\mathbf{u}\right)_f \cdot \mathbf{n}_f - \sum_{f=1}^{F} A_f \left(k\nabla T\right)_f \cdot \mathbf{n}_f + \\ + V\frac{\partial p}{\partial t} + V\left(\mathbf{f}_b\cdot\mathbf{u}\right) - V\frac{\partial\left(\frac{||\mathbf{u}||^2}{2}\right)}{\partial t} \tag{2.58}$$

Temperature Energy Equation

The energy equation written in terms of temperature (equation 2.5) in a control volume becomes:

$$\sum_{f=1}^{F} A_f \rho_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) c_p T_f = V\left(\lambda\Psi + \mu\Phi\right) - \sum_{f=1}^{F} A_f \left(k\nabla T\right)_f \cdot \mathbf{n}_f + V\left(\mathbf{u}\nabla p\right) + V\left(\mathbf{f}_b\cdot\mathbf{u}\right) \tag{2.59}$$

The viscosity terms were always neglected utilizing this equation, so their discretization will not be addressed in this section. The major difference, when compared to the enthalpy energy equation, lies in the pressure gradient term: $V\left(\mathbf{u}\nabla p\right)$. The velocities must be evaluated at the CV centers, and the CV centered gradient with either of the schemes described in section 2.2.3.

The discretization of most of the remaining terms is very similar to the equation 2.54 (temporal terms included), except no separation of the transported variable is needed, since it already is temperature.

Incompressible Flow Temperature Transport Equation

Like it was mentioned in section 2.1, this equation is useful when putting to use the Boussinesq approximation. By taking equation 2.16, replacing ϕ by T, and disregarding the density changes:

$$\sum_{f=1}^{F} c_p T_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) A_f = \sum_{f=1}^{F} \Gamma_f \left(\nabla T \cdot \mathbf{n}_f\right)_f A_f + V \dot{q}_T \tag{2.60}$$

It is straightforward to expand this equation in a cartesian and uniform grid:

$$F_e c_p T_e - F_w c_p T_w + F_n c_p T_n - F_s c_p T_s = \sum_{f=1}^{F} \Gamma_f \left(\nabla T \cdot \mathbf{n}_f\right)_f A_f + V_P \dot{q}_T \tag{2.61}$$

Final Form of the Equations

After discretizing the governing equations, they can be written in a general way, which is the one used to solve them. Each line of the system of linear equations is represented by:

$$A_P \phi_P + \sum_{f=1}^{F} A_L \phi_L = S_\phi \tag{2.62}$$

here, A_P and A_L are the (implicit) coefficients, which are multiplying by the CV centered values of either ϕ_P or $\phi_E, \phi_W, \phi_N, \phi_S$, respectively. The term S_ϕ is the source term where all the explicit terms and boundary conditions are included. Additionally, an under relaxation factor α is often included in these equations:

$$\frac{A_P}{\alpha} \phi_P + \sum_{f=1}^{F} A_L \phi_L = S_\phi + \frac{(1-\alpha)}{\alpha} A_P \phi_P^* \tag{2.63}$$

In this final equation, the $\frac{A_P}{\alpha}$ represents the under relaxed sum of the coefficients multiplied by ϕ_P, and ϕ_P^* is the value of ϕ_P resulting from the last iteration.

Chapter 3

Segregated Solvers for Incompressible and Compressible Flows

In this chapter, the structure of the algorithm and its implementation will be analysed. The original incompressible SIMPLE algorithm will be first explained, followed by the all-Mach version. This was the order in which these two algorithms were implemented throughout this work, so it will not only highlight the major differences between the two algorithms, but will also provide a glimpse on the implementation perspective.

3.1 SIMPLE Algorithm Implementation

In this section, a walkthrough on how the SIMPLE algorithm was implemented will be given. The intention is to give a fully detailed insight on how the was implementation was handled. Figure 3.2 (at the end of this section) describes the general steps of this algorithm, aiding the creation of a general picture of the flow of the algorithm. In this figure the starred variables mean that they were calculated using a transport equation and the double starred variables represent the corrected values.

3.1.1 General Description of the Algorithm

In this section, a brief description of the general steps of the algorithm will be given:

The first step is the initialization one, where the fields of all variables are created. It is followed by the preparation and solution of the momentum equations, which should be done sequentially, and yield the CV centered velocity values. These velocities are used to calculate the velocity values in the CV faces via the Rhie and Chow interpolation [11], that prevents the pressure checker board issue. Next, the pressure correction equation is assembled, solved, and the resultant pressure correction field is used to correct the face velocities, ensuring that continuity is respected in every CV. This quantity is also used to correct the CV centered values of pressure and velocity. After this step, the residuals are checked for convergence: if it wasn't achieved, the cycle is restarted by taking the latest values of every variable, otherwise, the cycle is stopped.

With the short review given in the previous section, a good understanding on the sequence of steps of the SIMPLE algorithm can be attained. Notwithstanding, this description isn't in depth enough to allow a full grasp on how the implementation was carried out, thereupon, this section gives a comprehensive description of each of the aforementioned steps.

3.1.2 Algorithm Initialization

In the initialization step, the initial fields for all the variables (u, v, p) are prescribed. This can be done by imposing a uniform initial field, or one in which the values vary across the domain. It is relevant to point out that the initial field should make sense physically, and, if viable, be as close to the solution as possible, since this can influence the amount of iterations required until convergence.

3.1.3 Preparing and Solving the Momentum Equations

After the initialization, the first step included in the iterative cycle is the preparation and solution of the momentum equations (equation 2.48), which yield the velocity values in each CV center. The importance of this step cannot be overstated, since it is here that it is made sure that momentum is respected for the pressure field provided.

If the flow is 2D, this step is actually comprised by two steps, since the equations of u and v momentum should be solved sequentially. To help smoothening the numerical solution process, but also because these equations are non-linear, an under relaxation (α_u) is applied to them. It is relevant to say that its value can be the difference between convergence, how quickly it can be achieved, or divergence.

It is also noteworthy that right before solving these equations, the momentum residuals are computed:

$$R_M = \frac{A_P u_P + \sum A_L u_L + (S_u)_P}{A_p max(u)}. \tag{3.1}$$

This information that will relevant in the convergence monitoring step. When debugging the implementation of these equations, a second residual computation could be done after the solution of the equations as a check that they are being solved correctly. In this case, it should be as small as the numerical solver's accuracy allows.

3.1.4 Rhie and Chow Interpolation

The Rhie and Chow interpolation is used to calculate the velocity values in the CV faces (u_f, vf). When calculating these values, the pressure gradient of the nodes adjacent to the face, as well as the facial one are considered. Exemplifying on the east face of a CV (assuming a cartesian and uniform grid):

$$u_e^* = \overline{u^*} - \overline{\left(\frac{V_P}{A_P}\right)}\left((\nabla p^n) - \overline{(\nabla p^n)}\right), \tag{3.2}$$

In this equation the overlined values represent a linear interpolation from nodes E and P:

$$u_e^* = \frac{u_E + u_P}{2} - V_P \frac{\frac{1}{A_E} + \frac{1}{A_P}}{2}\left((\nabla p)_E - \frac{(\nabla p)_E + (\nabla p)_E}{2}\right), \tag{3.3}$$

The face centered gradient of pressure can be calculated in any of the two ways mentioned in section 2.2.3. The variables V_P and A_P respectively represent the volume, and the momentum matrix's diagonal coefficient of the CV in question. The CV centered velocities used in this computation are the ones that result from the solution of the momentum equations.

3.1.5 Preparing and Solving the Pressure Correction Equation

With the face velocities computed in the previous step, it is necessary to calculate the continuity residual by summing the face fluxes $F_f = A_f\left(\mathbf{u}_f \cdot \mathbf{n}_f\right)$:

$$Q_m^* = F_e - F_w + F_n - F_s, \tag{3.4}$$

The sum of the absolute values of this vector will be used in the convergence check, and the vector itself will be used as a source term in the pressure correction equation (whose derivation is detailed in section 3.1.7). The pressure correction equation written for a cartesian grid is:

$$-A_e\overline{\left(\frac{V_P}{A_P}\right)_e}(\nabla p')_e + A_w\overline{\left(\frac{V_P}{A_P}\right)_w}(\nabla p')_w - A_n\overline{\left(\frac{V_P}{A_P}\right)_n}(\nabla p')_n + A_s\overline{\left(\frac{V_P}{A_P}\right)_s}(\nabla p')_s = -Q_m^*, \tag{3.5}$$

whose resultant pressure correction field will be used to correct the velocity and pressure fields:

$$u_f^{**} = u_f^* - \overline{\left(\frac{V_P}{A_P}\right)_f}(\nabla p')_f, \tag{3.6}$$

$$u_P^{**} = u_P^* - \frac{V_P}{A_P}(\nabla p')_P, \tag{3.7}$$

$$p_P^{**} = p_P^* + \alpha_p p'_P, \tag{3.8}$$

here, the pressure under relaxation factor α_p is important to "slow down" the speed at which pressure is corrected, while still assuring that continuity is being respected (it is not applied to the correction of velocities). This factor prevents the pressure field from changing to much between two iterations, something that could effectively thwart the convergence process. The optimal values of α_p change from case to case, but, in incompressible regime are usually kept in the range of $0.2 - 0.4$.

3.1.6 Convergence Criterion

In this step, convergence is assessed. For this, a criterion or several criteria have to be specified. Two common criteria, are:

- Consider the result converged if both the momentum and continuity residuals are below a given tolerance (in this work it was usually taken to be 10^{-6});
- Consider the result converged if the maximum change throughout all the variables between two consecutive iterations ($max(\phi^{n+1} - \phi^{n}) \leq tol$) is below a given tolerance (in this work it was usually taken to be 10^{-6});

If the current result checks the prescribed convergence criterion(a), the iterative process will be stopped. Otherwise, the cycle will continue.

3.1.7 Pressure Correction Equation

In this section, the derivation of the pressure correction equation of the incompressible version of the SIMPLE algorithm will be presented (for a cartesian grid).

The fact that the continuity residual (which can be calculated by using 3.4) is not null, means that the velocity field does not guarantee continuity, so a correction must be applied:

$$Q_m^* + Q_m' = 0, \tag{3.9}$$

Where Q_m' is the continuity correction, which can be written as a function of the face velocity correction:

$$Q_m' = A_e u_e' - A_w u_w' + A_n v_n' - A_s v_s', \tag{3.10}$$

The velocity correction will now be written as a function of the pressure correction gradient. To do this, a simplified momentum equation is considered, where the only term left is the pressure gradient:

$$u_f' = -\overline{\left(\frac{V_P}{A_P}\right)_f} (\nabla p')_f \,. \tag{3.11}$$

Substituting in equation 3.10 and 3.9, the pressure correction equation is obtained:

$$- A_e \overline{\left(\frac{V_P}{A_P}\right)_e} (\nabla p')_e + A_w \overline{\left(\frac{V_P}{A_P}\right)_w} (\nabla p')_w - A_n \overline{\left(\frac{V_P}{A_P}\right)_n} (\nabla p')_n + A_s \overline{\left(\frac{V_P}{A_P}\right)_s} (\nabla p')_s = -Q_m^*, \tag{3.12}$$

once again, the overlined values represent linear interpolation from the CV centres. The gradients in this equation can be discretized using the central differencing scheme. For the east face:

$$A_e \overline{\left(\frac{V_P}{A_P}\right)_e} (\nabla p')_e = A_e V_P \frac{\frac{1}{A_E} + \frac{1}{A_P}}{2} \left(\frac{p_E' - p_P'}{dx}\right). \tag{3.13}$$

The resultant equation is a Poisson equation, whose result can be used to correct the face mass fluxes and the CV centered pressure and velocity values.

Region					
Inlet		Outlet		Wall	
Pressure Inlet	Velocity Inlet	Pressure Outlet	Outflow	Symmetry	Viscous Wall

Table 3.1: Boundary conditions for incompressible flow.

3.1.8 Incompressible Flow Extensions of the SIMPLE Algorithm

Unsteady Flow

To simulate an unsteady flow, some minor additions have to be made to the algorithm. Namely, if the temporal scheme is implicit, an extra cycle where time is incremented step by step is created to encompass the one just described. This way, the idea is that in the outer cycle, the actions are:

- Incrementing time by one time step $t = t + \Delta t$ until the intended total time is achieved;
- Solving the original SIMPLE cycle with the added unsteady term in the momentum equations;

This means that several iterations of the SIMPLE algorithm must be computed for each time step, so that the time advance is physically grasped by the solver.

If the temporal scheme is explicit, the only action is to solve the equations explicitly, i.e. the only terms in the matrices are the time-related ones. This means that, in this case, only one iteration of the SIMPLE cycle must be completed for each time step.

Boussinesq Approximation

The Boussinesq approximation can be integrated in the SIMPLE algorithm by adding the appropriate source terms to the momentum equations (this was described in section 2.2.5). In addition, the temperature transport equation 2.61 must be solved in every iteration, by introducing an extra step in the algorithm before the convergence check and after the pressure correction steps. This approximation can be seen as a first step to extend of the SIMPLE algorithm to solve compressible flows.

3.1.9 Technical Aspects

In this section, some critical technical aspects will be focused on, including the boundary condition implementation and the solution of linear systems of equations.

Boundary Conditions

The physical boundary conditions for incompressible flow are summarized in table 3.1, and their implementation will be analysed in this section. It is important to point out that there are two equations to be solved in the form of systems of linear equations (momentum and pressure correction equations), so, in order to apply a physical boundary condition, two mathematical boundary conditions must be implemented (one for each equation).

Inlet Boundary Conditions

- **Velocity Inlet** The velocity inlet boundary condition states that the inlet velocity is prescribed. To apply this boundary condition, the following mathematical boundary conditions were implemented:

  Momentum Equation - Fixed velocity at the inlet, via the Neumann boundary condition.

  Pressure Correction Equation - $(\nabla p')_{inlet} = 0$, which corresponds to a null Neumann boundary condition. This way, there is no velocity correction at the inlet.

  This means that p'_{inlet} is used to recalculate the inlet pressure at each iteration. In incompressible flow, the velocity inlet is equivalent to setting the mass flow rate at the inlet.

- **Pressure Inlet** The pressure at the inlet is prescribed, by implementing:

  Momentum Equation - Fixed velocity at the inlet, via the Neumann boundary condition.

  Pressure Correction Equation - $p'_{inlet} = 0$, which is a null Dirichlet boundary condition.

  When this boundary condition is used, the velocity at the inlet is recomputed in the correction step of each iteration, by taking $(\nabla p')_{inlet}$ into account.

Outlet Boundary Conditions

- **Pressure Outlet** The pressure value is prescribed at the outlet. This boundary condition is applied in the same manner as the pressure inlet.

- **Outflow** The pressure and velocity gradients are set as null at the outlet. To use this boundary condition, the flow must be assumed to be fully developed, and is implemented by:

  Momentum Equation - Null velocity gradient at the outlet, via a null Neumann condition at the outlet.

  Pressure Correction Equation - $(\nabla p')_{outlet} = 0$, which is a null Neumann boundary condition.

  This boundary condition is usually applied with an additional correction to the outlet mass flow rate: $\dot{m}_{out} = \dot{m}_{out}\frac{\dot{m}_{in}}{\dot{m}_{out}}$, from which the outlet velocity is corrected, to guarantee that continuity is respected in the domain. The pressure at the outlet is extrapolated from the last CV center $(p_{outlet} = p_P)$.

Wall Boundary Conditions Two physical boundary conditions can be considered at a wall:

Viscous Wall If the wall is viscous (figure 3.1(a)), the fluid adheres to it. Hence, the component of velocity which is parallel (u) to the wall is set to be the same as said wall's velocity, while the perpendicular (v) component is set to be null (impermeability condition). Therefore, the boundary conditions of both velocity components are:

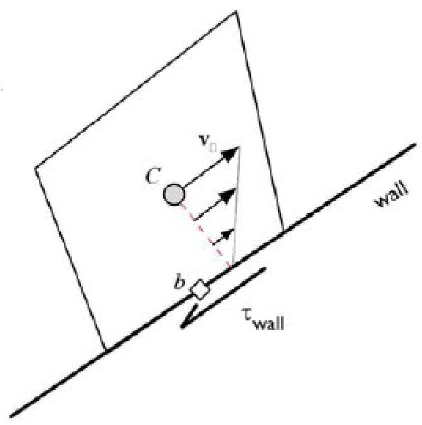
Momentum Equation - Dirichlet in the momentum equation, with $u_f = u_{wall}$ and $v = 0$.

Pressure Correction Equation - $(\nabla p')_{wall} = 0$ perpendicularly to the wall, so that the v is always null at the wall.

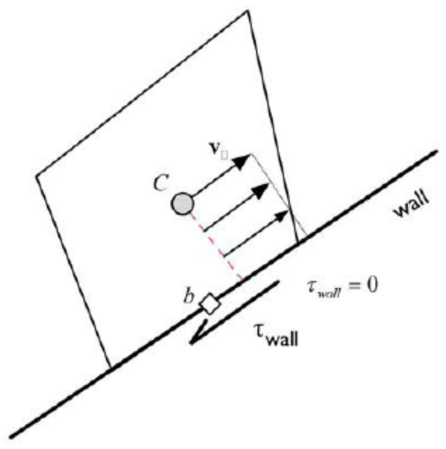
Inviscid Wall If the wall is inviscid, the fluid is assumed to be perfect (figure 3.1(b)). This is simulated by:

Momentum Equation - Specifying the gradient perpendicular to the wall of the velocity parallel to it to be null (null Neumann) $(\nabla u)_{wall} = 0 \Rightarrow u_f = u_P$, and the component of velocity normal to the wall to be $v = 0$ (null Dirichlet).

Pressure Correction Equation - $(\nabla p')_{wall} = 0$ perpendicularly to the wall, so that the wall's impermeability is kept.



(a) Viscous Wall.



(b) Inviscid Wall.

Figure 3.1: Wall boundary conditions (both taken from [30]).

