UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

BACHELOR FINAL THESIS

Study for the computational resolution of conservation equations of mass, momentum, and energy. Possible application to different aeronautical and industrial engineering problems: Case GD1.

Document:

Report

Author:

Silvia Gómez Blanco

Director/Co-director:

Carles David Pérez Segarra / Asensio Oliva Llena

Degree:

Bachelor in Aerospace Technology Engineering

Examination session:

Spring, 2023

Abstract

This work consists of understanding the Navier Stokes equations by programming codes, made in MatLab, that simulate fluid problems where the continuity and momentum equations take place. Firstly, other types of simpler cases will be studied to understand concepts such as heat conduction or the terms of convection and diffusion. This last concept will be studied by developing a code of two different cases: diagonal flow and Smith-Hutton flow.

Once these simpler cases have been analyzed and studied, a special case will be reached for the study of the aforementioned equations at different Reynolds numbers, that is, for laminar and turbulent fluids.

Finally, the final budget of the study is analyzed, and the impact it has had on the environment, for example, the carbon monoxide emissions emitted when carrying out the project.

Este trabajo consiste en entender las ecuaciones de Navier Stokes mediante la programación de códigos, hechos en MatLab, que simulan problemas de fluidos donde las ecuaciones de continuidad y momento forman parte. Primeramente se estudiarán otro tipo de casos más sencillos para entender conceptos como la conducción de calor o los términos de convección y difusión. En este último concepto se estudiarán dos casos por separado: Fluido diagonal y el caso especial Smith-Hutton

Una vez se hayan analizado y estudiado estos casos más simples, se llegará a un caso especial para el estudio de las ecuaciones antes mencionadas a diferente número de Reynolds, es decir, para fluido laminar y turbulento.

En última instancia, se analiza el presupuesto final del estudio y la repercusión que ha tenido en el entorno, por ejemplo, las emisiones de monoxido de carbono emitidas al realizar el proyecto.

Contents

List of Figures

List of Tables

List of abbreviations

CFD	Computational Fluid Dynamics
FVM	Finite Volume Method
PDE	Partial Differential Equation
CR	Crank Nickolson Method
TDMA	Tridiagonal Matrix Algorithm
VC	Control Volume
CTTC	Centre Tecnològic de Transfèrencia de Calor
GS	Gauss-Seidel
CDS	Central difference scheme
UDS	Upwind difference scheme
EDS	Exponential difference scheme
HDS	Hybrid difference scheme
SUDS	Second-Order upwind linear extrapolation
QUICK	Quadratic upwind interpolation for convective kinematics
FSM	Fractional Step Method
HHT	Helmholtz-Hodge theorem

Chapter 1

Introduction

1.1 Aim

The project's aim is to develop Matlab code that can be used to solve Navier-Stokes equation-related issues in fluid dynamics, heat transfer, and mass transfer.

1.2 Scope

The study includes the next tasks:

- A previous analysis to understand the Navier-Stokes equations and CFD and the different methods to solve differential equations using numerical methods.
- Creation of a code to solve the two-dimensional domain's transient conduction.
- Resolution of convection-diffusion problems using numerical methods.
- Develop a code to solve the Navier-Stokes equations.
- Validation and verification of the codes used in this project.
- Apply the codes to a particular aeronautical problem.

The tasks that are out of the scope:

- An experiment lab of any sort.
- Development of a code to solve a three-dimensional domain.

1.3 Requirements

The requirements for this project are:

- The use of Matlab to address all of the project's research problems.
- The codes have to be implemented in an intelligible and efficient way at the end of the project.
- Numerical solutions of the Navier-Stokes equations are based on Newtonian fluid that is incompressible.
- Post-processing of code results is compulsory. The generation of graphs from numerical data will be accomplished using Matlab codes.

1.4 Justification

Although the fluid motion is an area of inquiry for humans, the development of mathematical models began at the end of the 19^{th} century, following the industrial revolution. Nevertheless, in 1687, Isaac Newton gave the first accurate explanation of the motion of a viscous fluid [1].

Over the course of several decades, Claude-Louis Navier and Sir George Stokes separately and gradually developed the Navier-Stokes equations [1]. They were based on Newton's second law of motion applied to a fluid flow, accounting for the effects of viscosity and pressure to characterize a viscous fluid flow.

The Navier-Stokes equations are helpful because they explain the physics of numerous phenomena relevant to science and engineering. By using these equations to model fluid behavior, researchers can gain insight into how fluids behave in complex situations and develop more efficient and effective solutions to real-world problems. In fact, they can simulate airflow around a wing, water movement in a conduit, weather, and ocean currents among others. Moreover, they are helpful for designing vehicles and aircraft, studying blood flow, or designing power plants [2].

However, the Navier-Stokes equations are complicated to solve due to their lack of a simple linear relationship between their terms. This indicates that conventional methods cannot be utilized to solve the equations and numerical methods must be used.

The accuracy of this numerical method depends on the mesh size used, making it exceedingly computationally expensive and time-consuming to carry out an accurate simulation until a few years ago. Nowadays Computational Fluid Dynamics, also known as CFD, has grown significantly in importance within the industry, enabling flow behavior simulation in a fraction of the time and a fraction of the cost of older computers.

A self-developed code will be made in this thesis in order to understand the mathematical and physical basis of fluid mechanics, despite the wide variety and accessibility of CFD software. Additionally, those concepts will be refined, and advanced programming skills are needed to create the thesis' Matlab techniques.

In conclusion, the decision to investigate this topic for the final degree project was driven by the significance and difficulty of the numerical solution of these equations within the aerospace area.

Chapter 2

State of the art

Leonhard Euler discovered the equations of motion for a fluid in the middle of the eighteenth century [3]. In essence, everything he did was apply Newton's second law and the concept of conservation of mass to a broad range of fluid's material components. For each instant time and area around each point, Euler considered the matter enclosed within a cuboid with infinitesimally small dx, dy, and dz boundaries. This carried him to differential form motion equations.

The forces that influence a fluid's material components are classified as follows: those that act at a distance, like gravity, and those that act by coming into contact with the surrounding components, like pressure. Although Euler took into account these forces, a more crucial one was missing in the equations.

Viscosity-related contact forces were not adequately modeled until far into the nineteenth century [1]. This was created by George Gabriel Stokes, Adhémar Barré de Saint-Venant, Siméon Poisson, Augustin Cauchy, and Claude Navier [3].

As a result of these studies, they were discovered, commonly called, "Navier-Stokes equations". These equations are used to represent the motion of a viscous and incompressible fluid in a closed and immovable container. The differential form can be written as follows [2]:

Continuity Equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0 \tag{2.1}$$

Momentum Equation:

$$\frac{\partial v}{\partial t} + v \cdot \nabla v = -\frac{1}{\rho}\nabla p + \nu \nabla^2 + f \tag{2.2}$$

where ρ is the fluid density, v is the velocity vector field, t is time, p is the pressure, ν is the kinematic viscosity, and f represents the external forces, for instance, gravity.

The rate of change in the fluid density at a place is equal to the negative divergence of the fluid velocity vector field multiplied by the density, according to the continuity equation 2.1, which expresses the conservation of mass in a fluid.

The momentum equation 2.2 expresses the conservation of momentum and states that the rate of change of the fluid velocity at a point is equal to the sum of the pressure gradient, the viscous forces due to the kinematic viscosity, and any external forces acting on the fluid.

As it has been mentioned in the section 1.4, the Navier-Stokes equations are nonlinear, partial differential equations that are notoriously difficult to solve analytically. As a result, approximate solutions are frequently obtained using numerical approaches. Finite difference, finite element, and spectral methods are only a few of the numerous numerical techniques that have been developed over time [2]. Many fluid dynamics issues, including laminar and turbulent flows, and compressible and incompressible flows, have been addressed using these techniques.

Also, researchers have made significant progress in developing algorithms for simulating turbulent flows, such as large eddy simulation (LES) and direct numerical simulation (DNS) [4].