Additional Notes on Boundary Condition Implementation The momentum inlet and outlet boundary conditions are always the same, despite the physical boundary conditions. The reason for this is that prescribing a velocity value at the outlet results in ruining the diagonal dominance of the matrix, so the gradient's value must be specified. On the other hand, this also leads to the need to specify the velocity values somewhere, which must be at the inlet or at the walls.

Regarding the pressure correction equation, as a rule of thumb, wherever a velocity value is specified, the pressure correction equation should have a null Neumann boundary. The velocity correction is $u'_f = \overline{\left(\frac{V_P}{A_P}\right)_f} (\nabla p')_f$, so, by specifying a null gradient, there will be no velocity correction. On the other hand, whenever the pressure is specified (pressure inlet and outlet), the pressure correction is zero, as way to keep the pressure value unaltered in that boundary. In some cases, the value of pressure (and ultimately the value of the pressure correction) is not imposed in any boundary, possibly leading to very large or very small pressure correction values. This is not problematic, since, for incompressible flow, continuity is guaranteed by $\nabla p'$, not p'. Thus, when correcting the pressure values, the solution can be subtracted by a constant (e.g. the average pressure correction value) without loss of physical meaning.

Solution of Linear Systems

In section 2.2.5, the equations were written in the generic form in which they were implemented. This section will further clarify the reason for this formulation.

Going back to equation 2.62, it is shown that it is possible to write all the equations in a generic form for a CV:

$$A_P\phi_P + \sum_{f=1}^{F} A_L\phi_L = S_\phi, \tag{3.14}$$

where A_P represents the coefficients multiplied by ϕ_P, and A_L represents the coefficients multiplied by ϕ_E, ϕ_W, ϕ_N, and ϕ_S, all of which are implicit. It is important to remind that this equation is employed for every control volume throughout the whole domain, thus generating a system of linear equations. The A_L coefficients serve the purpose of linking the several equations, with contributions from the adjacent control volumes. Representing the system matricially:

$$\begin{bmatrix} A_P & A_N & \cdots & A_E & \cdots & \cdots & \cdots & \cdots \\ A_S & A_P & A_N & \cdots & A_E & \cdots & \cdots & \cdots \\ \cdots & A_S & A_P & A_N & \cdots & A_E & \cdots & \cdots \\ A_W & \cdots & A_S & A_P & A_N & \cdots & A_E & \cdots \\ \cdots & \cdots & A_W & \cdots & A_S & A_P & A_N & \cdots \\ \cdots & \cdots & \cdots & A_W & \cdots & A_S & A_P & A_N \\ \cdots & \cdots & \cdots & \cdots & A_W & \cdots & A_S & A_P \end{bmatrix} \begin{bmatrix} \\ \\ \\ \phi_P \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \\ \\ S_\phi \\ \\ \\ \\ \end{bmatrix} \tag{3.15}$$

These systems must be solved numerically due to their size, but, to guarantee convergence of these methods, diagonal dominance of the matrices must be ensured:

$$||A_P|| \geq \sum_L ||A_L|| \tag{3.16}$$

It is here that the under relaxation factor comes into play, for, by dividing the diagonal entries by α, the absolute value of A_P in every line is increased, ameliorating stability.

Implicit temporal discretization also enhances the stability of the numerical solution of linear systems. The principle is the same as for the under-relaxation factors, the diagonal dominance is improved by incrementing the values of A_P with the unsteady term contribution.

In this work, the numerical solution of the linear systems of equations was achieved by using either the CGS, or the BICGSTAB algorithms, while using the incomplete LU factorization as a pre-conditioner. All of these methods are all available as built-in functions in Matlab, so their implementation was not necessary.

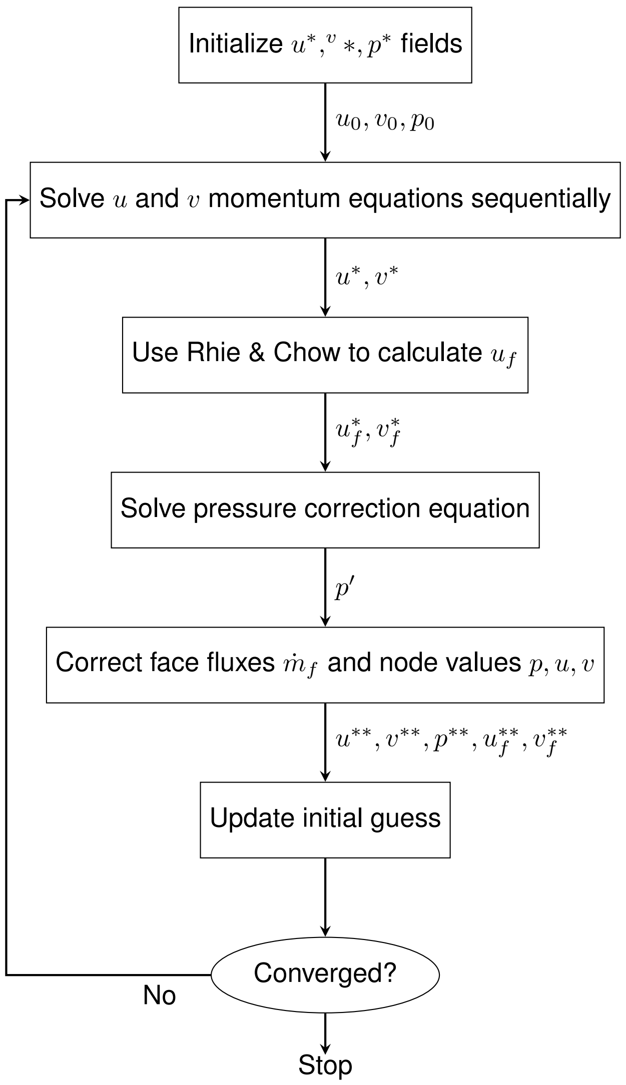

Initialize $u^*,^v *, p^*$ fields
u_0, v_0, p_0
Solve u and v momentum equations sequentially
u^*, v^*
Use Rhie & Chow to calculate u_f
u_f^*, v_f^*
Solve pressure correction equation
p'
Correct face fluxes $\dot{m}_f$ and node values p, u, v
$u^{**}, v^{**}, p^{**}, u_f^{**}, v_f^{**}$
Update initial guess
Converged?
No
Stop

Figure 3.2: Typical implementation of the SIMPLE algorithm to solve incompressible flows.

3.2 All-Mach Solver Algorithm

The algorithm of the all-Mach solver that was implemented is based on those proposed by [13, 16, 20]. Its main steps are depicted in figure 3.3 (at the end of this section), where the bolded ones aren't part of the original SIMPLE algorithm. In this figure, the starred and double starred values hold the same significance as in the previous section.

This algorithm will be presented in the same way that the SIMPLE algorithm was, i.e. starting by a brief description of its main steps, followed by a more thorough exposition on each step, along which the main differences between the SIMPLE algorithm will be emphasized. Afterwards, additional technical notes are included.

3.2.1 General Description of the Algorithm

The algorithm is started by initializing all the required fields (u, v, ρ, p, T). These values are then used to prepare and solve the momentum equations, which are solved sequentially for each direction. The following step is the density field recalculation, where the density values are computed by using the ideal gas equation. The CV centered velocity and pressure fields are then used to interpolate the face velocity values with the Rhie and Chow interpolation. The next step is to prepare and solve the pressure correction equation, which yields the corrections both to density and velocity fields, that are necessary to guarantee continuity. The pressure values are also corrected in this step. Afterwards, the energy equation must be prepared and solved (by using the corrected fields), yielding the temperature distribution. Convergence is then assessed: if it wasn't achieved, the cycle starts over, taking the last available fields of all variables. Otherwise, the program is stopped.

The algorithm's steps will now be looked in to, one by one.

3.2.2 Algorithm Initialization

In the initialization step the fields of every variable must be specified as an initial solution, but, in this case, density and temperature must also be specified. Some extra care must be taken in this step, since the density field impacts the mass flow rate, and, consequently, the continuity residual in each control volume, which, if too large, can hinder, or even foil the solution procedure in the very first iterations. The point being that some extra effort should be made to make sure the initial fields make sense physically.

3.2.3 Preparing and Solving the Momentum Equations

After the initial fields are prescribed, the momentum equations for compressible flow (2.42 must be solved. These equations are somewhat different from the incompressible flow ones, because density must be accounted for, as well as the additional viscous source terms. The strategy adopted to discretize these viscous source terms in this work is available in section 2.2.5.

An interesting behaviour was observed, regarding the under-relaxation factor α_u, as its optimal value can vary a lot between simulations, much more so, than in the incompressible flow SIMPLE.

3.2.4 Density Recalculation

This step is exclusive to the all-Mach version of the algorithm. Here, the density field is recalculated by using the ideal gas equation $p = \rho RT$. Only the CV centered values of density are being recalculated, though, since the pressure and temperature fields aren't stored at the faces. The value of ρ_f must then be approximated by one of the convective schemes described in section 2.2.3. In this work this was preferentially achieved using the UDS, due to its added stability.

3.2.5 Rhie and Chow Interpolation

Similarly to the original SIMPLE, the face velocity values must be recalculated by the Rhie and Chow interpolation (equation 3.3, by taking the resultant velocity fields from the momentum equations:

$$u_e^* = \overline{u_e^*} - \overline{\left(\frac{V_P}{A_P}\right)_e} \left((\nabla p^n)_e - \overline{(\nabla p^n)_e} \right), \tag{3.17}$$

this step doesn't differ from the original SIMPLE, as the density and temperature fields don't influence the interpolation procedure.

3.2.6 Preparing and Solving the Pressure Correction Equation

After the face density and velocity values have been recalculated, the face mass flow rates m_f have been changed. This implies that continuity is no longer guaranteed throughout the domain, so the pressure correction equation must be solved:

$$\sum_{F=1}^{f} u_f^* \rho_f' A_f + \rho_f^* u_f' A_f = -Q_m^*, \tag{3.18}$$

it is important to note that this equation is no longer a Poisson equation, and its solution might be more complicated than for incompressible flow(this is addressed in section 3.2.10). Nonetheless, the result is used to correct the density, velocity and pressure fields:

$$u_f^{**} = u_f^* - \overline{\left(\frac{V_P}{A_P}\right)_f} (\nabla p')_f\,, \tag{3.19}$$

$$\rho_f^{**} = \rho_f^* + \overline{\left(\frac{1}{RT}\right)_f} p_f', \tag{3.20}$$

$$p_P^{**} = p_P^* + \alpha_p p_P', \tag{3.21}$$

the correction step is slightly more complicated than for the original SIMPLE, since the face values of pressure correction are also required.

3.2.7 Preparing and Solving the Energy Equation

This step is exclusive to the all-Mach and to the extended SIMPLE algorithms. By taking the corrected fields, equations 2.54 or 2.59 must be solved. In compressible flow, this step is mandatory, since without it, the simulation would not be conservative in terms of energy.

Before solving any of these equations, the energy residual must be calculated through an equation is similar to the one used to compute the momentum residual:

$$R_M = \frac{A_P T_P + \sum A_L T_L + (S_T)_P}{A_P max(T)}. \tag{3.22}$$

3.2.8 Convergence Criterion

The convergence assessment step is essentially similar to the original SIMPLE algorithm. A convergence criterion is chosen, and, if said criterion is verified, the program is stopped, otherwise, another iteration is started. In the all-Mach solver, since viscosity and the thermal conductivity can change with temperature, their recomputation is done in this step.

This concludes the description of the all-Mach solver implementation.

3.2.9 Pressure Correction Equation

In this section, the pressure correction equation for the all-Mach version of the SIMPLE algorithm will be derived. The first step is to calculate the mass flow rate residual:

$$Q_m^* = \dot{m}_e - \dot{m}_w + \dot{m}_n - \dot{m}_s. \tag{3.23}$$

This time around, $\dot{m}_f = \rho_f u_f A_f$, hence, the correction of the mass fluxes requires that both density and velocity are corrected. By writing the corrected (**) velocity and density values as a sum of their present (*) values and their corrections ('):

$$u_f^{**} = u_f^* + u'_f, \tag{3.24}$$

$$\rho_f^{**} = \rho_f^* + \rho'_f, \tag{3.25}$$

the pressure correction equation for compressible flows then becomes:

$$Q_m^* + Q'_m = \sum_{F=1}^{f} \dot{m}_f^* + \dot{m}'_f = \sum_{F=1}^{f} u_f^* \rho_f^* A_f + u_f^* \rho'_f A_f + \rho_f^* u'_f A_f + \rho'_f u'_f A_f = 0. \tag{3.26}$$

Here, the last term is usually neglected, both to permit the linearization of the equation, and because its value is much smaller than the remaining ones, so:

$$\sum_{F=1}^{f} u_f^* \rho'_f A_f + \rho_f^* u'_f A_f = -Q_m^*. \tag{3.27}$$

To write the velocity correction as a function of the pressure correction, the simplified momentum equations must be considered for the velocity (as in the original SIMPLE algorithm). On the other hand, for the density correction, this function is the ideal gas equation:

$$u'_f = -\overline{\left(\frac{V_P}{A_P}\right)_f} \left(\nabla p'\right)_f , \tag{3.28}$$

$$\rho'_f = \overline{\left(\frac{1}{RT}\right)_f} p'_f, \tag{3.29}$$

where the face values of temperature, volume, and A_P(momentum matrix diagonal coefficients) must be interpolated from the CV centers. The pressure correction equation for compressible flow now becomes:

$$\sum_{F=1}^{f} u^*_f \overline{\left(\frac{1}{RT}\right)_f} p'_f A_f - \rho^*_f \overline{\left(\frac{V_P}{A_P}\right)_f} \left(\nabla p'\right)_f A_f = -Q^*_m. \tag{3.30}$$

By exemplifying on the east face of a cartesian and uniform grid:

$$\dot{m}'_f = A_f \rho^*_f \overline{\left(\frac{V_P}{A_P}\right)_f} \left(\nabla p'\right)_f + A_f u^*_f \overline{\left(\frac{1}{RT}\right)_f} p'_f. \tag{3.31}$$

The pressure correction values on the CV faces must be calculated from the centered ones, which, in this work, was usually accomplished via the upwind scheme. An alternative being the use of the deferred correction approach, if a better resolution is desired.

If unsteady flow is considered, an extra implicit term must be included, as well as an extra source term. By taking the continuity equation, discretizing in time, and inputting the density correction:

$$V\frac{(\rho)_p^{n+1} - (\rho)_p^n}{\Delta t} = V\frac{\left(\frac{p'}{RT}\right)_p^{n+1} - (\rho)_p^n}{\Delta t}, \tag{3.32}$$

where the explicit term becomes a source term.

3.2.10 Technical Aspects

Stability of the Pressure Correction Equation

The pressure correction equation was found to be the bottleneck of this algorithm, in terms of stability, and, at times performance. The instabilities linked to the process of solving this equation forced the following procedure to be adopted:

- Evaluate the matrix regarding its main diagonal dominance. If it is diagonally dominant, solve the equation as usual, otherwise:

  1 Find the CV where $||A_P|| - \sum ||A_L||$ is largest;

  2 Find the value of α which guarantees $\frac{A_P}{\alpha} = ||A_P|| + \sum ||A_L||$, i.e. the under relaxation required to turn the matrix in a diagonally dominant matrix;

3 Divide the matrix diagonal by α and add include the source term $\frac{1-\alpha}{\alpha} A_P p'_P$. Here the value of$A_P$ before the under relaxation was considered, otherwise, the source term becomes $(1-\alpha)\, A_P p'_P$;

4 Solve the system of equations numerically;

5 Update the value of p'_P, and repeat **4** and **5** until $(p'_P)^{n+1} - (p'_P)^{n} \leq tol$;

This tolerance was usually set to be 10^{-7}, as that was the tolerance when checking the matrices for the dominance of the main diagonal .

By following this procedure, the pressure correction is stabilized, and guaranteed to be solved, safe for some extreme cases where the residuals become too large. The major drawback of this implementation is the decrease in performance, as, in some conditions the time per iteration increased significantly, due to the large number of iterations needed to solve this equation. However, this was a shortcoming that was mostly confined to the quasi 1D test case.

As an alternative, the implementation of the successive over relaxation (SOR) method was tried, but the resulting toll on performance was much more significant.

Boundary Conditions

When considering the compressible flow equations, there are four variables: pressure, velocity, density and temperature. This must be taken into account when choosing the boundary conditions. [50] presents an analysis based on the characteristics, arriving to the conclusion that, when the flow is locally compressible, three boundary conditions are required to be imposed at the inlet, and none at the outlet. On the other hand, when the flow is locally incompressible, two boundary conditions need to be imposed at the inlet, and one at the outlet. This analysis is compatible with the boundary conditions presented by [45] and [30]. An overview of the boundary condition implementation will be presented in this section.

Inlet

- **Incompressible Flow** - The inlet boundary conditions whose implementation is described in section 3.1.9 can be used when the flow is locally incompressible, with the added condition of fixating the temperature at the inlet. This is done via a Dirichlet condition in the energy equation. Also, the value of p'_{inlet} must be used to correct the density value at the inlet, when it is not null.

- **Mass Flow Rate Inlet** - In compressible flow, fixating the velocity isn't the same as fixating the mass flow rate, due to the alterations in the value of density. To implement this physical boundary condition, a specific combination of mathematical boundary conditions is required:

  Momentum Equation - A Dirichlet boundary condition must be applied, imposing the inlet value of velocity.

  Pressure Correction Equation - The velocity and density values must be corrected during each iteration, but the overall mass flow rate must be kept constant. This is achieved by imposing a Robin boundary condition at the inlet $(k = 0)$, allowing the values of density and velocity to fluctuate, while keeping the mass flow rate constant.

Energy Equation - The temperature at the inlet must be specified, by using a Dirichlet boundary condition.

- **Supersonic Inlet** - When the flow is supersonic at the inlet, all the variables' values must be fixated at the inlet:

  Momentum Equation - The inlet velocity is prescribed by using a Dirichlet boundary condition.

  Pressure Correction Equation - $p'_{inlet} = (\nabla p')_{inlet} = 0$, imposing the velocity, pressure, and density's values to be constant between iterations.

  Energy Equation - The inlet temperature is specified at the inlet (Dirichlet boundary condition).

Outlet

- **Pressure Outlet** - The implementation of the pressure outlet boundary condition follows that of the original SIMPLE. An additional mathematical boundary condition must be used to solve the energy equation, which is a null Neumann condition, thus setting $(\nabla T)_{outlet} = 0$. The outlet density is kept during the correction step, since $p'_{outlet} = 0$, but recomputed during the density recalculation step.
- **Supersonic Outflow** - If the flow is supersonic at the outlet, the gradient of every variable is specified as being null. This is done by implementing a null Neumann condition in all three equations, and computing the outlet value of every variable as $\phi_{outlet} = \phi_P$.

Walls The implementation of the viscous and inviscid wall boundary conditions follows the same procedure as for incompressible flow. Yet, the additional boundary condition is applied to the energy equation, which can be:

- **Adiabatic Wall** - Null Neumann boundary condition, so that there is no heat transfer between the fluid and the wall $(\nabla T)_w = 0 \Rightarrow T_w = T_P$.
- **Prescribed Temperature Wall** - Dirichlet condition, where the wall temperature is specified $T_f = T_w$.

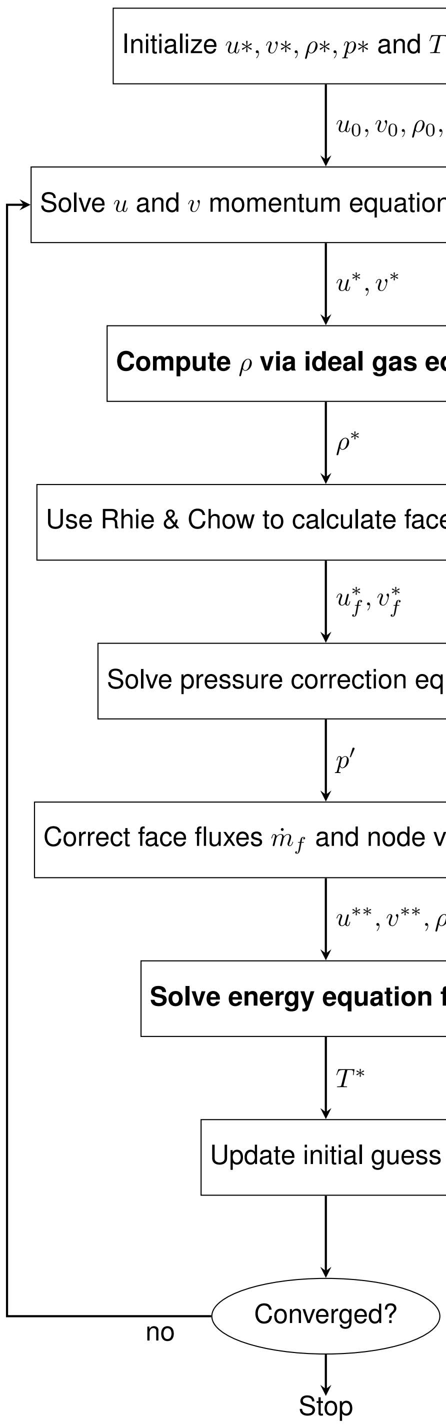
3.2.11 Comparison between the two Algorithms

The main differences between the original SIMPLE and the all-Mach algorithm are:

- Altered, more general, governing equations with additional terms that are neglected for incompressible flows
- Recalculating the density values at the cells, and faces
- A more complicated pressure correction equation with an additional convective term that depends on the local Mach number.

- Solving an energy equation of the flow.
- Possibility of having extra variables (variable μ)

On top of this, the algorithm is now solving for at least 4 variables: u, v, p, T (and additionally ρ, which is explicitly calculated from p and T), instead of the 3 present in the original SIMPLE: u, v, p.



Initialize $u*, v*, \rho*, p*$ and T^* fields
$u_0, v_0, \rho_0, p_0, T_0$
Solve u and v momentum equations sequentially
u^*, v^*
Compute ρ via ideal gas equation
ρ^*
Use Rhie & Chow to calculate face velocities $\dot{u}_f$
u_f^*, v_f^*
Solve pressure correction equation p'
p'
Correct face fluxes $\dot{m}_f$ and node values p, u, v, ρ
$u^{**}, v^{**}, \rho^{**}, p^{**}$
Solve energy equation for T
T^*
Update initial guess
Converged?
no
Stop

Figure 3.3: Flowchart of the All-Mach algorithm that was implemented.

Chapter 4

Results

4.1 Converging Diverging Nozzle

The first test case solved with the all-Mach algorithm was the flow in an inviscid isentropic quasi-1D converging diverging nozzle. This test case features a well-known analytical solution to compare the numerical results to, hence it is often used to validate all-Mach solvers. Furthermore, in different regions of the nozzle, the flow can be subsonic, supersonic and transonic, allowing a demonstration of an all-Mach solver's full capabilities. The nozzle shape will be defined in the next section, followed by a detailed demonstration of how the boundary conditions were implemented and the results obtained.

4.1.1 Problem Definition

The nozzle shape was presented by [29]:

$$S(x) = 0.1\left[1 + 4\left(\frac{x}{l}\right)^2\right] \tag{4.1}$$

Here $-0.5 \leq x \leq 0.5$ and $l = 1$. It is important to notice that, even though the nozzle shape is symmetric, in a real case, it would be attached to a reservoir with a large volume, where stagnation conditions could be assumed, and the type flow would depend on the pressure difference between the reservoir and the outlet. Because of the symmetry of the nozzle, the fully-subsonic (or fully-supersonic) flow inside it also becomes symmetric, with the symmetry line being the throat.

If the inlet and outlet pressures are different, one of two different phenomena might happen:

1. If the difference is large enough, the flow will be subsonic at the inlet and until it reaches the throat (where it becomes sonic). After the throat, the flow becomes supersonic and stays that way until it reaches the outlet. If the inlet flow is supersonic, the opposite can also happen, ie.: it will go from supersonic at the inlet to subsonic after the throat.

2. If the pressure difference is anywhere in between zero and the value required for the outlet to be

supersonic, a shockwave will appear. The fluid's behaviour will be the same until it reaches the throat, but, before it reaches the outlet, a discontinuity will happen, turning the supersonic flow into a subsonic one, which will stay subsonic all the way to the outlet.

If a shockwave is generated, its location can be controlled by altering the pressure difference between the inlet and the outlet: a larger pressure difference will push the shockwave towards the nozzle outlet, while a smaller one will pull it closer to throat. The behaviour of the flow inside a convergent-divergent nozzle is explained in detail in [34].

4.1.2 Quasi-1D Equations

Since this problem is not purely one-dimensional, some care must be taken when choosing the control volume over which the transport equations will be integrated, so that the area variation is taken into account (an alternative would be to use a purely 1D CV, and include the area variations in the equations themselves, as in [50]). The control volume that was considered is presented in figure 4.1.

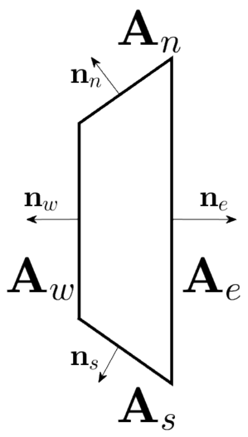


Figure 4.1: Quasi-1D control volume.

The north and south wall A_s and A_n are adiabatic and impermeable, so no energy or mass transfer occurs through them. However, the pressure acting on these walls must be taken into account in the momentum equation, by using Gauss's theorem. By taking equation 2.7, neglecting the shear and gravity terms, and discretizing it in a quasi-1D CV:

$$\sum_{f=1}^{F} \rho_f u_f \left(\mathbf{u}_f \cdot \mathbf{n}_f\right) A_f = -\sum_{f=1}^{F} A_f p_f, \tag{4.2}$$

expanding the sums, and taking $n_n = \left(n_{n_x}, n_{n_y}\right)$ and $n_s = \left(n_{s_x}, n_{s_y}\right)$, leads to:

$$\rho_e u_e u_e A_e - \rho_w u_w u_i A_w = -\left(A_e p_e - A_w p_w - A_{n_x} p_n + A_{n_y} p_n - A_{s_x} p_s - A_{s_y} p_s\right),$$

The pressure on the walls generates a force normal to them, whose vertical components cancel $A_{n_y} p_n = A_{s_y} p_s$, if it is considered that $p_s = p_n = p_P$. The horizontal components (whose direction coincides with that of n_w), become:

$$-\left(A_e p_e - A_w p_w - A_{n_x} p_n - A_{s_x} p_s\right) = -\left(A_e p_e - A_w p_w - 0.5\left(A_e - A_w\right) p_n - 0.5\left(A_e - A_w\right) p_s\right) =$$
$$-\left(A_e p_e - A_w p_w - \left(A_e - A_w\right) p_P\right),$$

leading to the final momentum equation:

$$\rho_e u_e u_e A_e - \rho_w u_w u_w A_w = -A_e p_e + A_w p_w + \left(A_e - A_w\right) p_P. \tag{4.3}$$

An alternative method would be to integrate the pressure gradient in volume and use the pressure in the CV centers immediately to the right and to the left (cell centered gradient). The method that was implemented was the one where the gradient is taken into account by using Gauss's theorem, and the two methods were compared for the fully subsonic problem.

Regarding the continuity equation, it can be derived by discretizing equation 2.6:

$$\rho_e u_e A_e - \rho_w u_w A_w = 0. \tag{4.4}$$

This equation can be transformed into the pressure correction equation:

$$\sum_{f=1}^{2} \dot{m}_f{}' = \dot{m}_w{}' + \dot{m}_e{}' = u_w^* A_w \overline{\left(\frac{1}{RT}\right)_w} p'_W - \rho_w^* A_w \overline{\left(\frac{V_p}{A_p}\right)_w} \left(\frac{p'_P - p'_W}{\Delta x}\right) +$$
$$+ u_e^* A_e \overline{\left(\frac{1}{RT}\right)_e} p'_e + \rho_e^* A_e \overline{\left(\frac{V_p}{A_p}\right)_e} \left(\frac{p'_E - p'_P}{\Delta x}\right) = -Q_{mp}^*, \tag{4.5}$$

where Q_{mp}^* is the continuity residual of the CV. The convective fluxes were discretized using first order upwind and the diffusive terms via central differences. The overlined values are linearly interpolated from the cell centered values,except for the boundary cells, where $\overline{\left(\frac{1}{RT}\right)_f}$ is directly calculated with the boundary temperature and $\overline{\left(\frac{V_p}{A_p}\right)_f}$ is calculated using the nearest cell's values for A_p and V_p. Also, in the boundary faces, the width of the cells is halved.
As for the energy equation, it can be obtained by discretizing equation 2.9:

$$\rho_e u_e A_e c_p T_e - \rho_w u_w A_w c_p T_w = -V\left(\nabla p\right)_P u_P. \tag{4.6}$$

In this equation, the cell centered gradient of pressure is calculated via the Gauss gradient.

4.1.3 Boundary Conditions in 1D

In this section, the boundary conditions used in this quasi-1D test case are stated. These boundary conditions are based on those presented by [29] and [34].

Inlet

1. **Velocity Inlet** - Used for very low Mach numbers, where the flow is virtually incompressible. As the inlet Mach number increases the convergence becomes increasingly difficult, and, eventually unachievable.

2. **Pressure Inlet** - Used to simulate shock-wave flows, in which a specific pressure difference between the inlet and the outlet is required. It was also used in the transonic flow example.

3. **Mass Flow Rate Inlet** - Used for the full range of fully subsonic flows. In this test case it was implemented in a way that takes advantage of the symmetry of the problem, providing a way to fixate the inlet velocity and Mach number (in conjunction with the Pressure Outlet).

4. **Supersonic Inlet** - Used for flows which are supersonic at the inlet.

All of the above boundary conditions are detailed in section 3.2.10, with the exception of the 'hybrid' mass flow rate inlet, which was implemented as explained by [29]: since the flow is symmetric, at convergence the inlet and outlet values for every variable will be the same. This means that with the prescribed temperature (inlet) and pressure (outlet) values, it is possible to calculate the inlet density at convergence. Since the mass flow rate is $\dot{m}_{inlet} = (\rho u A)_{inlet}$, it is possible to set its value, so that the desired inlet Mach number is achieved.

The pressure correction equation is treated in the same way as for the mass flow rate inlet boundary condition: by setting the sum of the convective and diffusive boundary terms as zero (Robin boundary condition). Yet, the inlet pressure is corrected in the same way as is done for the velocity inlet :it is extrapolated from the first cell center. Density is calculated from the ideal gas equation and the velocity value is calculated so that it guarantees continuity in the first CV.

It is also noteworthy that in all cases the temperature value was specified at the inlet.

4.1.4 Outlet

1. **Pressure Outlet** - Used for all that flows that are subsonic at the outlet.

2. **Supersonic Outflow** - Used for all flows which are supersonic at the outlet. A constant-area domain extension at the nozzle's outlet was considered whenever this boundary condition was used. Since the area isn't constant between the outlet and the face between the last two CVs, setting all the outlet gradients to zero could influence the results (due to the area not being constant), an effect that ceases to be important when the mesh becomes fine enough.

Once again, all of these boundary conditions are explained in detail in section 3.2.10. An important aspect of the implementation is that the temperature gradient was always taken to be null at the outlet.

4.1.5 Results

In this section, the results are compared to those available in the literature and to the analytical solution. It is important to note that all the simulations utilized the upwind scheme for the convective fluxes and the central differencing scheme for the diffusive terms. A table with the under relaxation factors used for every simulation is also provided.

Fully Subsonic Flow

The fully subsonic test was run by setting the steady state inlet Mach=0.25, by using the strategy detailed in section 4.1.3. The outlet pressure was also fixed and the solution ran on a uniform 400 cells mesh.

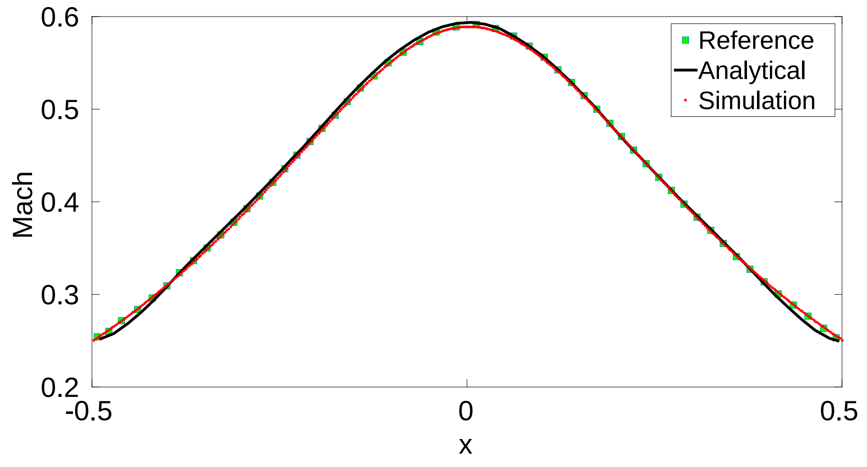


Figure 4.2: Fully subsonic predicted Mach profile.

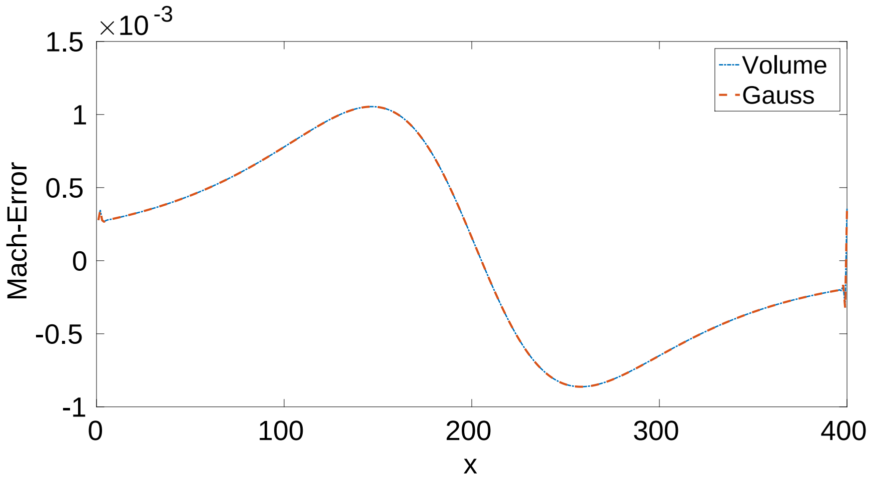


Figure 4.3: Absolute error of the Mach number comparison for the fully subsonic flow.

Figure 4.2 shows a very good agreement with the results by [29], which were obtained using a 100 control volume uniform mesh. Both the reference and the simulated results overlap the analytical solution far from the throat and from the the inlet and outlet. These are the areas where the diffusivity of the upwind scheme is most visible.

The way in which the pressure integral was handled was compared for this problem, and the absolute error of the Mach number was compared in figure 4.3. The profiles overlap throughout the whole domain, showing that there isn't a major difference in the results, at least for this case.

Shockwave Flow

The case that featured a shock wave was simulated by imposing a ratio between the inlet and outlet pressures: $\frac{p_{in}}{p_{out}} = 1.2$. To enforce this pressure difference, the pressure and temperature were specified at the inlet and the pressure was specified at the outlet. A 500 control volume uniform mesh was utilized to solve this problem, in hopes that the shock intensity and its location were well predicted. The results were compared in figure 4.4 to those obtained by [29], who ran the simulation on a 400 control volume uniform mesh, and the analytical solution.