On the other hand, mathematical theories have been developed to understand these equations better. In fact, the theory of blow-ups is one example of this [5]. It explores the issue of whether solutions of the Navier-Stokes equations can grow infinite in a finite amount of time. For instance, the equations have been used to study fluid mechanics in the context of quantum field theory and string theory.

In conclusion, even though there are still many unknowns to be resolved, the understanding of the Navier-Stokes equations has significantly advanced recently, and researchers are still working to improve the numerical methods, theoretical understanding, and applications for these equations.

Chapter 3

Theory background

It has been mentioned in chapter 2 that there are huge possibilities of numerical methods to find an approximate answer to the equations. The aim of this section is to introduce the different alternatives and choose which one is going to be used in this project.

3.1 Spatial discretization

A continuum space is divided into a finite number of discrete points or cells by the process of spatial discretization. When the continuous space needs to be expressed by a discrete set of values for computing purposes, it is frequently utilized in the numerical simulation of physical systems.

The division of the continuous space into a grid or mesh of discrete points is the process of spatial discretization as has been mentioned before. A value that represents the physical quantity being replicated, such as temperature, pressure, velocity, or concentration, is assigned to each point or cell in the grid.

The size of the grid and the distance between the point affect how accurate the simulation is. The accuracy of the simulation will rise with a finer grid and closer spacing between the points but at the expense of more processing complexity and more time of compilation. It can be divided into structured meshes and unstructured ones [6]. In the figure below, 3.1, they are shown.

The structured meshes mean each point of the mesh has the same number of neighbors. Moreover, nodal connectivity can be described as a constant dimension that alludes to having a clear structure and easy to locate the nodes inside the grid. The most important advantage of this structure is that the compilation time is shorter than other types of mesh because their coefficient matrix tends to be diagonally dominant.

However, this structure is difficult when the shapes of the objects become complex.

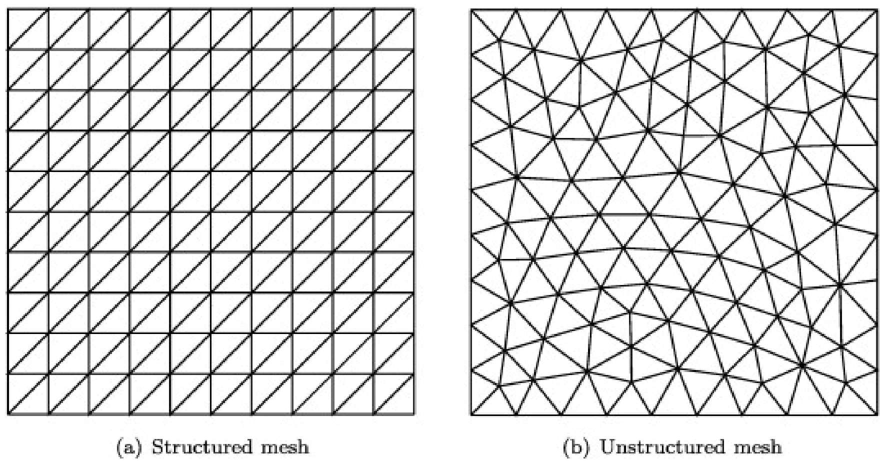


Figure 3.1: The two different types of the mesh structure. Extracted from [6]

On the other hand, the unstructured meshes are more complex, and the compilation time is longer. Nonetheless, if the shape of the object is complex this mesh is useful to obtain better results.

Nowadays, the hybrid structure is used in most of the studies because it has the advantages of both frameworks.

3.1.1 The Finite Volume Method (FVM)

Numerical simulations use a range of spatial discretization techniques. These are some examples, the finite difference method, the Finite Volume Method, the Finite Element Method, or the Spectral Method.

Nevertheless, the most popular for modeling fluid flow and transport processes is the FVM [7]. This method approximates solutions to partial differential equations governing physical systems.

To approximate the values of the physical quantities being solved for, such as velocity, pressure, temperature, or concentration, the FVM divides the physical system into several small volumes or cells. The approach then creates a set of conservation equations for each cell that links the physical quantity's rate of change to its flow across the cell's boundaries [7].

Using a flux formula, which is often derived from the governing equations and indicates the speed at which the physical quantity is moved over the border, the flux across each cell boundary is calculated. A combination of physical laws, such as conservation laws, and empirical observations, such as experimental data, is often used to develop the flux formula.

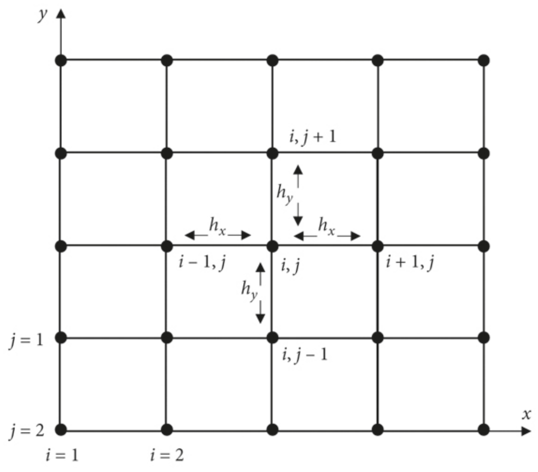


Figure 3.2: Example of the method mesh. Extracted from [7]

The FVM is a conservative approach, which means that it accurately conserves mass, momentum, energy, and other conserved quantities. Both steady-state and time-dependent problems can be solved with it, and it is effective at handling complex geometries and boundary conditions (boundaries will be explained in section 3.1.2).

One feature of the method is that it naturally discretized the physical system into a collection of discrete volumes, which makes it simple to implement and parallelize. Additionally, it has high stability and convergence characteristics (to know more about convergence methods see section 3.3), allowing it to deliver accurate and dependable solutions even for complex situations.

However, it must be taken into account the choice of cell size and geometry, as well s the numerical algorithms utilized to compute fluxes because the method has certain limitations. Also, it might be computationally expensive, especially for issues with demanding resolutions or board domains.

Overall, it is a crucial tool for many scientific applications due to its precision in quantity conservation and capacity to manage complex geometries.

3.1.2 Boundaries

To solve the partial differential equation or a system of them a boundary condition must be defined to obtain a unique solution since there are typically infinitely many possible answers that satisfy these equations within the domain. By specifying the behavior of the solution at the boundary, one can determine which of the solutions is physically possible. The most common boundary conditions types are as follows [8] :

- **Dirichlet boundary condition**: These specify the value of the solution on the boundary.

$$\phi = c \tag{3.1}$$

- **Neumann boundary condition**: These specify the derivative of the solution normal to the boundary.

$$\frac{\partial \phi}{\partial n} = k \tag{3.2}$$

where c and k are constants and have a determinate value. This value changes if the problem characteristic changes.

3.2 Time discretization

By describing a continuous-time system or signal with a collection of discrete time points or samples, a process known as time discretization is used to approximate it.

The system's computing needs as well as the precision of the time discretization can both be significantly influenced by the sampling technique chosen. Higher-order approaches typically produce more accurate findings, but they also call for greater computer power. The most ordinarily used in numerical methods are:

- **Explicit scheme**: It is an approximation of first order, meaning that is less accurate than other schemes of second or third order. The value of the property is determined at the subsequent time step using its current function value. Although it reduces the computational time of a program it is conditionally stable. This means that if a big-time step is set, some stability issues may arise and it would not find a unique solution to the problem. In this type of discretization, it has to be a fixed precise time step value.

- **Implicit scheme**: It is an approximation of first order as the explicit scheme. The property value is determined at the following time step using its function value at the future time. The benefit of utilising that method is that is unconditionally stable (it is the only advantage regarding the scheme before), allowing for the usage of a big-time step value. The only disadvantage is that it requires more compilation time than the other one and has more complexity.

- **Crank-Nicolson scheme (CR)**: It is an approximation of second order, meaning it is more accurate than the other ones. This discretization uses its function value at the current and future times, the property value is determined at the following time step. Also, it is unconditionally stable.