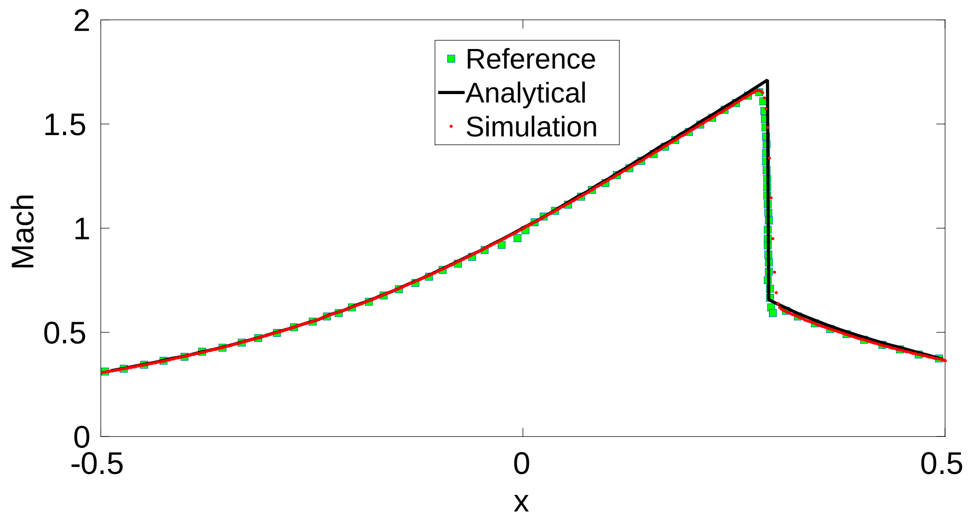


Figure 4.4: Flow with a shock

The figure shows that the shock location and its intensity are relatively well captured. The result does not feature any oscillations near the shock, but its wideness was predicted to be slightly larger than the one predicted by [29], who used a TVD scheme. Both solutions almost overlap the analytical solution throughout most of the domain. Near the shock, the differences between the solutions are bigger, and can be clearly seen in figure 4.5, where this area was zoomed in to: the shock location appears to be very slightly mispredicted, and the Mach peak before the shock is clearly dampened by the upwind scheme.

The entropy constant $\left(\frac{p}{\rho^{\gamma}}\right)$ was calculated relative to its inlet value and is depicted in figure 4.6. The only meaningful change occurs at the shock wave location, where its value increases abruptly, as expected. The overshoot is likely to be caused by the use of different order schemes for pressure (CDS is 2^{nd} order) and density (UDS is 1^{st} order), causing the shock to be wider in the density profile than in

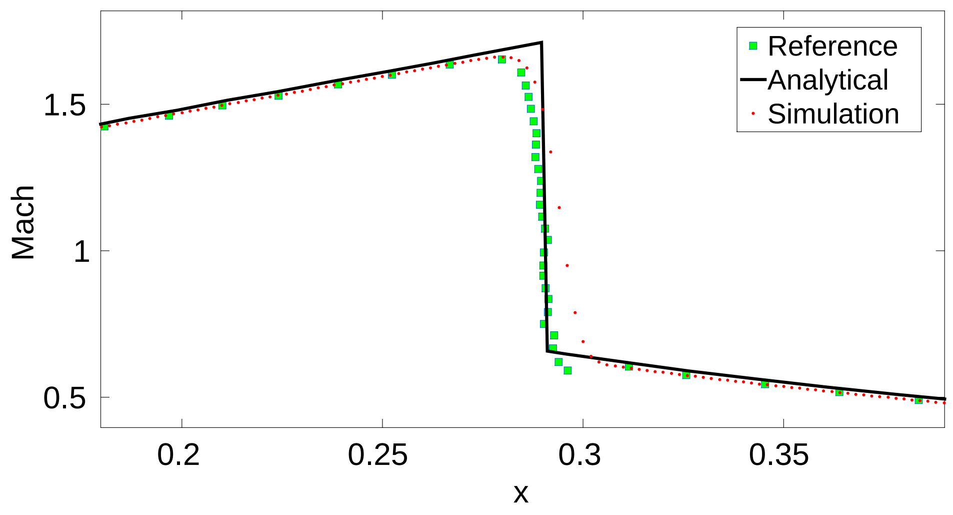


Figure 4.5: Zoom of flow with a shock wave near the shock.

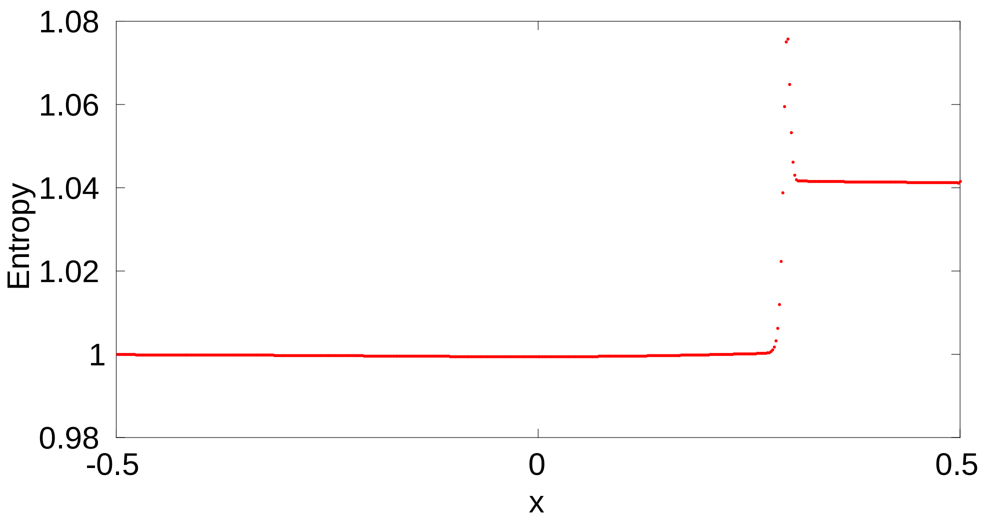


Figure 4.6: Entropy constant $\left(\frac{p}{\rho^{\gamma}}\right)$ relative to its inlet value for a shock wave flow.

the pressure profile.

Subsonic-Supersonic Flow

To simulate the subsonic-supersonic flow across the nozzle, the inlet temperature and pressure were specified and the outlet was left free, using the supersonic outflow boundary condition. 400 control volumes were used and the result compared to the one obtained by [29] and the analytical solution in figure4.7, where the proximity between the solutions is patent, overlapping in most of the domain. Yet, at the outlet, where the velocity is largest, there is a discrepancy between the simulated results, and the analytical and reference ones. This discrepancy might be connected to the upwind scheme's low order of accuracy, which, since the outlet values were not fixated, is felt heavier in this area of the nozzle.

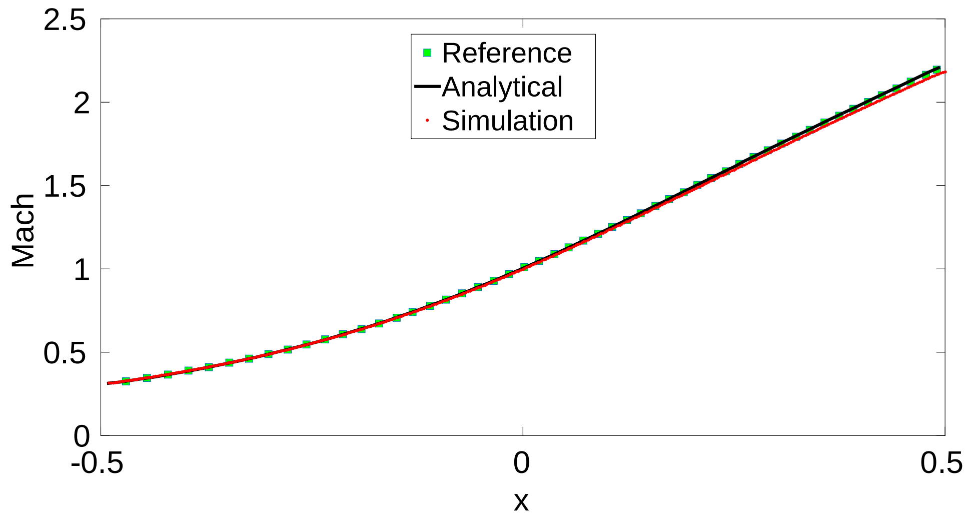


Figure 4.7: Subsonic-Supersonic flow

Fully Supersonic

The fully supersonic case was simulated using the supersonic inlet and the supersonic outflow boundary conditions on a uniform mesh with 400 control volumes. The inlet Mach number is reported to be 2.5, albeit the data plotted by [29] is closer to 2.52 at the inlet. Hence, the latter value was used in this simulation.

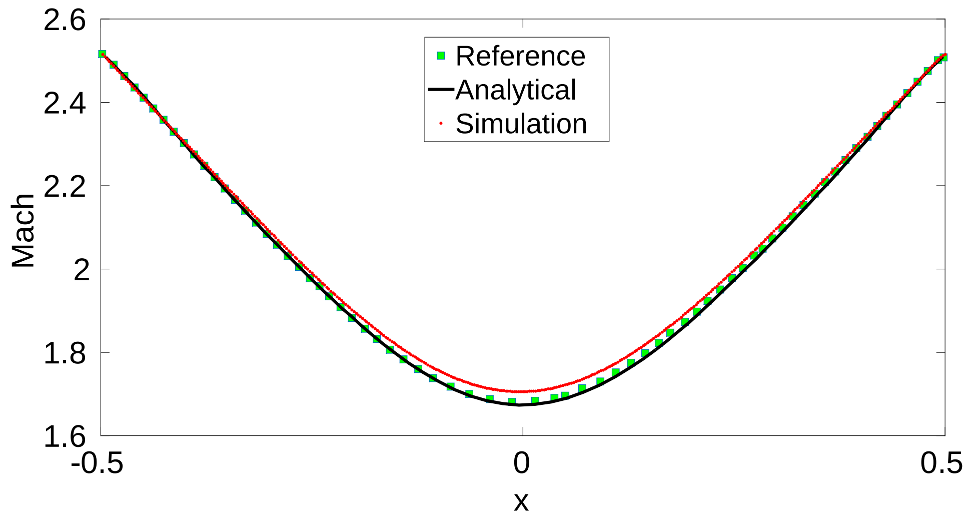


Figure 4.8: Fully supersonic flow

The two results are very close in the vicinity of the nozzle's inlet and outlet, where the two Mach profiles overlap. Near the nozzle's throat, however, the Mach profile is somewhat different from the one obtained by [29] and the analytical one, since it doesn't decrease as much in this area. This might be related to the upwind scheme's large diffusivity.

M_{in}	Case	Boundary Conditions	α_u	α_p	α_T	Grid	Iterations
0.01	Subsonic	Velocity Inlet, Press. Outlet	0.7	0.3	0.9	400	74
0.01	Subsonic	MFR Inlet, Press. Outlet	0.7	0.3	0.9	400	59
0.1	Subsonic	Velocity Inlet, Press. Outlet	0.2	0.02	0.3	100	759
0.1	Subsonic	MFR Inlet, Pressure Outlet	0.5	0.1	0.7	400	380
0.25	Subsonic	MFR Inlet, Pressure Outlet	0.2	0.05	0.2	200	1568
0.25	Subsonic	MFR Inlet, Pressure Outlet	0.3	0.1	0.3	400	4789
0.3	Shock	Press. Inlet and Outlet	0.4	0.05	0.4	200	4127
0.3	Shock	Press. Inlet and Outlet	0.3	0.05	0.3	500	19620
0.3	Subsonic-Supersonic	Press. Inlet, Outflow	0.6	0.3	0.8	400	1119
2.5	Supersonic	Supersonic Inlet, Outflow	0.7	0.3	0.8	400	588
3.5	Supersonic	Supersonic Inlet, Outflow	0.7	0.3	0.8	400	390

Table 4.1: Number of iterations until convergence for several conditions

4.1.6 Notes on Convergence

All the simulations were run until the momentum, energy and continuity residuals were below 10^{-6}, which were calculated as specified in section 3.

A relevant matter for achieving convergence when simulating the different variations of this test case was the value of the under-relaxation factors. These values varied substantially from case to case as can be seen in table 4.1. In some cases, this might be to blame for a part the difference in convergence time between simulations.

For fully compressible or fully incompressible flow, the farther the inlet Mach number is from the non-physical inlet Mach interval (0.3 to ≈2), the less iterations were needed for convergence, which, in the fully incompressible case, goes hand-in-hand with the allowance for increasing the under-relaxation factors without divergence occurring.

For fully subsonic flow, the previously stated defective behaviour of the velocity inlet boundary condition can be seen very clearly: for $M_{in} = 0.1$ the grid had to be less refined and the under-relaxation factors decreased for the simulation to converge, and it still took longer than the mass flow rate inlet to do so.

The shock wave cases were significantly more difficult to solve (as was expected), which might be due to fixating the pressure both at the inlet and the outlet, leaving the mass flow rate completely free to change.

Another important matter is the choice of initial solution. Every simulation was started by inserting the inlet value of every variable for all the domain. An alternative would be to use the analytical solution, which would've accelerated the convergence rate or to use a multigrid method. This way, the oscillations generated during the first iterations (which are the biggest ones and usually accountable for divergence) would either be diffused in a less refined grid or would not happen at all.

4.2 2D Subsonic Flow

The first two dimensional test cases solved with the previously discussed algorithm were the flow inside a lid-driven cavity and the flow in a convection driven cavity. These are two very commmon test cases for Navier-Stokes solvers, due to their domain being a simple square.

4.2.1 Lid-Driven Cavity

In this test case a viscous fluid inside a square cavity is set in motion by the top wall, which has a fixed velocity. Since the fluid is enclosed in the cavity, it starts to rotate, generating a big, more central vortex, and several smaller ones near the cavity's corners. This is a very well documented test case [32, 33, 43, 44], that has been solved for a wide range of Reynolds numbers and is easy to set up. Furthermore, it has been solved successfully by other all-Mach solvers [28, 29].

To set $Re = \frac{UL}{\nu}$ as desired, the top wall's velocity is fixed: $U = 1m/s$, the cavity's side length taken to be $L = 1m$, and the viscosity's value chosen accordingly. Bearing in mind that a viscous fluid adheres to a wall when it comes into contact with it, the fluid's velocity at each wall was set to be the same as said wall's velocity (i.e. null in the side and bottom walls and U in the top wall). The pressure and temperature values were imposed in the top wall ($P_n = 10^5 Pa, T_n = 300K$) and extrapolated in the rest of the boundaries.

Results

The results were compared to those provided by [32] for $Re = 100$,and [32, 33] for $Re = 1000$. This comparison is shown in figures 4.9 to 4.12, where the u-velocity profile along the vertical line and the v-velocity profile along the horizontal line through the cavity's center have been delineated.

It is worth to mention that, as the Reynolds number is incremented , the grid had become more refined in order to capture the rapid escalation of the velocity gradients near the walls. This effect was felt to such an extent, that, for $Re = 1000$, it became clear that using the upwind scheme wasn't feasible, thence, the linear interpolation scheme was implemented.

For $Re = 100$ and a 120 by 120 CV grid, the velocity profiles seem to be well predicted using both UDS and LIN, with the values following closely the reference ones. The biggest discrepancy happens in the same area of the domain for both schemes: the prediction of the peak values of v-velocity, with the upwind scheme under-predicting them and the LIN over-predicting them. Au contraire, no discrepancies exist in the u-velocity values, especially when linear interpolation was used. The UDS simulation's agreement with the reference values is slightly worse, likely owing to its numerical diffusion. All things considered, it is safe to say that the u-velocity seems to be well predicted by both schemes.

In the case where $Re = 1000$, a 160 by 160 CV grid was chosen. This time, the results obtained by using the UDS seem to be far from the expected ones. This dissimilarity is linked to this scheme's numerical diffusion, which, in a test case where the v-velocity is characterized by two peaks near the

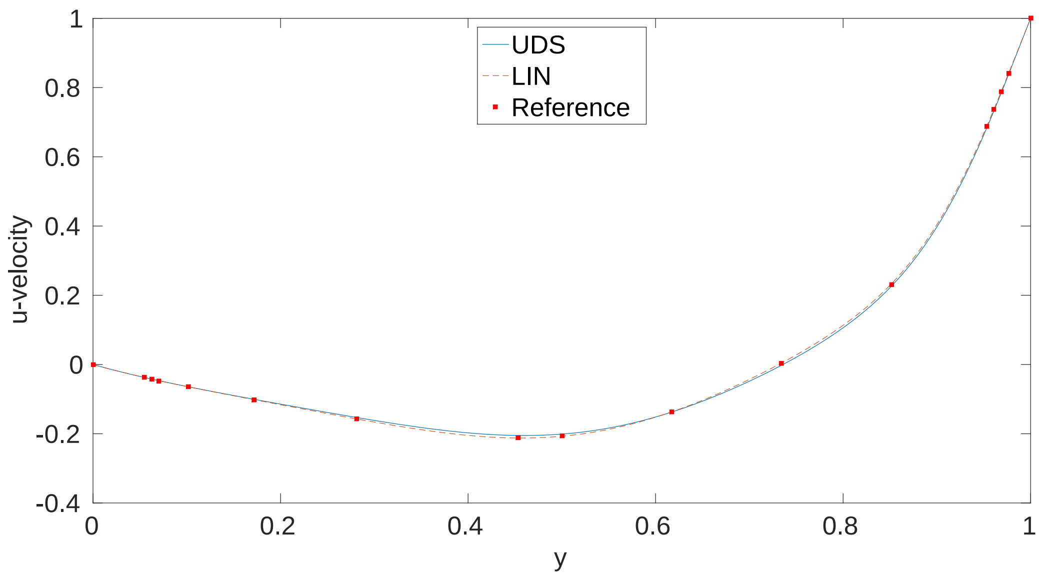


Figure 4.9: $Re = 100$: u velocity along vertical middle line for a 120x120 CV grid. Reference corresponds to references [32].

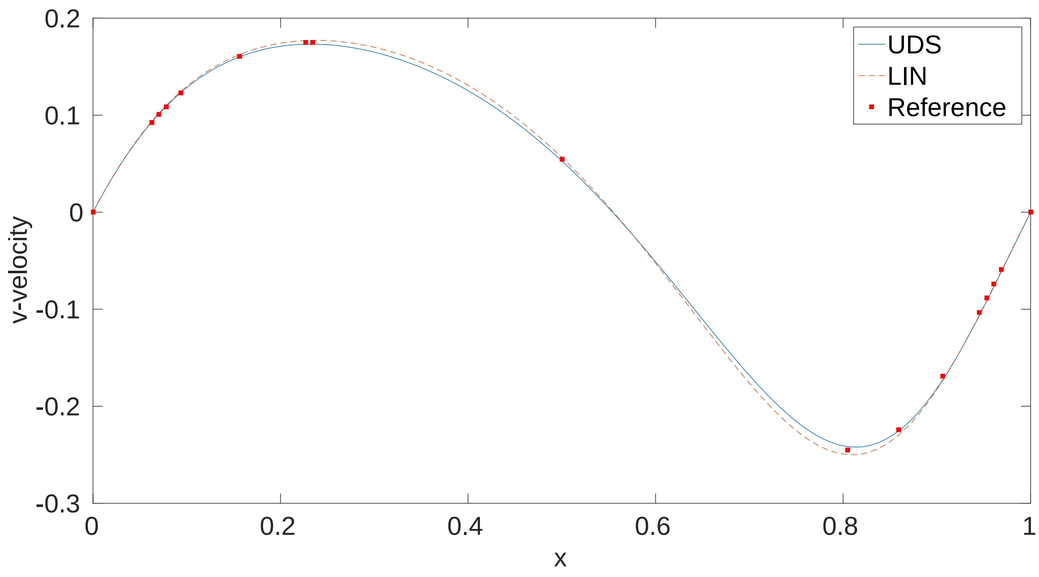


Figure 4.10: $Re = 100$: v velocity along horizontal middle line for a 120x120 CV grid. Reference corresponds to references [32].

walls, seriously hinders the accuracy of the simulations.

When the LIN is employed, the simulation's velocity values of both the u and v velocity are in-between those predicted by [32, 33] throughout the whole domain. This means that using the LIN proved to be worthwhile in this test case, as the flow features have been well captured without the rampant amount of points that the UDS would have required. In addition to this, no added stability issues were perceived when simulating the lid-driven cavity using linear interpolation.

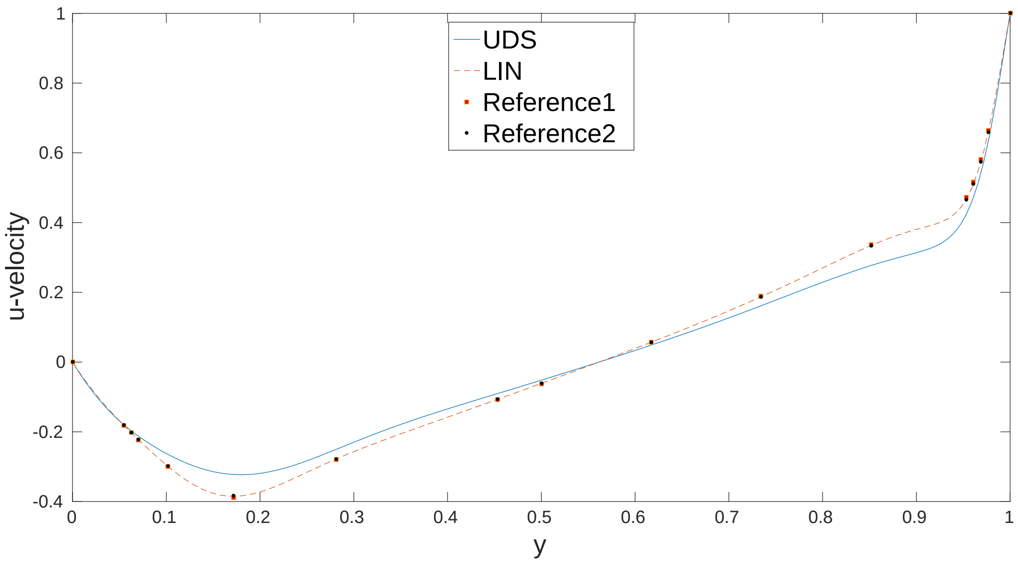


Figure 4.11: $Re = 1000$: u velocity along vertical middle line for a 160x160 CV grid. References 1 and 2 correspond to references [32, 33], respectively.

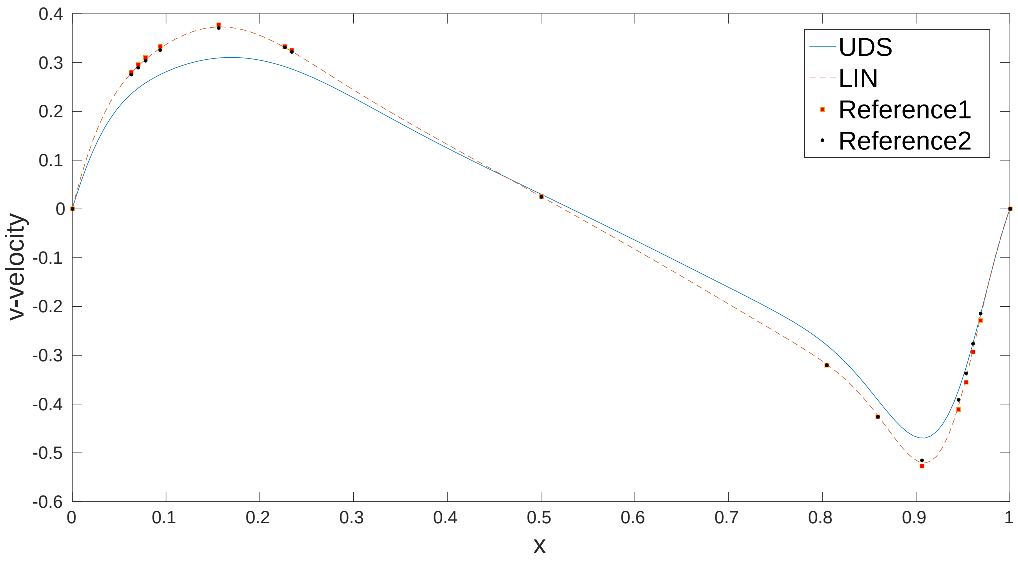


Figure 4.12: $Re = 1000$: v velocity along horizontal middle line for a 160x160 CV grid. References 1 and 2 correspond to references [32, 33], respectively.

Additional notes on the implementation procedure This test case was particularly useful to make sure the the momentum and pressure correction equations were rightfully implemented. It was so, because the lid-driven cavity flow is approximately isothermal, so it can be simulated without solving an energy equation. This means that said could be left unsolved with no impact on the final solution. Eventually, it could be reintegrated in the algorithm after the momentum and pressure correction equations' implementations were made correctly.

Tm	Pr	ϵ	p_o	μ	k	viscosity
293	0.71	0.034	100000	$1.7956 \cdot 10^{-5}$	0.02540	constant
600	0.71	0.6	101325	$2.954565 \cdot 10^{-5}$	0.04180	constant and variable

Table 4.2: Convection driven cavity initial conditions.

4.2.2 Convection Driven Cavity

The second 2D test case to be simulated was the convection driven cavity. Once again, the fluid is trapped inside a square cavity, yet, instead of being set in motion by the movement of top wall, the west and east walls' temperatures (T_h and T_c, respectively) are specified. The temperature difference sparks a density change in the fluid, which, due to the force of gravity, makes the fluid rotate in the cavity (whose remaining walls are treated as adiabatic).

To set up the convection driven cavity correctly and compare the results to the ones available in the bibliography [35–41], two non-dimensional numbers must be calculated: The Rayleigh number $Ra = Pr\frac{\rho_o^2 g(T_h-T_c)L^3}{T_o\mu_o^2}$ and the non-dimensional temperature difference: $\epsilon = \frac{T_h-T_c}{T_h+T_c}$.

When computing Ra, all the values are the initial condition values and the Prandtl number is 0.71 (for air). Density can be calculated from the initial pressure and temperature values, by using the ideal gas equation. The dynamic viscosity can be computed using Sutherland's law $\mu(T) = 1.68 \cdot 10^{-5}\left(\frac{T}{273}\right)^{\frac{3}{2}}\frac{273+S}{T+S}$, with $S = 110.5K$ or from a given kinematic viscosity $\mu = \nu\rho$, and the thermal conductivity using the definition of the Prandtl number $Pr = \frac{c_p\mu}{k}$. These viscosity and thermal conductivity values may or not vary during the simulation, meaning the there are two variants of this test case. The initial values for both variants are listed in table 4.2. In both cases, by fixating Ra and ϵ, the size of the cavity L can be computed.

Convection Driven Cavity $\epsilon = 0.034$

To analyse the accuracy of the $\epsilon = 0.034$ simulations, the adimensionalized velocity profiles in the middle sections are displayed in figures 4.14 and 4.15 and the temperature contour plots in figures 4.13. The non-dimensional velocity values are given by: $\frac{v}{\sqrt{g\beta\Delta TL}}$, with $\beta = \frac{\rho R}{p}$. In addition, the average Nusselt number in the hot and cold walls were also calculated (since $\overline{Nu}_w \simeq \overline{Nu}_e$ in all cases, the average value was considered) and documented in tables 4.4 and 4.5, using both upwind and linear interpolation . This way, a more general inspection of the flow field can be made through the plotted variables and a quantitative confirmation on the correctness of the solution can be made through the Nusselt number, which is calculated as: $\overline{Nu} = \frac{1}{L}\int_{y=0}^{y=L} Nu(y)\,\mathrm{d}y$, where $Nu(y) = \frac{L}{K_o(T_h-T_c)}k\frac{\partial T}{\partial x}|_w$.

The integral was computed using Simpson's rule and the temperature derivative via either forward or backward finite differences using 2 points.

The limiting reference values in table4.4 for $\epsilon = 0.034$ have been obtained by [36–39], using the Boussinesq approximation (table 4.3 includes all of these values), as well as the velocity profiles obtained by [35, 36]. Despite the differences in the modelling equations, generally speaking, the results are

Ra	[36]	[37]	[38]	[39]
10^3	1.114	1.108	1.118	1.105
10^4	2.245	2.201	2.243	2.302
10^5	4.51	4.43	4.519	4.646
10^6	8.806	8.754	8.799	9.012

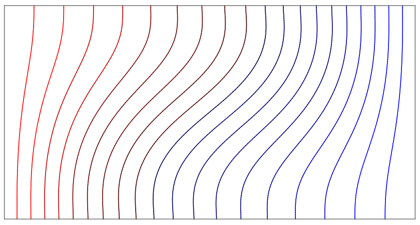
Table 4.3: Reference values for various Rayleigh numbers.

Grid	Ra	ϵ	$\overline{Nu}$	Reference $\overline{Nu}$	α_u	α_T	α_p	iterations
100x100	10^3	0.034	1.1330	1.105 - 1.119	0.8	1.0	0.4	1669
100x100	10^4	0.034	2.2683	2.201 - 2.302	0.8	1.0	0.4	965
40x40	10^5	0.034	4.7242	4.430 - 4.646	0.8	1.0	0.4	225
60x60	10^5	0.034	4.6492	4.430 - 4.646	0.8	1.0	0.4	369
80x80	10^5	0.034	4.6207	4.430 - 4.646	0.8	1.0	0.4	563
100x100	10^5	0.034	4.6064	4.430 - 4.646	0.8	1.0	0.4	887
80x80	10^6	0.034	9.1784	8.754 - 9.012	0.8	0.9	0.4	6871
100x100	10^6	0.034	9.0985	8.754 - 9.012	0.8	0.9	0.4	1191
120x120	10^6	0.034	9.0526	8.754 - 9.012	0.8	0.9	0.4	1566
140x140	10^6	0.034	9.0234	8.754 - 9.012	0.8	0.9	0.4	1591

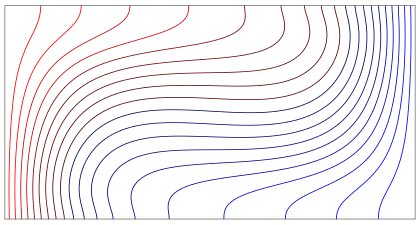
Table 4.4: Constant viscosity convection driven cavity results with the UDS.

Grid	Ra	ϵ	$\overline{Nu}$	its
40x40	10^3	0.034	1.1300	357
100x100	10^3	0.034	1.1291	1687
100x100	10^4	0.034	2.2641	965
100x100	10^5	0.034	4.5854	885
100x100	10^6	0.034	9.0082	1266
140x140	10^6	0.034	8.9597	2095

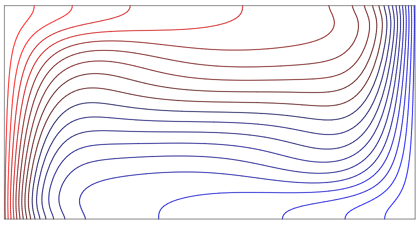
Table 4.5: Constant viscosity convection driven cavity results with the LIN.



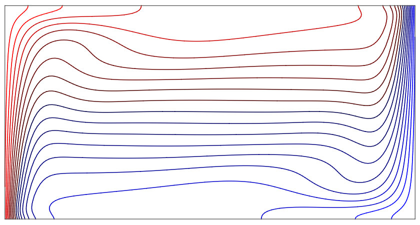
(a) $Ra = 10^3$



(b) $Ra = 10^4$



(c) $Ra = 10^5$



(d) $Ra = 10^6$

Figure 4.13: Temperature contour plots for several Rayleigh numbers ($\epsilon = 0.034$).

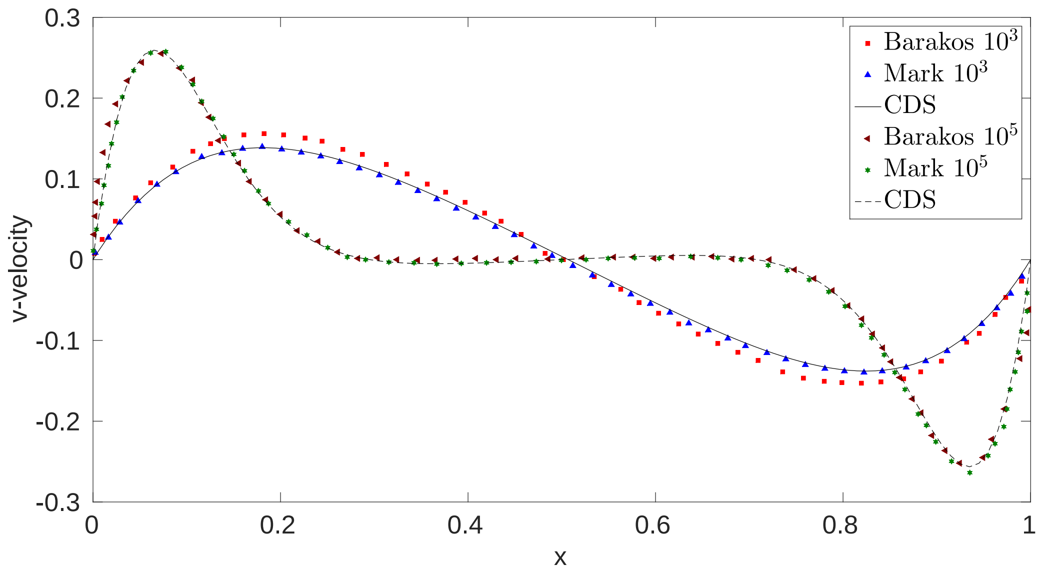


Figure 4.14: V velocity along horizontal middle line for $Ra = 10^3$ and $Ra = 10^5$ ($\epsilon = 0.034$). References 1 and 2, correspond to [35, 36], respectively.

satisfactory, since the average Nusselt number and velocity profiles are almost always close to, or even inside the interval of reference values.

The biggest divergence in surprisingly happens for $Ra = 10^3$, where the velocity gradients are smaller and farthest from the wall, when compared to the other simulations. The non-dimensionalized velocity profile follows the one obtained by [35] very closely, seemingly underpredicting the velocity values, by comparison to the one obtained by [36]. This in itself might be accountable for the difference in the prediction of the average Nusselt number on the wall, and, even for more refined meshes, the results did not change much.

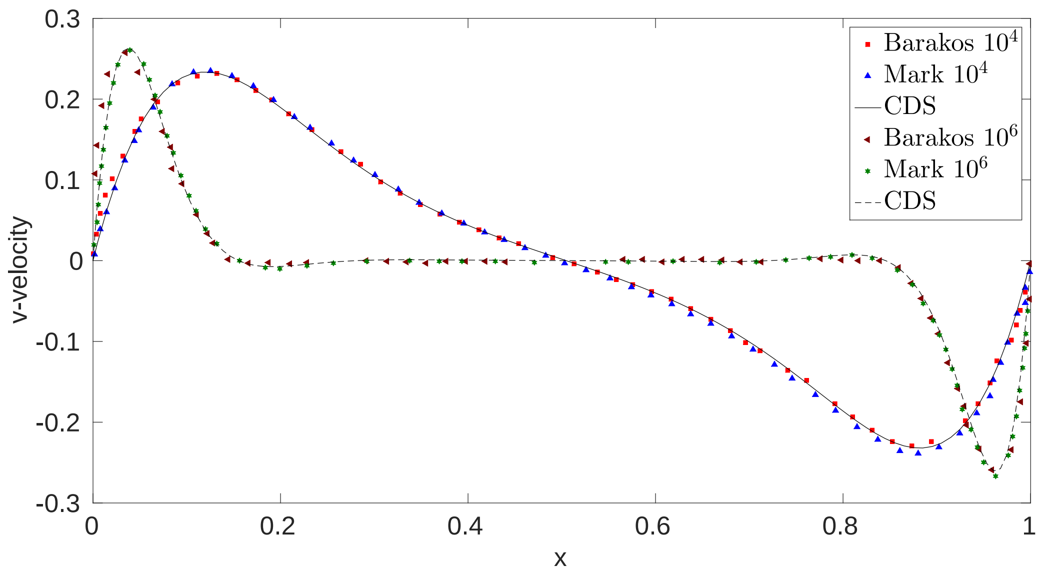


Figure 4.15: V velocity along horizontal middle line for $Ra = 10^4$ and $Ra = 10^6$ $(\epsilon = 0.034)$.References 1 and 2, correspond to [35, 36], respectively.