3.3 Solvers

A solver is a software program or method that is used in computational engineering to solve equation systems. The common type of solver used in computational fluid dynamics is the iterative one. This solver works through an initial guess until the necessary level of accuracy is reached before solving the equation. The Gauss-Seidel method or the Line-by-line are a few examples of this type.

3.3.1 Gauss-Seidel method

A system of linear equations can be solved iteratively using the Gauss-Seidel method. It bears the name of two German mathematicians who separately created the technique in the early 19^{th} century: Carl Friedrich Gauss and Philipp Ludwig von Seidel.

Using the most recent values of the other components, the Gauss-Seidel technique updates the solution vector one component at a time. Using the most recent estimations of the other components, the approach solves for each element in the system's associated equation to determine its new value at each iteration. In section 3.4 it can be seen the scheme of resolution of this method.

3.3.2 Line-by-line method (TDMA modification)

A specific form of the Tridiagonal Matrix Algorithm (TDMA), the Line by Line method, sometimes referred to as the Thomas Algorithm, can be used to resolve systems of linear equations with a banded matrix. A matrix that has non-zero entries confined to a band around the main diagonal is said to be banded.

The Line by Line method is an effective approach for solving banded systems of linear equations, much like the TDMA algorithm. On the other hand, this method is more adaptable than the TDMA algorithm since it can accommodate matrices with different bandwidths (and it can be used in 2D).

Compared to the Gauss-Seidel method, this one offers faster convergence and fewer iterations, making this method more efficient talking about the compilation time. However, since only lines can be paralleled, it is more challenging to parallelize than the other method.

3.4 Algorithm in use

This thesis is going to be solved different types of cases of fluid dynamics so it has to be implemented a basic algorithm for all the cases. The algorithm is explained below:

1. **Data entry**: There are two types of data, numerical and physical. The numerical ones are the parameters for the solution accuracy, for instance, the number of nodes or the relaxation coefficient. On the other hand, the physical data is the data from the special case, for example, boundary conditions or material properties.

2. **Previous calculations**: To solve the equations first of all there are some calculations that have to be done. The calculation of the position vector or the control volumes is an example of this stage of the code.

3. **Initial map**: The initial map is to assign one value to the initial variables, most of the time these variables are the temperature, the pressure, the velocity...

4. **Discretization coefficients**: The main computation in this stage is to know the coefficients of the equations after they have been discretized.

5. **Solver**: In this step the solver mentioned in section 3.3 must be used. In the case of Gauss-Seidel, it has to be verified that the convergence of the initial value and the value calculated in this part is correct. If not the initial map has to be modified with the new calculations and done from step 3 to this one again.

6. **Time step new**: The solver only calculates a one-time step at a given time. If the problem that is solved is transitional, meaning it changes over time, the program must have this phase. This step consists to ask the program if the time step that has been calculated was the last one. If the answer is yes the solver has finished the problem. On the other hand, if the answer is no it has to do from step 3 to step 6 again with the new time step.

Chapter 4

Heat conduction case

The goal of this chapter is to create a Matlab program that can use the FVM (see section 3.1.1) on a structured orthogonal mesh to solve a transient, 2D heat conduction problem. First of all, an overview of the heat conduction mechanism will be explained. Later, the discretization of the transient heat conduction equation for solid materials will be detailed. It is worth mentioning that the problem will involve four materials and it will change the boundary conditions. Finally, the verification of the code will be tested by numerically solving a straightforward heat conduction problem case.

4.1 Heat conduction theory

When solids, liquids, or gases have no relative motion between their molecules, heat conduction - an energy transmission mechanism - occurs. Temperature differences, commonly called the gradient of temperatures, inside the material cause this occurrence, which causes an energy transfer to take place in order to maintain equilibrium. This can be written as follows:

$$\frac{\partial}{\partial t}\int_{vc} u\rho dV = \dot{\mathbf{Q}} \Rightarrow \frac{\partial \rho u}{\partial t} = \dot{\mathbf{q}} + \dot{q}_v \tag{4.1}$$

where the first term is the variation of internal energy during a period of time, the second term is the energy that transfers from one space to another and, at least, the third term is due to internal sources of heat due to Joule's effect.

Moreover, the heat through a material can be expressed with Fourier's Law. According to this, the rate of heat transfer is inversely related to its thermal conductivity, λ, and directly proportional to the temperature differential across the material. In mathematical terms, it is expressed as:

$$\dot{\mathbf{q}} = -\lambda \nabla T \tag{4.2}$$

4.2 Discretization of the equations

The aim of this section is to transform the equations of heat conduction from the domain of spatial and time to the algebraic domain to obtain a unique solution.

4.2.1 Spatial domain discretization

In section 3.1 it has been seen that the best option for simple problems was the structured mesh. For this reason, the spatial domain will be discretized into quadrilateral equal finite volumes. Each volume will have a central node where the properties of the finite element will be assigned. To have a better understanding a scheme is shown:

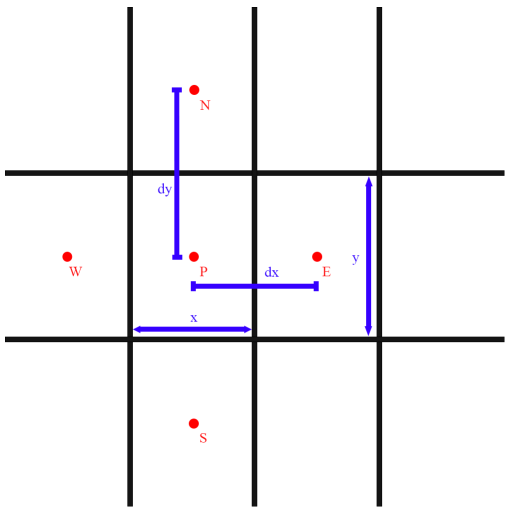


Figure 4.1: Scheme of the discretized nodes

where P is the central node and the neighbors are called W (west), E (east), S (south), and N(north). Also, A_i is the area of each surface where x, y are the height and the length of this surface. Finally, the distance d_x and d_y are the distance between nodes.

The heat conduction equation, 4.1 can be integrated over a control volume of the node P.

$$\frac{\partial}{\partial t}\int_{vc} u\rho \, dV_c = \int_{vc} \dot{\mathbf{q}} \, dV_c + \int_{vc} \dot{q}_v dV_c \tag{4.3}$$

The first term of the equation 4.3 can be integrated directly because the density does not vary over the volume of control and neither the internal energy. Once the control volume is integrated, it can be written as,

$$V_c = x \cdot y \cdot 1 \tag{4.4}$$

This simplification can be made due to the bi-dimensional discretization of all the problem cases written in this report.

Secondly, the heat flux of the equation varies along it. Because of that this integral is more complex than the other one. Figure 4.2 shows how the heat moves in the volume of control of node P to the other closest nodes.