A grid convergence study was made for $Ra = 10^5$ and $Ra = 10^6$. The gradients near the walls for all variables tend to increase with the Rayleigh number, something that, until $Ra = 10^5$ is reached, doesn't seem to alter the nature of the simulation greatly, so, by doing the grid convergence for this value of the Ra, the grid size for all the lower values was set. However, upon reaching $Ra = 10^6$, the flow becomes much harder to predict, which manifests itself in oscillations in the residuals, justifying the need to decrease the under relaxation factors (values much smaller than the ones in table 4.4 have been tried, but the oscillations persisted). An increase in the number of grid points to obtain a reasonable accuracy was also required, which, when added to the oscillatory behaviour of the residuals greatly lengthened the simulation time for these simulations. The final 140x140 grid was used as a starting point in the simulations where $\epsilon = 0.6$, soon to be discussed, where a possible reason for this change in flow demeanour is also presented.

The temperature contour plots for the several simulations demonstrate that, as the Rayleigh number is increased, convection becomes increasingly important (relative to conduction), especially near the walls, where the velocities are largest. For $Ra = 10^6$, the once diagonally aligned lines at the center obtained for $Ra = 10^3$ have become completely horizontal, meaning that the heat transfer in the central area is negligible, when compared to the one near the walls (further demonstrating the increase in importance of convection).

Comparing the discretization schemes, the linear interpolation (as expected) always outdid the upwind differencing scheme, to the point that, in the $Ra = 10^3$ case, the results using LIN and a 40x40 grid, proved to be better than the results using the UDS with a 100x100 CV grid, thoroughly justifying the extra iterations needed to converge. This way, the results for upwind differencing will not be considered in the next section.

Grid	Ra	ϵ	$\overline{Nu}$	Reference	Viscosity
100x100	10^5	0.6	4.5685	45629	Constant
160x160	10^6	0.6	8.9045	8.85978	Constant
160x160	10^6	0.6	8.7566	8.6866	Variable

Table 4.6: $\epsilon = 0.6$ flow for constant and variable viscosities (LIN).

Convection Driven Cavity with $\epsilon = 0.6$

The $Ra = 10^6$ and $\epsilon = 0.034$ convection driven cavity flow is almost turbulent, according to [36], who used the $k - \epsilon$ model to predict transition. By increasing ϵ to $\epsilon = 0.6$, the velocity tends to increase as well, even if the cavity is smaller and putting at even bigger risk the laminar flow assumption. However, a benchmark solution for laminar flow assuming constant and variable viscosity in these conditions is available [40], to which the results obtained are compared in table 4.6.

When simulating the $\epsilon = 0.6$ cavity flow by assuming steady state, the residuals had a curious behaviour: convergence until 10^{-5} was achieved, but the residuals halted before reaching 10^{-6} and started fluctuating, never again decreasing (like the ones show in figure 4.16). This was accompanied by average Nusselt values that were far from the reference ones. On the other hand, by assuming transient flow, the results show a much better agreement with the reference ones, and the residuals converged until they reached the precision of the linear system solver. The convergence criterion for the unsteady simulations was $\phi_0^n - \phi^{n+1} < 10^{-4}$, where $\phi_0^n - \phi^{n+1}$ is the maximum difference between the converged value of the previous time step and the first value of the next time step (throughout the whole domain). This criterion guarantees that the maximum difference in values between two consecutive time steps is small enough and was applied to all variables.

This stability issue is thought to be related to the fact that, as mentioned above, the flow is likely to be approaching (or even to be in actual) transition from laminar to turbulent, thus generating a small instability unable to be solved in steady state and leading to a different solution.

The contour plots for density obtained with each assumption are shown in figure 4.17. As can be seen, there is a substantial difference in the behaviour of the flow, thus justifying the inequality in results when considering steady and transient regime.

It is also noteworthy that the both UDS and LIN were employed and the behaviour was found to be independent of the discretization scheme.

The reference values for $Ra = 10^5$ were taken from [42], who studied the behaviour of several solvers when simulating this test case, and were included in this section because the same oscillating demeanour was observed in the residuals.

As can be seen, for $Ra = 10^5$ the results show an extremely good agreement with the reference ones, without needing a very refined mesh. Meanwhile, in the two $Ra = 10^6$ test cases, the results are not as concurrent with the reference values. This discrepancy, although smaller than 1% of the reference value, must be given a reason for. This reason might be attributed to the difference in mesh refinement, since the finite volume low-Mach solvers ran on a very thin 512x512 mesh.

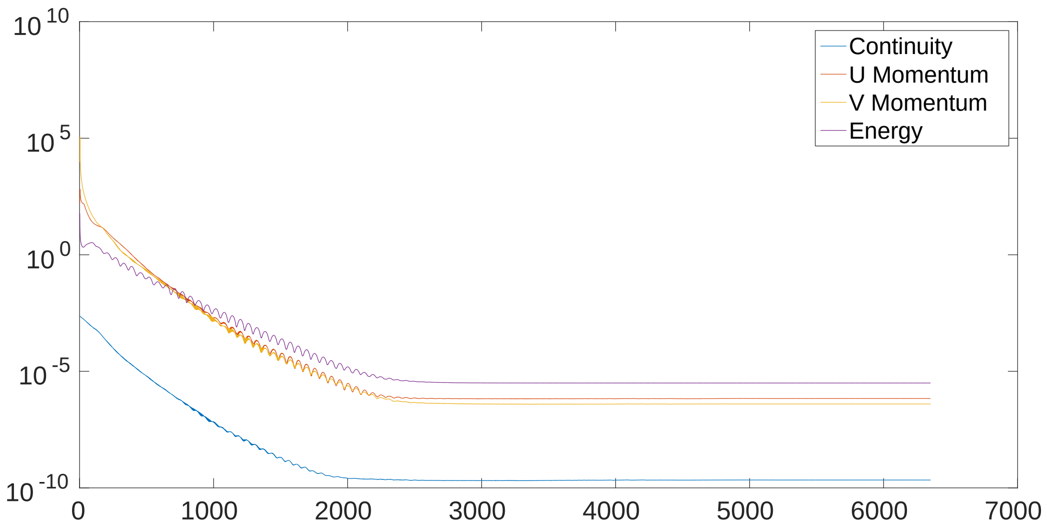


Figure 4.16: Residuals for steady state computation of $Ra = 10^6$ $\epsilon = 0.6$ convection driven cavity.

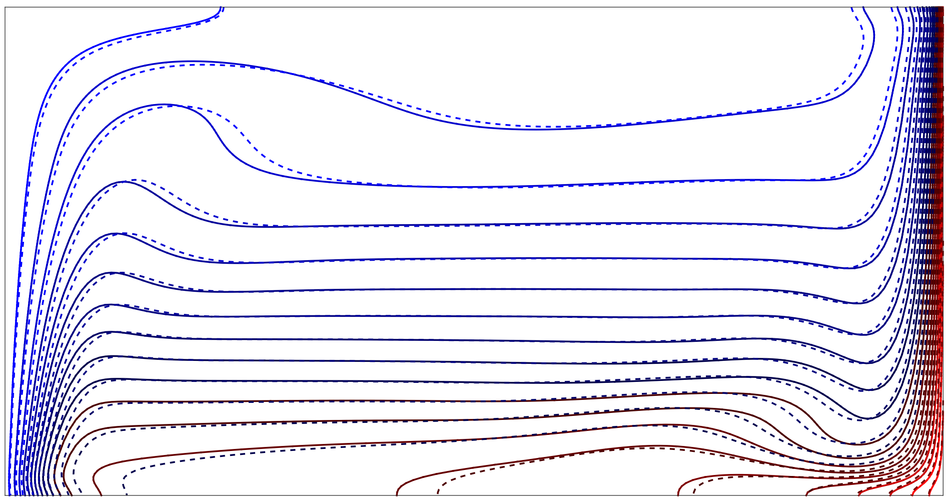


Figure 4.17: Density contour plots for $\epsilon = 0.6$ convection driven cavity. Dashed contours represent the unsteady solution, while the filled contours represent the steady one.

Additional Notes The inclusion of a variable viscosity example served the purpose of testing the implementation of the extra terms in the viscous stress tensor of the momentum equation, an implementation, which, can be considered to have been successful. Since the flow is subsonic, the energy dissipation due to viscosity can still be neglected, as mentioned in [40].

4.3 2D Supersonic Flow

4.3.1 Inviscid Supersonic Flow Over Forward Facing Step

In this test case, an inviscid Mach 3 fluid flows over a forward facing step. It is a well documented case and a good test for the solver developed, as it features several shock waves. This problem needed to be simulated assuming unsteady flow, in order to be compared to the results in the literature at $t = 4s$, before it reaches steady state at around $t = 12s$. It is important to note that the algorithm was found to be unable to simulate this flow without the added stability provided by the unsteady terms.

The fluid's properties are different from the usual: $\gamma = 1.4$, $R = 0.714 J/KgK$, and $c_p = 2.499 J/KgK$, allowing simplified boundary conditions: $T = 1K$, $p = 1Pa$, $u = 3m/s$, which are both the initial values and the ones specified at the inlet. At the outlet, the supersonic outflow boundary condition is used, and, every wall in the domain is a symmetry plane (including the step).

The geometry of this test case is depicted in fig.4.18. As mentioned by [46] and further detailed by [47] and [48] , the top corner of the step is a paramount region of this test case, for it generates numerical errors, which, if not corrected, will impact the solution. This correction is only valid when the flow near the corner becomes steady state, and the procedure to implemented is fully detailed by [47] , but, in a synthesized manner:

1 An entropy correction is applied on the first few CVs to the east east of the corner of the step, making sure that the entropy on both sides of the step is equal.

2 An enthalpy correction is then applied to the same CVs, to guarantee enthalpy conservation when the fluid crosses the corner, which alters the velocity magnitudes, pressure and the temperature in these CVs.

One of the goals of this study is to analyse how important this correction really is.

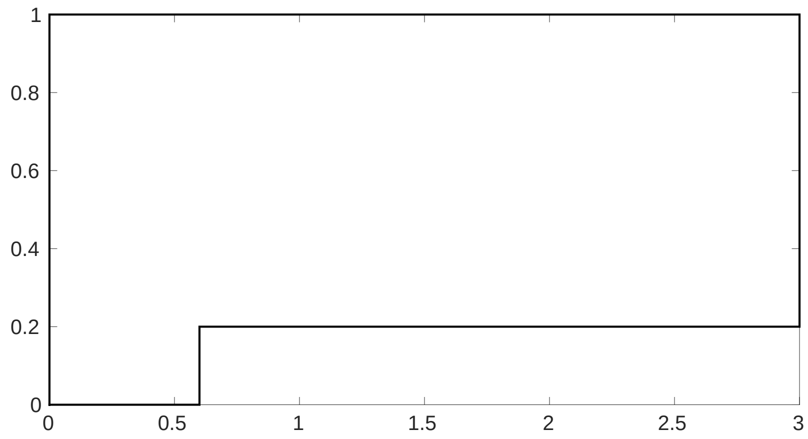
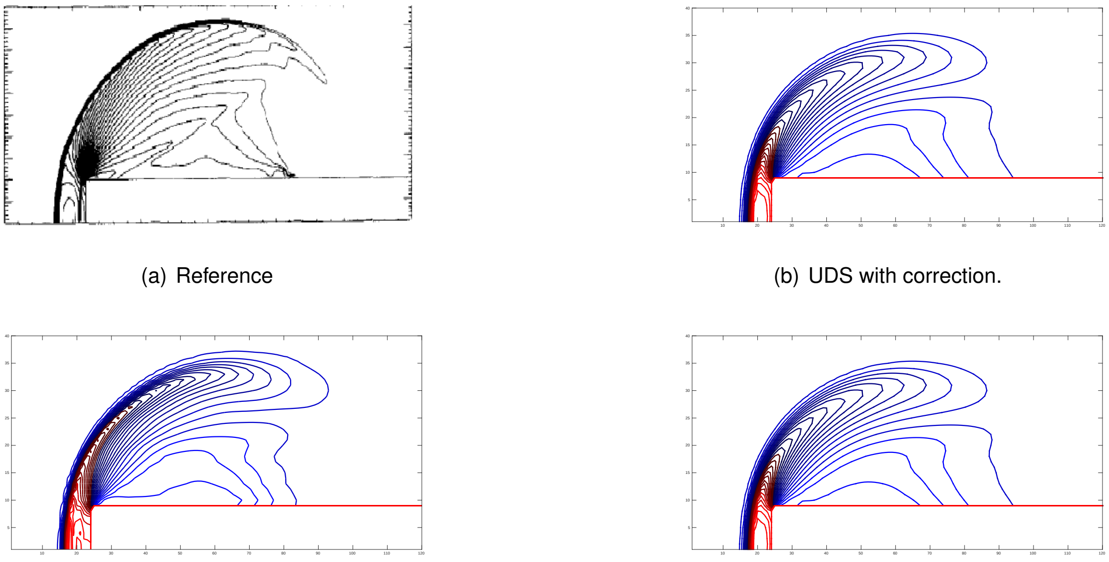


Figure 4.18: Mach 3 flow over step's domain geometry.

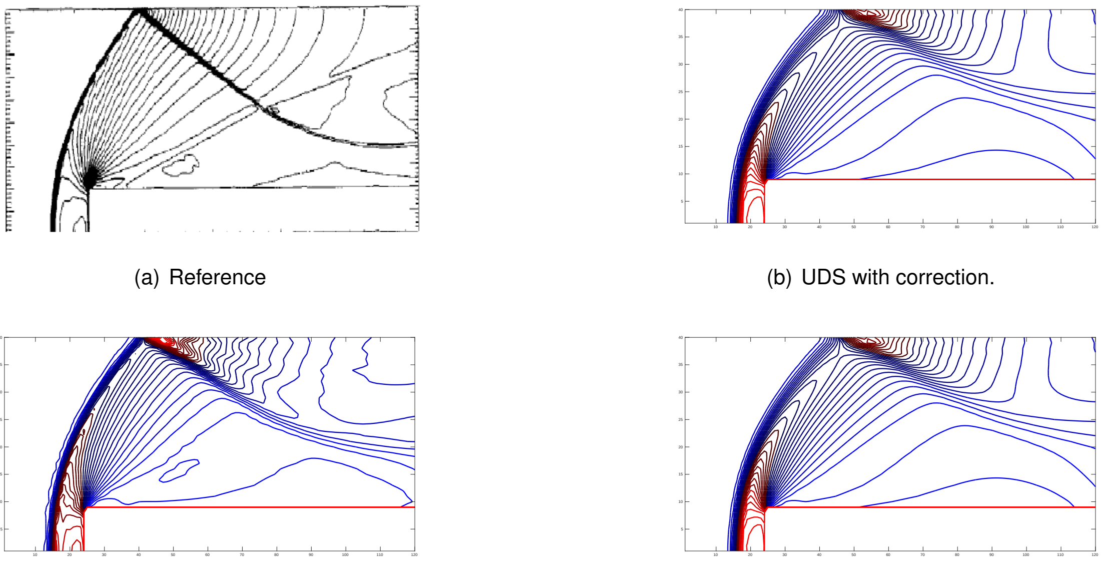
Results

To compare this result to [46], the evolution of the solution at every $0.5s$ is depicted in figures 4.19 to 4.26. It is important to confront the results where the enthalpy and entropy corrections were applied (starting at around 1.5s) near the corner, with the results obtained when no corrections were made. The residuals were impacted when the entropy and enthalpy values started being corrected, (perhaps due to the segregated structure of this solver), yet, no visible stability issues occurred, during these simulations. The comparison between different schemes was achieved using $\Delta x = \Delta y = \frac{1}{40}$, by drawing 30 density 30 contours, while using a courant number of 0.8. This grid is less refined than the one used by the reference, due to the large number of simulations and the long time they take to converge, yet, the qualitative behaviour of the schemes can still be assessed .


(a) Reference (b) UDS with correction.

(c) Gamma scheme with correction. (d) UDS without correction.

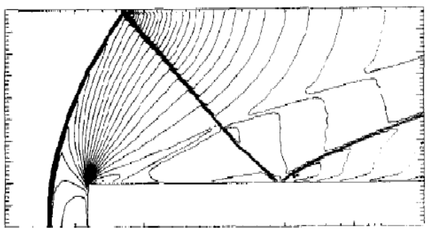
Figure 4.19: Density contours at $t = 0.5s$.


(a) Reference (b) UDS with correction.

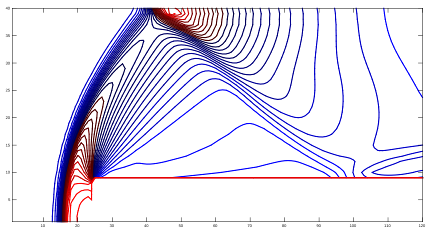
(c) Gamma scheme with correction. (d) UDS without correction.

Figure 4.20: Density contours at $t = 1$.

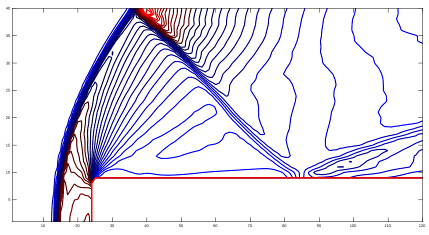
At $t = 0.5s$, not many remarkable differences exist between the simulations, as all of them seem to follow the reference evolution relatively well. At $t = 1s$, the weak shock generating at the corner of the



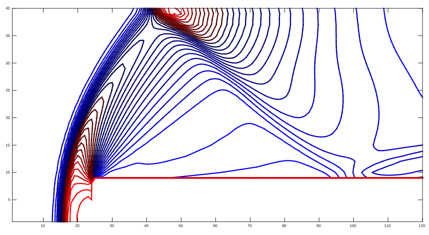
(a) Reference



(b) UDS with correction.