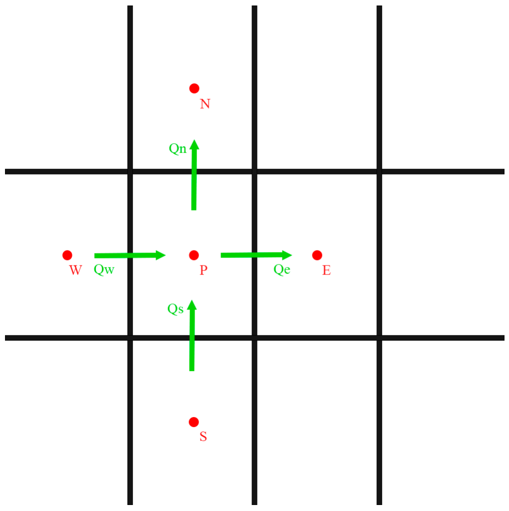


Figure 4.2: Heat fluxes inside the spatial discretization

The flux heat can be written as:

$$\int_{vc} \dot{\mathbf{q}}\, dV_c = \dot{Q}_x - \dot{Q}_{x+dx} + \dot{Q}_y - \dot{Q}_{y+dy} + \dot{Q}_z - \dot{Q}_{z+dz} = \dot{Q}_x - (Q_x + \frac{\partial Q_x}{\partial x}dx) + \dot{Q}_y - (Q_y + \frac{\partial Q_y}{\partial y}dy) + + \dot{Q}_z - (Q_z + \frac{\partial Q_z}{\partial z}dz) \tag{4.5}$$

A rearrangement of the terms can be done in the equation above. Once the expression is reduced Fourier's Law, 4.2, can be applied.

$$-\frac{\partial Q_x}{\partial x}dx - \frac{\partial Q_y}{\partial y}dy - \frac{\partial Q_z}{\partial z}dz = -\frac{\partial}{\partial x}\left(-\lambda\frac{\partial T}{\partial x}dydz\right)dx - \frac{\partial}{\partial y}\left(-\lambda\frac{\partial T}{\partial y}dxdz\right)dy - \frac{\partial}{\partial z}\left(-\lambda\frac{\partial T}{\partial z}dxdy\right)dy$$

The aim is to discretize equation 4.1, so in terms of the neighbor nodes it can be expressed as:

$$\lambda_w \frac{T_P - T_W}{d_{PW}} S_w - \lambda_e \frac{T_E - T_P}{d_{PE}} S_e + \lambda_s \frac{T_P - T_S}{d_{PS}} S_s - \lambda_n \frac{T_N - T_P}{d_{PN}} S_n \tag{4.6}$$

where S_i is the perpendicular area of the volume that produces heat transfer. This surface can be written as $S_w = S_e = y \cdot 1$ and $S_n = S_s = x \cdot 1$. Notice that the thickness of VC is 1 because it is a bi-dimensional problem (as it was mentioned before in this section).

On the other hand, d_{PI} is the distance between the node P and each of their neighbor. This distance can be written as $d_{PE} = d_{PE} = d_x = x$ and $d_{PS} = d_{PN} = d_y = y$. It can be said that $d_x = x$ and $d_y = y$ because the mesh has all the volume of control equal in size and shape.

To sum up, in the equation 4.6, it appears the term λ_i which is the thermal conductivity of each surface that separates the node P from their neighbors. This thermal can be written using the harmonic mean, meaning:

$$\lambda_i = \frac{d_{PI}}{\frac{d_{Pi}}{\lambda_P} + \frac{d_{Ii}}{\lambda_I}}$$

where λ_I and λ_P are the thermal conductivity of the neighbor and the node P, d_{Pi} and d_{Ii} are the distance between the node and the control surface. Below can be found a scheme to have this concept more clear.

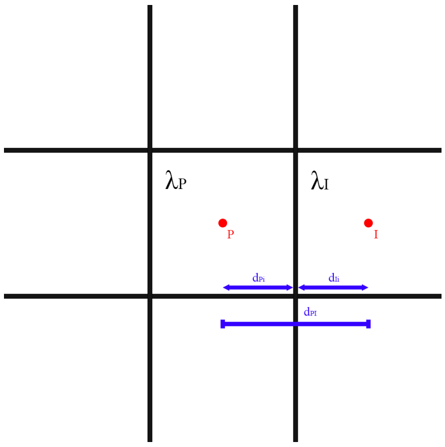


Figure 4.3: Harmonic mean scheme

Finally, the last term of the equation 4.1 has to be discretized. This is simple as the first one.

$$\int_{vc} \dot{q}_v \; dV_c = \dot{q}_v \; V_c$$

where V_c is the same volume as the first term of this equation.

The final discretized equation is:

$$\frac{\partial(\rho u)}{\partial t} x \cdot y = \lambda_w \frac{T_P - T_W}{x} \cdot y - \lambda_e \frac{T_E - T_P}{x} \cdot y + \lambda_s \frac{T_P - T_S}{y} \cdot x - \frac{T_N - T_P}{y} \cdot x + \dot{q}_v \cdot x \cdot y \qquad (4.7)$$

4.2.2 Time domain discretization

Transient impacts are considered, as was previously stated. As a result of that, equation 4.1 also needs to be temporally discretized. It has to be integrated into a general VC for this purpose over a finite period. Therefore,

$$\int_{t^n}^{t^{n+1}} \frac{\partial(\rho_P u_P)}{\partial t} x \cdot y \; dt = \int_{t^n}^{t^{n+1}} \dot{Q}_P \; dt \qquad (4.8)$$

In the accumulated energy term an assumption can be made. The hypothesis is that the parameters of density and distance remain constant over time (semi-perfect solids) but the internal energy can vary. In addition, the internal energy, u, is a state function, which implies that its variation only depends on the beginning and final value. Therefore the integral solution is:

$$\int_{t^n}^{t^{n+1}} \frac{\partial(\rho_P u_P)}{\partial t} x \cdot y \, dt = \rho_P \cdot x \cdot y (u_P^{n+1} - u_P^n) \tag{4.9}$$

The internal energy of a semi-perfect solid can be expressed as $du = c_p \, dT$. For this reason equation 4.9 is:

$$\int_{t^n}^{t^{n+1}} \frac{\partial(\rho_P u_P)}{\partial t} x \cdot y \, dt = \rho_P \cdot x \cdot y \cdot \bar{C}_{P_p} (T_p^{n+1} - T_P^n) \tag{4.10}$$

where $\bar{C}_{P_p}$ is the material-specific heat capacity in the node P.

The second term, is the flux heat of the node P, in terms of time discretization it states, using the trapezium method:

$$\int_{t^n}^{t^{n+1}} \dot{Q}_P \, dt = \left[\beta \sum \dot{Q}_p^{n+1} + (1-\beta) \sum \dot{Q}_P^n\right] \Delta t \tag{4.11}$$

The time discretization strategy that will be utilized has an impact on the term β. In this case, the time discretization chosen, according to section 3.2, is the implicit scheme meaning $\beta = 1$.

Therefore the result of the equation 4.8 is:

$$\rho_P \cdot x \cdot y \cdot \bar{C}_{P_P} (T_P^{n+1} - T_P^n) = \sum \dot{Q}_p^{n+1} \Delta t \Rightarrow \frac{\rho_P \cdot x \cdot y \cdot \bar{C}_{P_P}}{\Delta t} (T_P^{n+1} - T_P^n) = \sum \dot{Q}_p^{n+1} \tag{4.12}$$

4.2.3 Coefficient calculations

To make the code easier for a numerical resolution the use of coefficients has to be compulsory. These coefficients are all the values known that multiply the temperature. First of all, equation 4.12 must be written for a general node using equation 4.6.

$$\rho_P \cdot x \cdot y \cdot \bar{C}_{P_P} (T_P^{n+1} - T_P^n) = \lambda_w^{n+1} \frac{T_P^{n+1} - T_W^{n+1}}{x} y - \lambda_e^{n+1} \frac{T_E^{n+1} - T_P^{n+1}}{x} y + \lambda_s^{n+1} \frac{T_P^{n+1} - T_S^{n+1}}{y} x - \lambda_n^{n+1} \frac{T_N^{n+1} - T_P^{n+1}}{y} x + \dot{q}_v^{n+1} \cdot x \cdot y \tag{4.13}$$

Now, the equation 4.13 can be expressed with the coefficients.

$$a_P T_P^{n+1} = a_W T_W^{n+1} + a_E T_E^{n+1} + a_N T_N^{n+1} + a_S T_S^{n+1} + b_P \tag{4.14}$$

where,

$$
\begin{aligned}
a_S &= \frac{\lambda_s^{n+1} \cdot y}{x} \\
a_N &= \frac{\lambda_n^{n+1} \cdot y}{x} \\
a_W &= \frac{\lambda_w^{n+1} \cdot x}{y} \\
a_E &= \frac{\lambda_e^{n+1} \cdot x}{y} \\
a_P &= a_S + a_N + a_E + a_W + \frac{\rho_P \bar{C}_{P_P} \cdot x \cdot y}{\Delta t} \\
b_P &= \dot{q}_v^{n+1} \cdot x \cdot y + \frac{\rho_P \bar{C}_{P_P} \cdot x \cdot y \cdot T_P^n}{\Delta t}
\end{aligned}
$$

From a computational point of view, this equation can be seen as a matrix with all the nodes inside them. Each of the nodes inside this matrix can be related to their neighbors with their temperatures. The equation 4.13 can be written as,

$$
a_P(i,j)T^{n+1}(i,j) = a_E(i,j)T^{n+1}(i+1,j)+a_W(i,j)T^{n+1}(i-1,j)+a_N(i,j)T^{n+1}(i,j+1)+a_S(i,j)T^{n+1}(i,j-1)+b_P(i,j) \tag{4.15}
$$

4.2.4 Boundary conditions

To find a unique solution the problem must have boundary conditions. In section 3.1.2 it has been shown the two different types that the PDE can have. In this particular case,

Dirichlet boundary conditions

In heat conduction, this type of boundary condition refers to nodes whose temperature is constant during the problem, for instance, the temperature of the wall. The heat balance of a boundary node in figure 4.4.