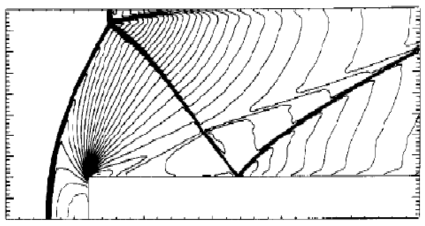


(c) Gamma scheme with correction.

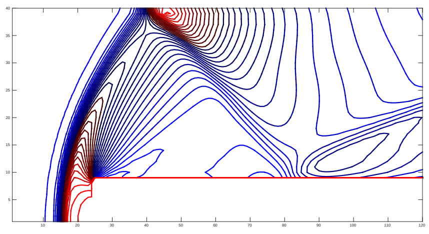


(d) UDS without correction.

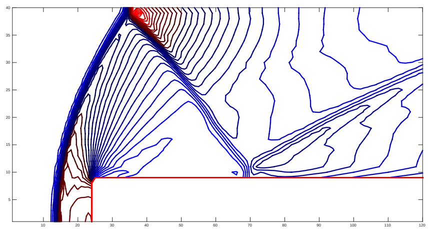
Figure 4.21: Density contours at $t = 1.5s$.



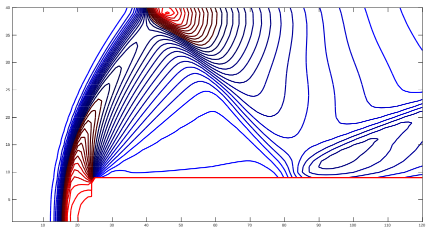
(a) Reference



(b) UDS with correction.

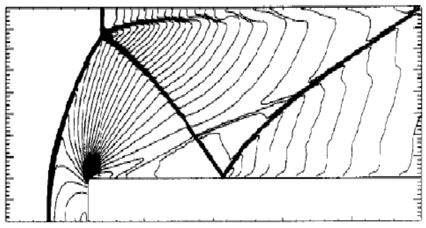


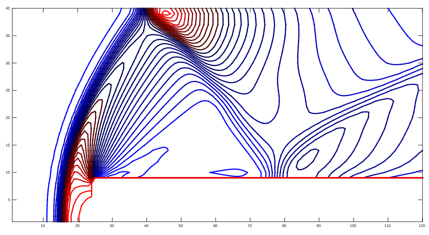
(c) Gamma scheme with correction.

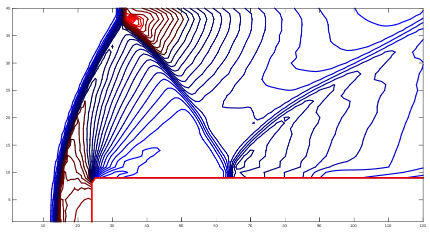


(d) UDS without correction.

Figure 4.22: Density contours at $t = 2s$.


(a) Reference


(b) UDS with correction.


(c) Gamma scheme with correction.

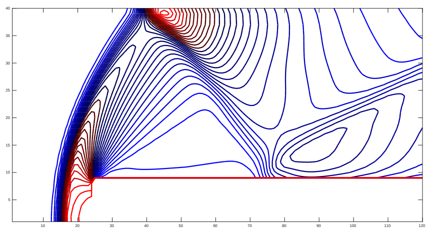
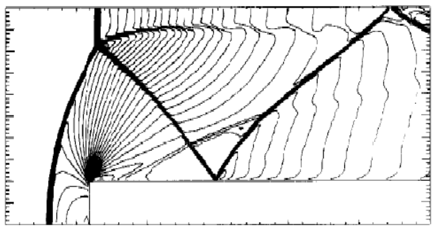
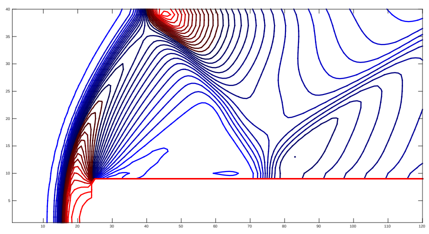
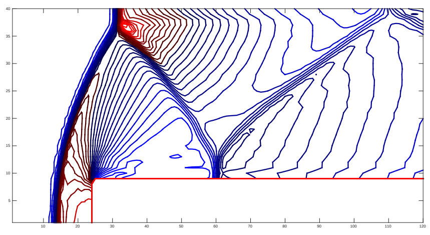

(d) UDS without correction.

Figure 4.23: Density contours at $t = 2.5s$.


(a) Reference


(b) UDS with correction.


(c) Gamma scheme with correction.

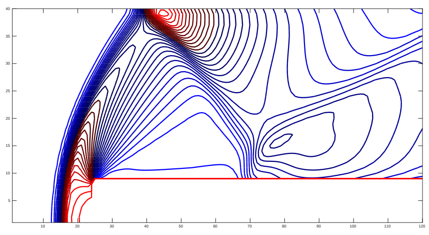
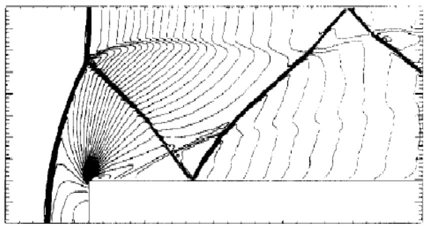
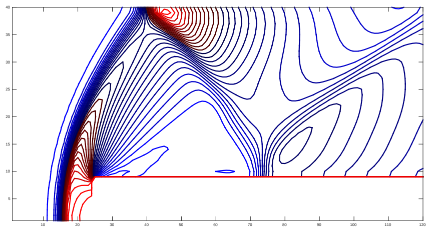
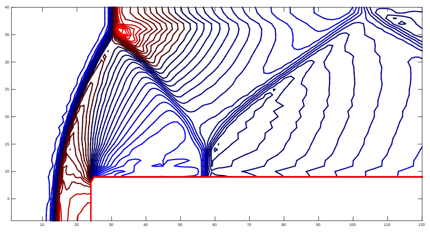

(d) UDS without correction.

Figure 4.24: Density contours at $t = 3s$.


(a) Reference


(b) UDS with correction.


(c) Gamma scheme with correction.

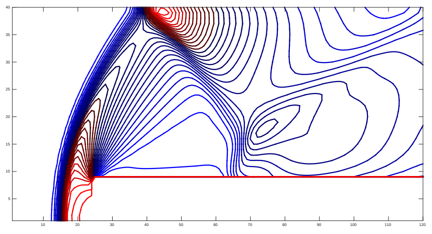
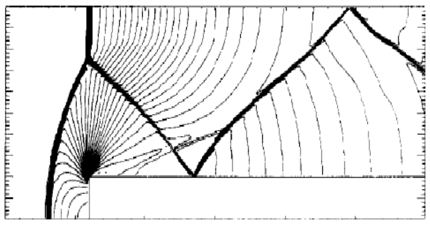
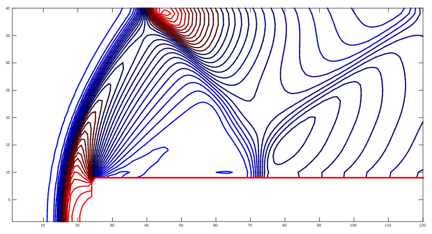
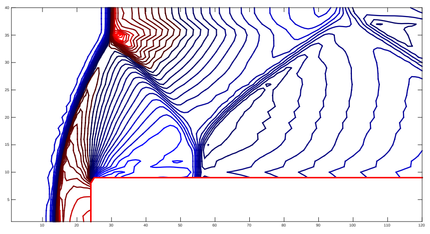

(d) UDS without correction.

Figure 4.25: Density contours at $t = 3.5s$.


(a) Reference


(b) UDS with correction.


(c) Gamma scheme with correction.

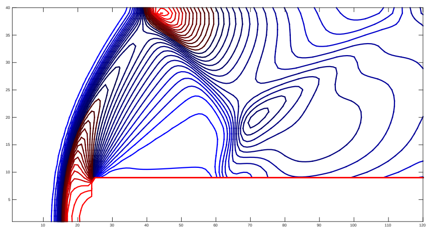

(d) UDS without correction.

Figure 4.26: Density contours at $t = 4s$.

step was not predicted by any of the simulations, and, especially the ones ran using the UDS, seem to smear the shock was significantly, thickening them.

When arriving at $t = 1.5s$, the gamma scheme still seems to follow the reference solution closely, but both UDS simulations are clearly lagging behind, seeing that the position of the shock reflection on the bottom wall is dislocated in the downstream direction.

A Kelvin-Helmholtz instability starts to appear at $t = 2s$ near the reflection on the top wall, which begins to separate itself from said wall. However, none of the simulations ran appear to recognize this structure at this time. In the case of the Gamma scheme simulation, this is likely to be due to the less refined grid. The simulations ran with the UDS are lagging behind even more clearly, in what seems to be an accumulation of numerical errors due to the diffusion, that has been increasing throughout each time step.

The Gamma scheme simulation begins capturing the instability on the top wall at around $t = 2.5s$, with the shock reflection also being separated from the top wall in a more distinguishable way. On the other hand, it seems to be lagging behind slightly, when compared to the reference, as the shock moving upwards across the outlet still hasn't reached the top right corner of the domain.

At time $t = 3s$, the UDS simulation ran without the correction, starts to become discernible from the one with correction, with the shock reflection on the lower wall seeming to become slanted and separated from the wall. These are the first visible effects of the numerically induced boundary layer. At the same time, the gamma scheme also features a increase in the height of that shock reflection.

The UDS simulation without correction begins to stand out even more at time $t = 3.5s$, with a huge increase in the height of the shock reflection of the bottom wall, while the corrected UDS simulation lags behind the reference, with the shock at the outlet only reaching the to right corner at this time.

At the end of the simulation ($t = 4s$), the Kelvin-Helmholtz instability has subsided in the reference solution, unlike in the Gamma solution, where it can still be traced. On the other hand, the shock induced at the corner, has not been captured by this scheme, and the shock reflection on the bottom is too vertically stretched.

The UDS scheme, when used without simulation predicts the shock reflection on the bottom wall to be very distorted and separating from the wall at this time. The corrected UDS simulation does not feature this separation, yet, it is clearly lagging behind

Final Remarks The Gamma scheme is clearly the one that follows the evolution of the reference solution closest. However, it never seemed to be able to capture the weak shock originated at the corner, and the instability on the top wall has been predicted to both start and end at a later time. Other than that, the shock reflection on the bottom wall is stretched vertically. These imperfections are likely to be due to the less refined grid.

The corrected UDS simulation was not able to capture most of the structures of the flow, due to the errors due to diffusion accumulating with time, which led to a solution that is closer to the reference one at $t = 2.5s$, than at the final time.

Correcting the entropy and enthalpy values around the corner of the step, definitely has an impact on the solution, since, without doing it, the numerical boundary layer pushed and distorted the shock reflection at the bottom wall.

4.3.2 Supersonic Flow Over Flat Plate

The Mach 4 flow over a flat plate is a classical example of a laminar viscous supersonic flow that is simple to set up geometry-wise, and features variable viscosity. As mentioned by [34], there is no exact analytical solution for this flow, yet, it still is an appealing problem, since it has real-life applications. Additionally, to solve it correctly, the viscous stress terms (2.55) must be included in the energy equation.

It was noted in section 4.2.1 that, as Re increases, the ammount of points required for the solution to have the desired accuracy increased, leading to longer simulations. This fact, allied to the need to be sure the flow is fully laminar, let to the choice by [34] to stick to $Re = 1000$.

Having decided on the value for Re, choosing the inlet values for pressure and temperature to be $T_\infty = 288.16K$ and $p_\infty = 101325Pa$, along with $Pr = 0.71$ and $\mu_o = 1.7894 \cdot 10^{-5}$, the plate length can be calculated: $L = 10^{-5}m$.

The height of the domain is trickier to compute, however: both the shock wave and the boundary layer must fit inside the domain, and the top boundary should be sufficiently far from these flow structures not to meddle in their formation. The domain's height was then calculated as being 5 times larger than the boundary layer thickness: $H = 5\delta$, with $\delta = \frac{5L}{\sqrt{Re}}$(Blasius).

The boundary conditions of this simulation are the standard boundary conditions for supersonic flow: supersonic inlet, supersonic outflow at the outlet and at the top, and the lower wall has a set temperature and is viscous.

Results

The results obtained for this problem will now be presented. To obtain them, the simulations where made by using the unsteady version of the equations, as, especially when the Gamma scheme was employed, the convergence proved to be complicated, due to instabilities. Additionally, the under relaxation factors

Pressure Profile Along the Wall The resultant non dimensional pressure profile $\left(p^* = \frac{p}{P_\infty}\right)$ on the viscous wall, will be first analysed, since it permits an evaluation of the solution over the whole flat plate, and not only at the outlet. This result is depicted in figure 4.27 along the grid location x, and compared to the reference values. It can be observed that there are some discrepancies, as the reference profile features a much lower pressure peak, which appears to be dislocated towards the downstream direction of the flow. This peak is followed by an oscillation, which was not captured in any of the simulations ran. Two possible reasons for the disparity of results are conjectured: The first one is that the reference simulation uses the finite difference framework, which allows the prescription of pressure at the corner of the domain, unlike in FV, where the value is prescribed at the face.

The second, more likely, possibility is that, seeing that this is the point where the flow first meets the viscous wall, it is the start of a strong (although rapidly subsiding) shock, and that the simulation ran by [34] used the MacCormack scheme, a non-physical oscillation might have been created in this area. This has been recorded to happen near discontinuities for this scheme, as depicted by [49].

On the other hand, the UDS is highly diffusive, which could allow the oscillation (if it is physical) to be filtered. Thus, to confirm this wasn't a grid or scheme related issue, several simulations were ran, featuring more refined grids, and the Gamma NVD, with the results seemingly converging and revealing a pressure peak at the inlet. Therefore, and seeing that, downstream of the shock, the reference solution overlaps the results obtained, the outlet results will be compared to verify whether this behaviour affects the flow downstream of it.

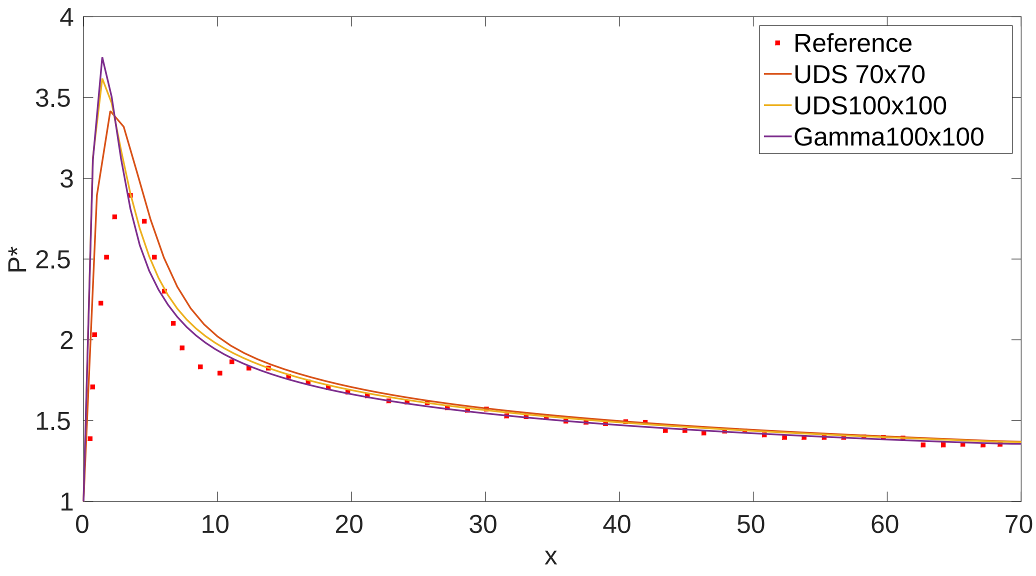


Figure 4.27: Non-dimensional Pressure profile at the lower wall of the Mach 4 flow over a flat plate.

Outlet Temperature Profile The outlet profiles will be compared by plotting them over the non dimensional y coordinate, as suggested by [34]: $y^* = \frac{y}{x}\sqrt{Re_x}$.

The outlet profile of non-dimensional temperature, $T^* = \frac{T}{T_\infty}$ is illustrated in figure 4.29. In this figure, the structure of the flow (figure 4.28) at the outlet is well defined, with the boundary layer (thermal, in this case) nearest to the wall, topped by the inviscid shock layer, which is limited by the shock wave itself.

Comparing the resultant values to the reference ones, there seems to be a general agreement on the profile structure, yet, the upwind scheme proved too diffusive to fully capture the decrease in temperature inside the inviscid shock layer. Furthermore, both the Gamma scheme and the UDS seem to overpredict the height at which the shock happens, and its wideness, with the Gamma NVD resembling the reference solution at a closer level than the UDS (as expectable). On the other hand, the small oscillation above the shock location predicted by the reference solution was not picked up by any of two the schemes.

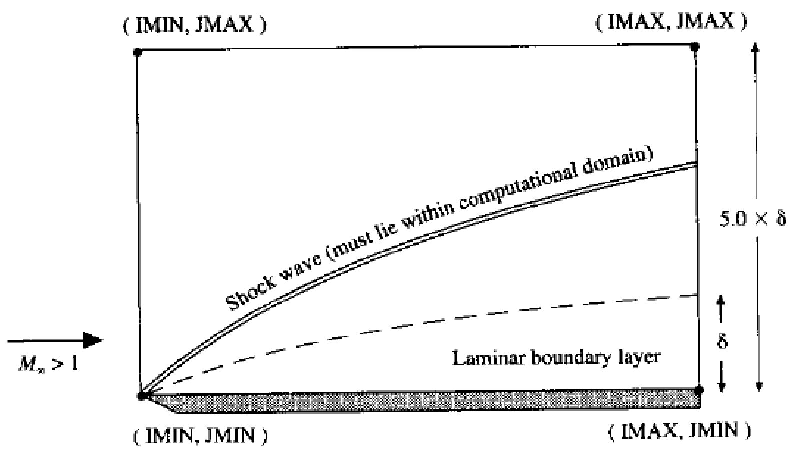


Figure 4.28: Structure of the supersonic flow over a flat plate (image taken from [34]).

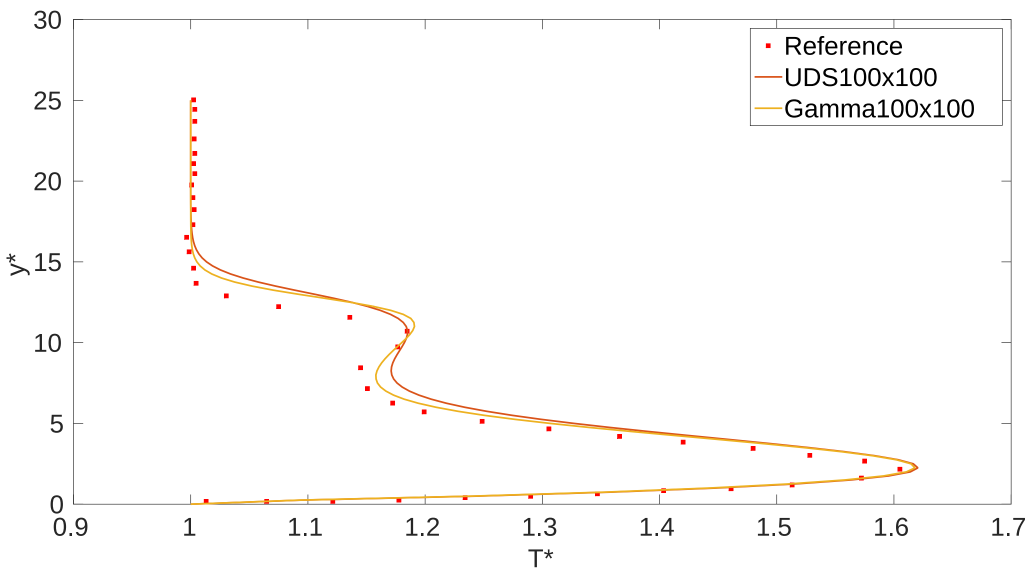


Figure 4.29: Non-dimensional Temperature profile at the outlet (Mach 4 flow over a flat plate).

Outlet Mach Profile The Mach profile at the outlet is detailed in figure 4.30. The results seem to be in close agreement with the reference ones, with the Gamma scheme outperforming the UDS in terms of accuracy.

The UDS smeared the shock and the small Mach number peak in the inviscid shock layer, much like what happened in the inviscid shock layer of the temperature profile.

Both the Gamma scheme and the UDS overpredicted the height at which the shock happens, but the Gamma scheme seemed to capture its thickness to a degree comparable to the one achieved by the reference solution. In fact, other than the previoulsy stated difference, the profile plotted with the Gamma scheme is in close agreement with the one obtained by [34].

In the outlet region above the shock wave, the Mach profile is constant, with all the simulations' results overlapping in that region.

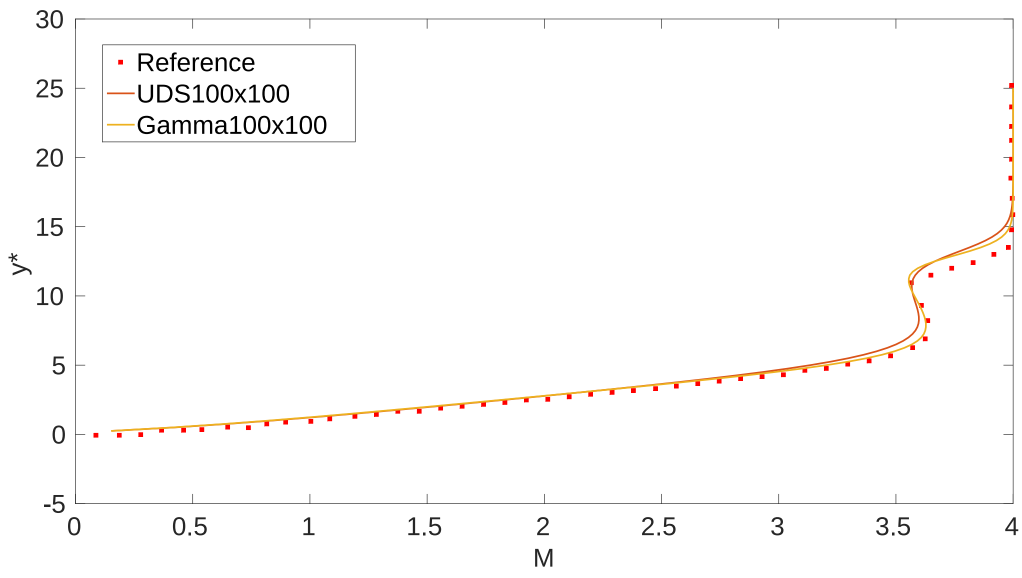


Figure 4.30: Mach profile at the outlet (Mach 4 flow over a flat plate).

Chapter 5

Conclusions

The SIMPLE algorithm and an extension to allow the computation of all-Mach flows have been implemented and tested in a 1D and several 2D cases with uniform grids. This was accomplished through the addition of several steps to the algorithm, as well as the expansion of the pressure correction equation, which takes the density changes into account when they are relevant by adding a convective term to the equation.

Stabilization of the pressure correction equation was achieved by adding a self-calculating relaxation factor. This was critical in the simulation of the converging-diverging nozzle, as the area variation across the domain frequently caused the matrices to become non-diagonally dominant. In this case, the role of the under relaxation factors to achieve convergence was found to be essential in the simulation of compressible flows.

The test cases allowed for a gradual assemblage of the algorithm, i.e. by starting with steady incompressible flow and gradually building up the algorithm to encompass viscous supersonic, and unsteady flows.

The implementation of the upwind, linear, and NVD Gamma convective schemes, as well as their applicability to the test cases that were solved have been documented. The upwind scheme provides stability to a level beyond any of the other schemes, yet, the amount of detail lost when applying it sometimes cannot be tolerated. The Gamma NVD scheme implemented allows a much better resolution, yet, especially at higher Mach numbers, it sometimes required the added stability of the unsteady terms to converge, for these test cases.

The use of uniform grids facilitated the process of discretization and implementation of the governing equations, yet, the increase in the amount of points necessary to capture the details of the flow, sometimes caused the simulations to become more expensive computationally.

Apart from the lid driven cavity flow, and the lower Ra convection driven cavity flows, all of the test cases had some quirkiness to them, e.g.: the inviscid flow over a step required a correction near the step to be made, and, the quasi-1D test case featured a variable area over the domain. These peculiarities, which complicated the simulation of these flows at first, served the purpose of generating a much more robust algorithm, and, in the end, led to a much better comprehension on how the implemented algorithms work.

All in all, the implementation of the algorithm can be considered to have been successful, as the results for both compressible, and incompressible flow simulations were close to the references'. This meaning that the algorithm can qualify to solve flows at all Mach numbers.

5.1 Future Work

The logical step after developing this study would be to implement it in SOL. To accomplish this, the solver would have to be adapted to unstructured grids, something that would not require any major changes in the algorithm itself (in terms of its main steps and their order). Since the code already features unstructured grid manipulation functions, and the specific schemes for these types of grids, the verification of this implementation could be done by simulating while using the same uniform grids in both codes.

A successful implementation in SOL would open the possibility of integrating the all-Mach solver with several high-order numerical schemes, such as the weighted least-squares, immersed boundary methods, and adaptive refinement algorithms. The latter would be very interesting in compressible flow, since grids could be intelligently refined near shocks, permitting a better resolution near these discontinuities while keeping the total number of cells as low as possible.

Another logical next step would be to expande the algorithm to allow the simulation of 3D flows could be considered, since all that needs to be done is include and extra momentum equation and adapt all the other equations to include the 3rd dimension. This would, perhaps, require some optimization, since 3D flows can be very demanding to solve.

At last, the algorithm could be adapted into a coupled all-Mach solver. The different options available when coupling the governing equations (i.e. which equations should be coupled) would have to be assessed and compared regarding their stability, speed of convergence and memory requirements.

Bibliography

[1] Joel H. Ferziger , M. Perić, *Computational Methods for Fluid Dynamics*, Springer, 3rd edition, 2002.

[2] L.Mangani, *Development and Validation of an Object Oriented CFD Solver for Heat Transfer and Combustion Modeling in Turbomachinery Applications*, Univerist degli Studi di Firenze, 2008.

[3] H. K. Versteeg, W. Malalasekera *An Introduction to Computational Fluid Dynamics*, Pearson Education Limited 1995, 2nd edition, 2007

[4] H. Jasak, *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows* PHD thesis, Imperial College, University of London, 1996

[5] R. W. MacCormack, H. Lomax, *Numerical Solution of Compressible Viscous Flows* Annual Review of Fluid Mechanics, vol. 11, pp.289-316 (1979)

[6] S. V. Patankar, D. B. Spalding, *A Calculation Procedure for Heat,Mass and Momentum Transfer in Three-Dimensional Parabolic Flows* Int. J. Heat Mass Transfer Vol. 15, pp. 1787-1806 (1972)

[7] S. V. Patankar, *Numerical Heat Transfer* Hemisphere, Washington, D.C. (1980)

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

[9] R. I. Issa, *Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting* Journal of Computational Physics 62, 40-65 (1985)

[10] D. S. Jang , R. Jetli, S. Acharya, *Comparison of the PISO, SIMPLER, and SIMPLEC Algorithms for the Treatment of the Pressure-Velocity Coupling in Steady Flow Problems* Numerical Heat Transfer: An International Journal of Computation and Methodology, 10:3, 209-228 (1986)

[11] C. M. Rhie , W. L. Chow, *A Numerical Study of the Turbulent Flow past an Isolated Airfoil with Trailing Edge Separation*, AIAA J., vol. 21, pp. 15251532, 1983.

[12] R. I. Issa, F. C. Lockwood *On the Prediction of Two-Dimensional Supersonic Viscous Interactions Near Walls*, AIAA JOURNAL VOL. 15, NO. 2 (1977)

[13] J. P. Van Doormal, G. D. Raithby, B. H. McDonald *The Segregated Approach to Predicting Viscous Compressible Fluid Flows*, Journal of Turbomachinery, vol.109, pp.268-277 (1987)

[14] K. C. Karki, S. V. Patankar *A Pressure Based Calculation Procedure For Viscous Flows at All Speeds In Arbitrary Configurations*, AIAA-88-0058 26th Aerospace Sciences Meeting January 11-14, Reno, Nevada (1988)

[15] W. Shyy, Mark E. Braaten *Adaptive Grid Computation For Inviscid Compressible Flows using a Pressure Correction Method*, 1st National Fluid Dynamics Conference Cincinnati, OH,U.S.A. (1988)

[16] M. H. Kobayashi, J. C. F. Pereira *Predictions of Compressible Viscous Flows at all Mach Number Using Pressure Correction, Collocated Primitive Variables and Non-Orthogonal Meshes*, AIAA-92-0426 30th Aerospace Sciences Meeting Exhibit, January 6-9, Reno, Nevada (1992)

[17] W. Shyy, M. Ghent,C. Sun *Pressure-Based Multigrid Algorithm for Flow at All Speeds*, AIAA Journal, Vol. 30, No. 11, November 1992

[18] C.M. Rhie, *Pressure-Based Navier-Stokes Solver Using the Multigrid Method*, AIAA Journal 1017, Vol. 27, NO. 8, August 1989

[19] C. H. Marchi, C. R. Maliska, *A Non-orthogonal Finite-Volume Methods for the Solution of All Speed Flows Using Co-Located Variables*, Numerical Heat Transfer, Part B, vol. 26, pp. 293-311, 1994

[20] I. Demirdzic , Z. Lilek , M. Peric, *A Collocated Finite Volume Method For Predicting Flows At All Speeds*, International Journal for Numerical Methods in Fluids, vol.16, 1029-1050(1993).

[21] F. S. Lien,M. A. Leschziner, *A Pressure-Velocity Solution Strategy for Compressible Flow and Its Application to Shock/Boundary-Layer Interaction Using Second-Moment Turbulence Closure*, Journal of Fluids Engineering, vol. 115, pp. 717-725, 1993.

[22] E. S. Politis, K. C. Giannakoglou, *A Pressure-Based Algorithm for High-Speed Turbomachinery Flows*, International Journal For Numerical Methods in Fluids, vol. 25, 63-80 (1997)

[23] M. Darwish , F. Moukalled, *A High-Resolution Pressure-Based Algorithm for Fluid Flow at All Speeds* Journal of Computational Physics 168, 101–133 (2001)

[24] I. Senocak, W. Shyy, *A Pressure-Based Method For Turbulent Cavitating Flow Computations* Journal of Computational Physics 176, 363-383 (2002)

[25] M. Darwish , I. Sraj, F. Moukalled, *A Coupled Incompressible Flow Solver on Structured Grids*, Numerical Heat Transfer, Part B: Fundamentals, 52:4, 353-371 (2007)

[26] M. Darwish , F. Moukalled, *A Fully Coupled Navier-Stokes Solver For Fluid Flow At All Speeds*, Taylor & Francis Group, LLC, 2013.

[27] M. Darwish , F. Moukalled, *An OpenFOAM Pressure-Based Coupled CFD Solver for Turbulent and Compressible Flows in Turbomachinery Applications*, Taylor & Francis Group, LLC, 2013.

[28] Z. J. Chen , A. J. Przekwas, *A coupled pressure-based computational method for incompressible/compressible flows*, Elsevier, 2010.

[29] C. Xiao , F. Denner , B. G .M. van Wachem, *Fully-Coupled Pressure-Based Finite-Volume Framework for the Simulation of Fluid Flows at All Speeds in Complex Geometries*, Journal of Computational Physics 346 (2017) 91–130.

[30] F. Moukalled, L. Mangani , M. Darwish *Implementation of Boundary Conditions in the Finite-Volume Pressure-Based Method—Part I: Segregated Solvers* Numerical Heat Transfer, Part B, Vol. 69, No. 6, 534–562 (2016)

[31] D. M. S. Albuquerque , J. M. C. Pereira , J. C. F. Pereira, *Residual Least-Squares Error Estimate for Unstructured h-Adaptive Meshes*, Numerical Heat Transfer, Part B: Fundamentals 67 (3), 187-210.

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

[33] O. Botella , R. Peyret, *Benchmark Spectral Results On The Lid-Driven Cavity Flow* Computers & Fluids Vol. 27, No. 4, pp. 421-433, 1998

[34] John D. Anderson, *Computational Fluid Dynamics*, McGraw-Hill Education

[35] A. Mark, E. Svenning, F. Edelvik, *An immersed boundary method for simulation of flow with heat transfer*, International Journal of Heat and Mass Transfer 56 (2013) 424–435

[36] G. Barakos, E. Mitsoulis, D. Assimacopoulos, *Natural Convection Flow in a Square Cavity Revisited: Laminar and Turbulent Models with Wall Functions*, International Journal for Numerical Methods in Fluids, vol. 18, 695-719 (1994)

[37] N. C. Markatos, K. A. Pericleous, *Laminar And Turbulent Natural Convection In An Enclosed Cavity*, Int. Journal of Heat and Mass Transfer, vol. 27. No. 5, pp. 755 772 (1984)

[38] G. De Vahl Davis *Natural Convection of Air In a Square Cavity, A Bench Mark Numerical Solution* International Journal For Numerical Methods In Fluids, VOL. 3, 249-264 (1983)

[39] T. Fusegi, J. M. Hyun, K. Kuwaharas and B. Farouk, *A Numerical Study of Three-Dimensional Natural Convection in a Differentially Heated Cubical Enclosure*, Int. Journal of Heat and Mass Transfer, vol. 34, No. 6. PP. 1543-1557 (1991)

[40] P. Le Quéré, C. Weisman, H. Paillere, J. Vierendeels, E.Dick, R.Becker, M. Braack, J. Locke,, *Modelling of Natural Convection Flows With Large Temperature Differences: A Benchmark Problem For Low Mach Number Solvers. Part1. Reference Solutions*, ESAIM: Mathematical Modelling and Numerical Analysis, Vol.339(3), pp.609-616 (2005)

[41] M. Lappa, T. Gradinscak, *On the oscillatory modes of compressible thermal convection in inclined differentially heated cavities*, International Journal of Heat and Mass Transfer 121 (2018) 412–436

[42] H. aillere, C. Viozat, A. Kumbaro, I. Toumi, *Comparison of low Mach Number Models for Natural Convection Problems*, Heat and Mass Transfer, vol. 36, pp. 567-573 (2000)

[43] J. P. P. Magalhães, D. M.S. Albuquerque, J. M.C. Pereira, J. C.F. Pereira *Adaptive mesh finite-volume calculation of 2D lid-cavity corner vortices*, Journal of Computational Physics, vol. 243, pp. 365–381 (2013)

[44] L. F. Bergamoa, E. M. Gennaro, V. Theofilisc, M. A.F.Medeirosa *Compressible modes in a square lid-driven cavity* Aerospace Science and Technology 44 (2015) 125–134

[45] F.Moukalled, L.Mangani, M.Darwish *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFoam and Matlab* Springer

[46] P.Woodward, P.Colella, *The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks* Journal of Computational Physics, vol. 54, p. 115-173 (1984)

[47] R. Donat, A. Marquina, *Capturing Shock Reflections: An Improved Flux Formula* Journal of computational physics, vol. 125, pp.42–58 (1996)

[48] R. Sanders, A. Weiser, *High Resolution Staggered Mesh Approach for Nonlinear Hyperbolic Systems of Conservation Laws* Journal of computational physics,vol. 101, pp.314-329 (1992)

[49] G. A. Sod, *A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws* Journal of computational physics, vol. 27, Issue 1 (1978)

[50] C. Hirsch, *Numerical Computation of Internal and External Flows Volume 2: Computational Methods for Inviscid and Viscous Flows* Wiley, 1990