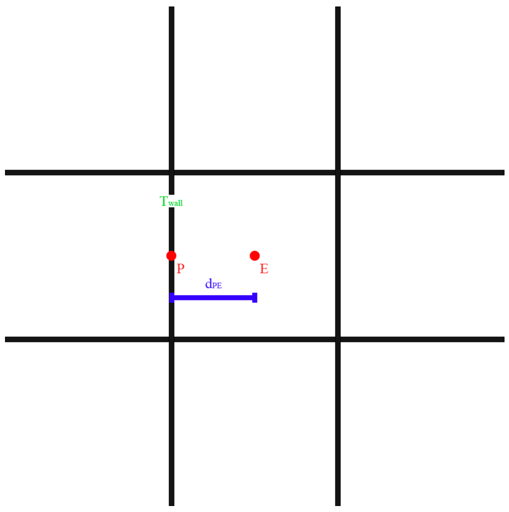


Figure 4.4: Heat balance with a temperature of the wall

These kinds of nodes are particular because they do not have control of volume. The flux heat can be expressed as,

$$Q_{cond} = 0 \Rightarrow -\left(\lambda \frac{\partial T}{\partial x}\right) = -\lambda \frac{T_{wall}^{n+1} - T_E^{n+1}}{d_{PE}} S \tag{4.16}$$

To assure that the temperature of these nodes is always the temperature of the wall the coefficients of equation 4.13 will be:

$$\begin{aligned} a_S &= 0 \\ a_N &= 0 \\ a_W &= 0 \\ a_E &= 0 \\ a_P &= 1 \\ b_p &= T_{wall}^{n+1} \end{aligned}$$

Neumann boundary condition

In this type of boundary, the imposed condition is the value of the derivative, in dynamic fluids, it is the heat flux ($\dot{q}$). For this type of node, equilibrium of the flux heat can be done but taking into account the heat flux from the other node.

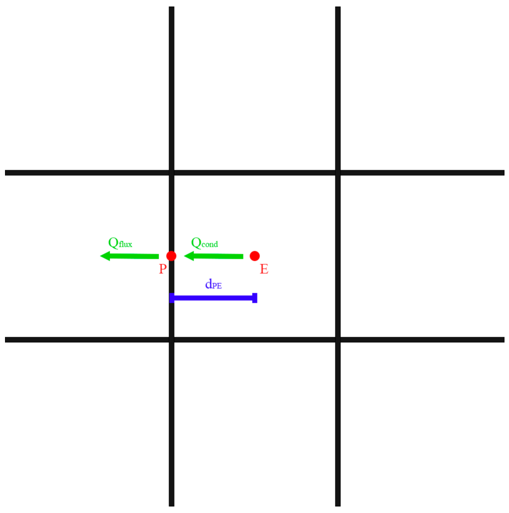


Figure 4.5: Heat balance between two nodes and with flux heat

$$\dot{Q}_{cond} - \dot{Q}_{flux} = 0 \Rightarrow -\lambda \frac{T_P^{n+1} - T_E^{n+1}}{d_{PE}} S - \dot{q}_{flux} S = 0 \tag{4.17}$$

As the other boundary condition, the coefficients can be described as,

$$
\begin{aligned}
a_S &= 0 \\
a_N &= 0 \\
a_W &= 0 \\
a_E &= \frac{\lambda}{d_{PE}} \\
a_P &= a_S + a_N + a_E + a_W \\
b_P &= \dot{q}_{flux}
\end{aligned}
$$

The last case of this type of boundary condition is when there is a convective exchange with the environment. Convective heat is heat that moves from a solid surface to a flowing fluid, such as air or water. The movement of the fluid particles, which transmit energy from the surface to other portions of the liquid, causes heat transfer. An example of this convective heat can be seen in figure 4.6.

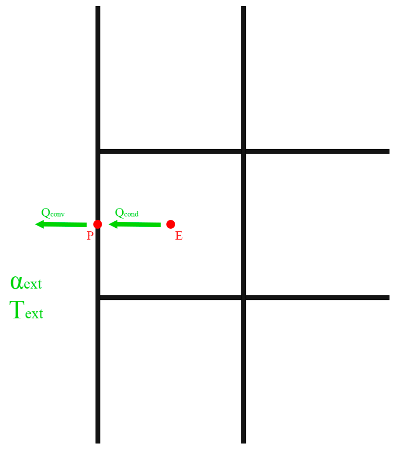


Figure 4.6: A convective and conductive heat balance

The heat equation can be written as,

$$-\lambda \frac{T_P^{n+1} - T_E^{n+1}}{d_{PE}} S - \alpha_{ext}(T_P^{n+1} - T_{ext})S = 0 \tag{4.18}$$

where the discretized coefficients are,

$$a_S = 0$$
$$a_N = 0$$
$$a_W = 0$$
$$a_E = \frac{\lambda}{d_{PE}}$$
$$a_P = a_S + a_N + a_W + a_E + \alpha_{ext}$$
$$b_p = \alpha_{ext} T_{ext}$$

4.2.5 Algorithm

The following calculating algorithm describes how the implemented Matlab code is organized. It follows the general structure that has been seen in section 3.4.

1. Data entry (user input). Physical data such as materials, geometry, boundary condition... Numerical data, for instance, the number of nodes, convergence criteria, etc.
2. Previous calculations. Spatial and time discretization using vectors.
3. Initial map ($t^n = 0$). $T^n[i][j] = T_0(x, y)$
4. Computation of the next time step, $t^{n+1} = t^n + \Delta t$
   (a) Estimation of $T^{n+1*}[i][j] = T^n[i][j]$
   (b) Computation of the discretization coefficients, $\lambda[i][j], a_E[i][j], b_P[i][j]$
   (c) Use line by line method (section 3.3). Using the general discretized equation, 4.13, mentioned before.
   (d) Max $|T^{n+1[i][j]} - T^{n+1*}[i][j]| < \delta$ where δ is the convergence criteria.
      - If the answer is **Yes** go to step 5.
      - If the answer is **No** go to step (a). And $T^{n+1*}[i][j] = T^{n+1}[i][j]$
5. New time step?
   - If the answer is **Yes** go to step 4 and $T^{n+1}[i][j] = T^n[i][j]$.
   - If the answer is **No** go to step 6.
6. Print the results and show the conclusions.

4.3 Heat conduction in 1D

The objective of this section is to understand the results of a simple heat conduction problem of a wall. Once this problem has been explained, the report will be focusing on more complex problems. In this chapter, it is going to be created a Matlab program that uses TDMA solver 3.3 to obtain the temperatures along a thin wall with thickness e.

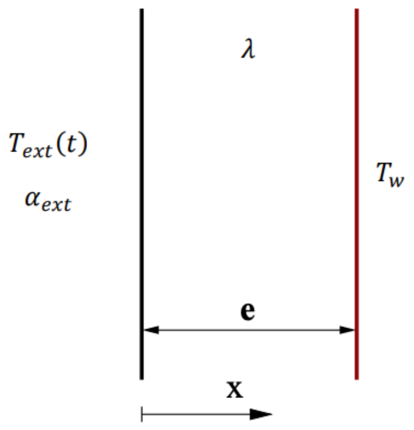


Figure 4.7: 1D conduction problem scheme

The properties of the case can be found in table 4.1

e(m)	$\rho(kg/m^3)$	$c_p(J/KgK)$	$\lambda(W/mK)$	α	T_0(ºC)	A_0(ºC)	A_1(ºC)	A_2(ºC)	T_w
1	2400	900	220	8.7	21	21	4.3	7.5	21

Table 4.1: Properties of the problem

The temperature of the wall can be expressed as,

$$T_{ext} = A_0 + A_1 sin(\omega_1 t) + A_2 sin(\omega_2 t)$$

where the time is calculated with the following expressions,

$$\omega_1 = \frac{2\pi}{24x3600}$$
$$\omega_2 = \frac{2\pi}{365x24x3600}$$

The right part is an isotherm, which means the temperature there is constant, $T = T_w$. The initial temperature of the rest of the wall can be considered as $T = T_0$. Moreover, this a simple exercise which means some assumptions has to be done before starting the problem.

- It will be supposed that the heat will be unidirectional, making the problem 1D.
- The property λ will remain constant during the exercise. The material will not change.

4.3.1 Spatial and time discretization

For this problem the spatial discretization is simple. It will have nodes along the direction x.

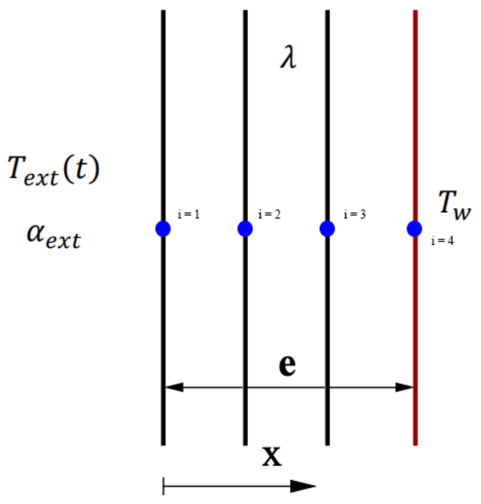
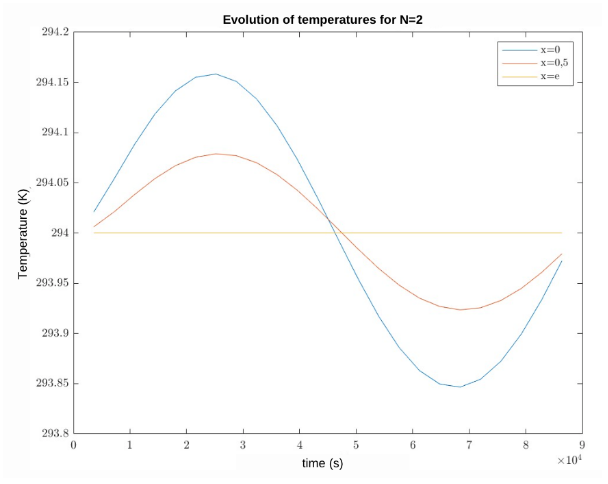
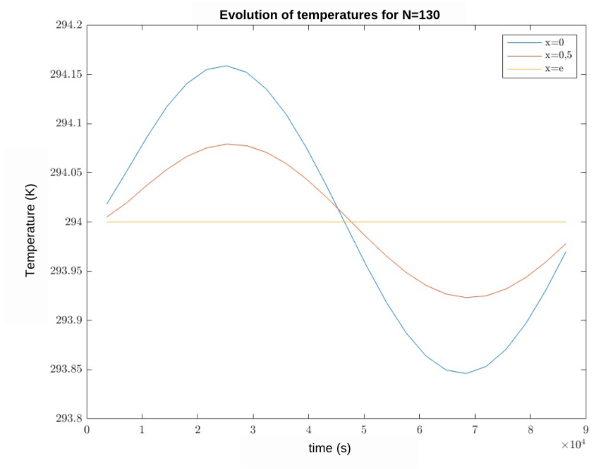


Figure 4.8: 1D conduction problem spatial discretization

At this point, it has to be chosen the elements of N. In this case, the number of elements will be 2 because as is a simple case the results of the problem will not vary. In the next figures, it can be analyzed the difference between one and the other,



(a) Results for N = 2



(b) Results for N = 130

Figure 4.9: Results of the problem using different numbers of elements to be discretized

On the other hand, to solve this exercise the Δt chosen is $6 \cdot 3600$ s. As this case is calculated with TDMA, there is no relaxation coefficient.

4.3.2 Boundary conditions

In this problem, the boundary conditions are found in the right and left walls. The discretization of this wall coefficients are written,

On the left wall:

$$a_e = \frac{\lambda}{d_{PE}}$$

$$a_p = a_e + \alpha_{ext}$$

$$a_w = 0$$

$$b_p = \alpha_{ext} \cdot T_{ext}$$

On the right wall:

$$a_e = 0$$

$$a_p = T_p$$

$$a_w = 0$$

$$b_p = 1;$$

Notice that the right wall has to be always, at any time, isotherm that is because it has to be imposed a boundary condition.

4.3.3 The code

According to the algorithm in section 4.2.5, the Matlab code created to tackle the transient heat conduction problem is divided into many procedures and subroutines. The temperature map at the final calculation time is the code's output.

The problem variables are first written and this includes geometric and numerical parameters for the convergence of the problem. The developed code is shown in another document called "Annex" with all the thesis codes.

4.3.4 Results

This part of the report, it is exposed the results of this 1D conduction case. First is going to be shown the energy exchange in this simple case. Then, the temperature evolution is shown in this same section.

Energy exchange

The exchange of energy will be analyzed in two points of the domain. First of all, it is going to be seen in x=0. In this part, there is convection from the exterior and conduction inside the wall as mentioned in the introduction of the problem.

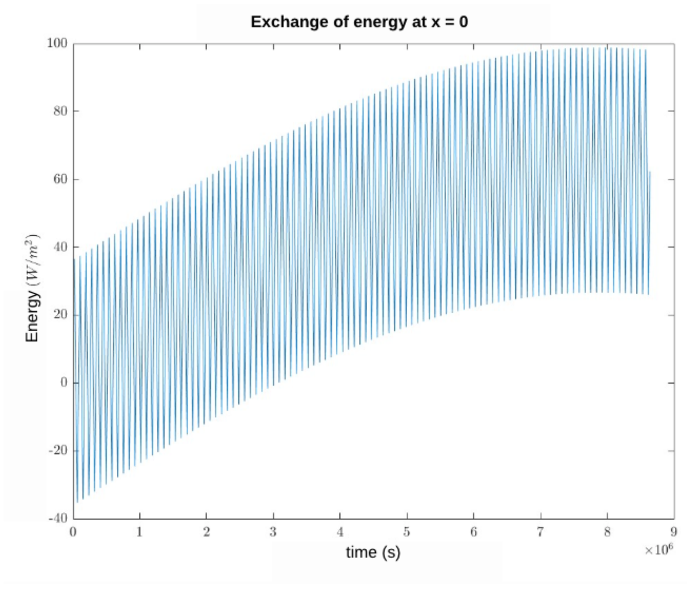


Figure 4.10: Energy exchange at x = 0 m

As can be observed the transference of heat increases as time goes by. This exchange of energy is always from the outdoors to the wall because this term is always positively looking at this equation,

$$q_{convection} = q_{conduction} = \alpha_{ext} \cdot (T_{ext} - T_{wall})$$

Second of all, the exchange of energy at the end of the wall, which means, in x = e.

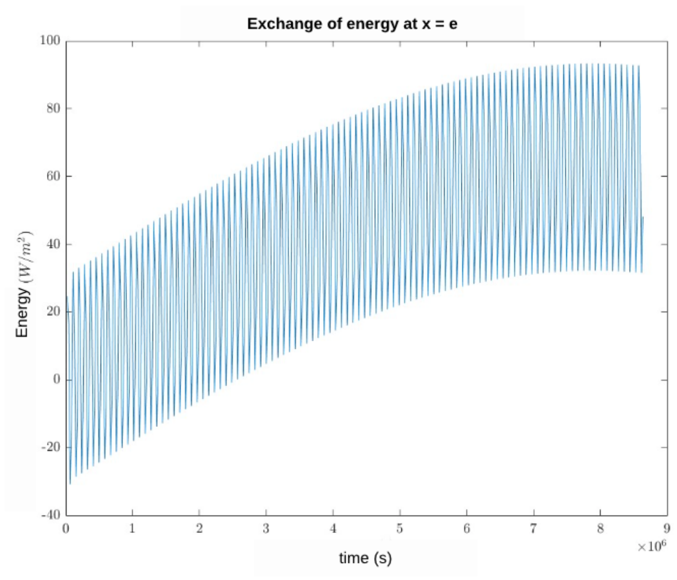


Figure 4.11: Energy exchange at x = e m

To obtain this heat exchange it has to be used the Fourier law 4.2 for this case.

$$q_{conduction} = -\lambda \frac{T_n(N+2) - T_n(N+1)}{x(N+2) - x(N+1)} = -\lambda \frac{T_n(4) - T_n(3)}{x(4) - x(3)}$$

The exchange of energy is positive, as the other one, but in less quantity this is due to the temperature of the exterior being higher than the temperature inside the wall.

Temperatures evolution

Below is the evolution of temperatures against the total time for different coordinates.

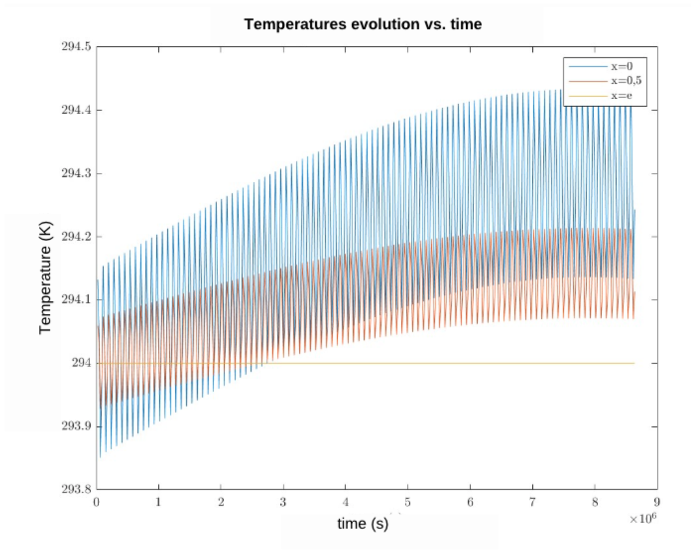


Figure 4.12: Evolution of the temperatures

In the figure above,4.12, is the temperature evolution during time. It can be noticed that the highest temperature is in $x = 0$, the value of this is higher than $294.4K$.

The temperatures at $x = 0$ and $x = e$ followed the same tendency as the exchange of energy. It can be detected that as the energy transferred is higher in the initial wall the temperature evolution is much higher than in the other part, as this part of the wall is an isotherm.

4.3.5 Verification of the case

As has been explained in the introduction of this section the verification of this case is to do a balance of energy. The equation followed was,

$$q_{accumulated} + q_{x=e} + q_{x=0} = 0$$

Using the code before explaining can be obtained in the next graphic to confirm this equation that has to be checked.

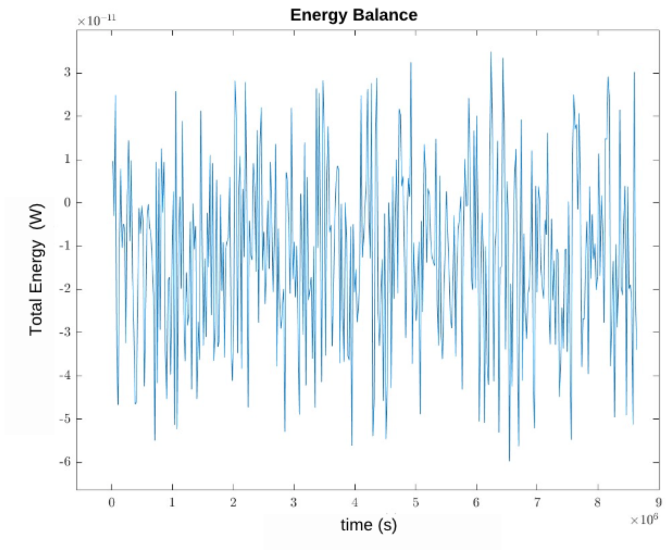


Figure 4.13: Balance of the total energy to verify the case

The figure demonstrated that the total energy balance of the problem is 0 which means the problem has been successfully verified.

4.4 Four material conduction problem

This section's goal is to create a MatLab program that uses the FVM (section 3.1.1) to solve a 2D transient heat conduction problem with four different materials with different boundary conditions. Once the code is done it will be verified with the solutions given by the CTTC (Centre Tecnològic de Transferència de calor).

4.4.1 Description of the problem

The scenario involves a region made up of four materials with different physical properties that are subjected to a range of boundary conditions. The general structure of the problem is in the next figure.

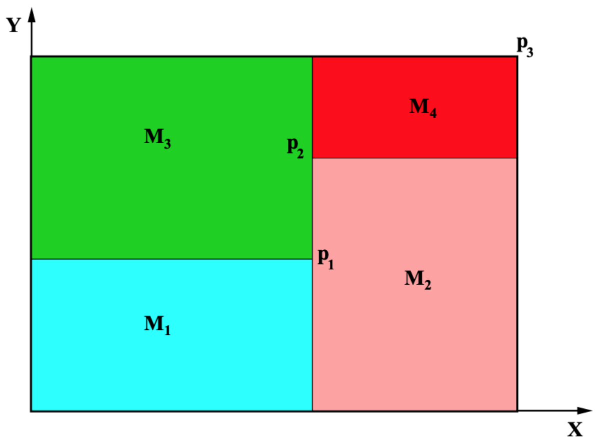


Figure 4.14: four materials problem given by the CTTC

A very long rod is composed of four different materials (M1 to M4), represented with different colors in the figure below. All the lines are parallel to the coordinate axis. the coordinate of the points p_1 to p_3 is given in table 4.2. The properties are in table 4.3. Each of the four sides of the rod interacts with the surrounding in a different manner, as described in table 4.4. The initial temperature field is $T = 8.00C$

	x[m]	**y[m]**
p_1	0.50	0.40
p_2	0.50	0.70
p_3	1.10	0.80

Table 4.2: Problem coordinates

	$\rho[kg/m^3]$	$c_p[J/kgK]$	$k[W/mK]$
M_1	1500	750	170
M_2	1600	770	140
M_3	1900	810	200
M_4	2500	930	140

Table 4.3: Physical properties

Cavity wall	**Boundary condition**
Bottom	Isotherm at $T = 23$ ºC
Top	Uniform $Q_{flow} = 60W/m$ length
Left	In contact with a fluid a $T_g = 33$ ºC and heat transfer coefficient $\alpha_g = 9W/m^2K$
Right	Uniform temperature $T = 8 + 0.005t$ ºC (where t is the time in seconds)

Table 4.4: Boundary conditions

4.4.2 Spatial discretization

The problem requires the definition of a geometric mesh. This mesh is going to be structured with quadrilateral equal finite volumes as it was mentioned in the section before. Moreover, it is going to be nodes in each finite volume and also in the material's corners and borders. The figure below it is shown the discretization.

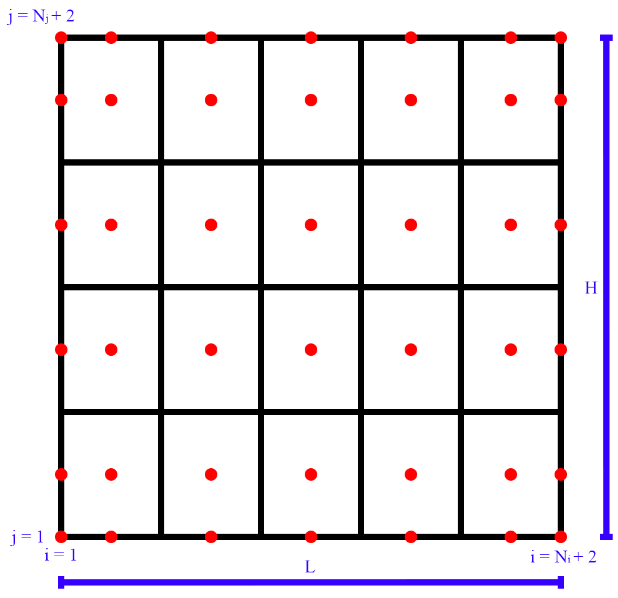


Figure 4.15: Spatial discretization of the problem

The elements density, which is defined as an input variable in the code, is the parameter that controls the mesh size. It is defined N_i and N_j as the total number of elements inside the mesh in the x direction and in the y direction respectively. The size of each volume can be defined as,

$$\Delta x = L/N_i$$

$$\Delta y = H/N_j$$

$$S = \Delta x \cdot \Delta y$$

The following equations can be used to relate the number of elements in the problem to the number of nodes in the problem because each element has a center node and by adding those on the border the equations finally state:

$$i = N_i + 2$$
$$j = N_j + 2$$

It is necessary to specify criteria to assign each element its properties, though, because the domain is made up of four different materials, each of which has its own thermophysical properties. The blocking-off procedure, which entails giving the element the characteristics of its internal node, is applied as the heat transfer is produced in a solid element.

On the other hand, the nodal density of the mesh has a significant impact on how precise that approach is. All control volumes exactly fit the geometry in the limit situation where the nodal density goes to infinity.

4.4.3 Boundary condition

Using the data in table 4.4 the boundary conditions of the problems can be established. The discretization of the wall coefficients in this case is shown on the next page.

In the left boundary:

$$a_E = 2\frac{\lambda_e}{\Delta x}$$
$$a_P = a_E + \alpha_g$$
$$b_P = \alpha_g T_g$$

In the right boundary:

$$a_P = 1$$
$$b_P = 8 + 0.005 \cdot t$$

In the bottom boundary:

$$a_P = 1$$
$$b_P = T_{bottom}$$

In the top boundary:

$$a_S = 2\frac{\lambda_s}{\Delta y}$$
$$a_P = a_S$$
$$b_P = \dot{q}_{flow}$$

4.4.4 The code

According to the algorithm in section 4.2.5, the Matlab code created to tackle the transient heat conduction problem is divided into many procedures and functions. The temperature map at the final calculation time is the code's output and is displayed as a matrix in a text file.

The problem variables are first written and this includes geometric parameters and numerical parameters for the convergence of the problem. The declaration of the used calculation functions follows. Thirdly, the primary part of the program, in which the calculating functions involved. The results are obtained in section 4.4.5 and the developed code is shown in another document called "Annex" with all this thesis's codes.

4.4.5 Results

In this section, the temperature map for $t = 10000s$ is shown. In figure 4.16 it can be seen the different temperatures among the scenario. As can be analyzed materials M2 and M4 have more high temperatures at the end of this time. This can be because of the different boundary conditions of each wall and the properties of each material (if it has less conductivity and more specific heat capacity it reaches faster at a higher temperature).

The numerical parameters for this case were:

- $N_x = 50$
- $N_y = 50$
- $\delta = 10^{-6}$
- $\Delta t = 1\ s$

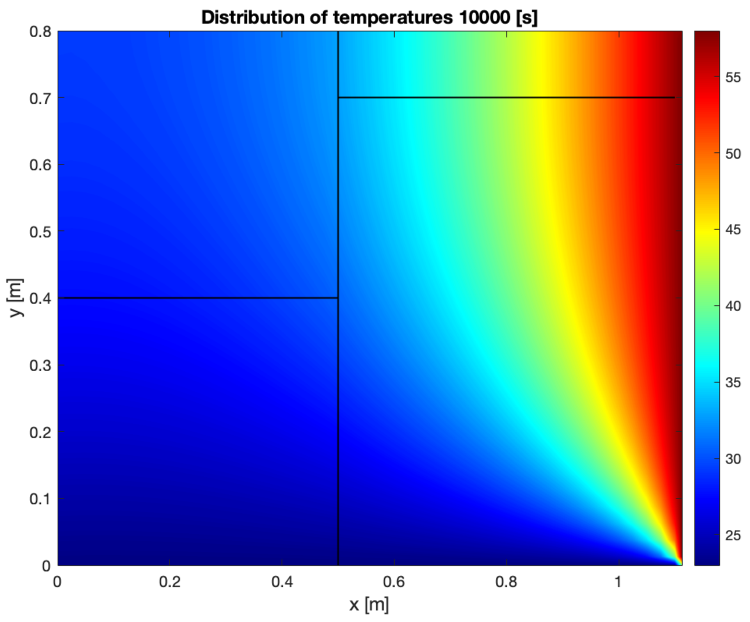


Figure 4.16: Temperatures distribution case 1 at t=10000s

As can be seen in figure 4.16 the temperature distribution is quite particular because of the material properties. On the other hand, to explain better the conduction of this wall with different materials the results for time $t = 500s$ are shown in figure 4.17.

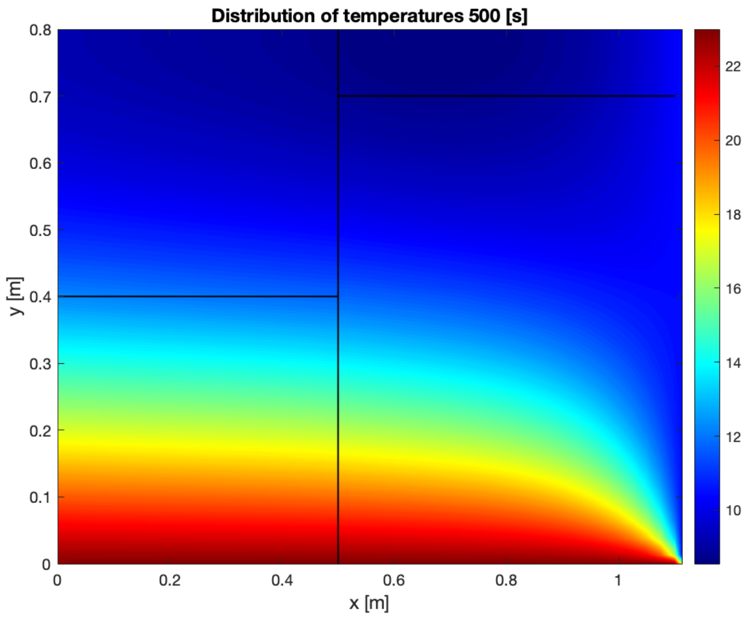


Figure 4.17: Temperatures distribution case 1 at t=500s

It can be observed that the distribution at this earlier time is different. The isotherm wall (the bottom wall) is the hottest wall of the case. This is because the initial temperature of the case is $T = 8$ºC which means all the heat of the isotherm, including the uniform heat flow that goes into the top wall, makes the right wall, goes to the right wall (which also this wall increases with the time).

However, the left wall, which has an equilibrium between convection and conduction goes from hotter to cooler in the part of the material M1. This is because the material M1 has less conductivity and specific heat which makes it takes more time to reduce the heat of the fluid with $T_g = 33$ ºC that passes through this wall.

Finally, a figure of $t = 5000s$ is presented in this report to see if there is a difference between the final time and this mid-time.

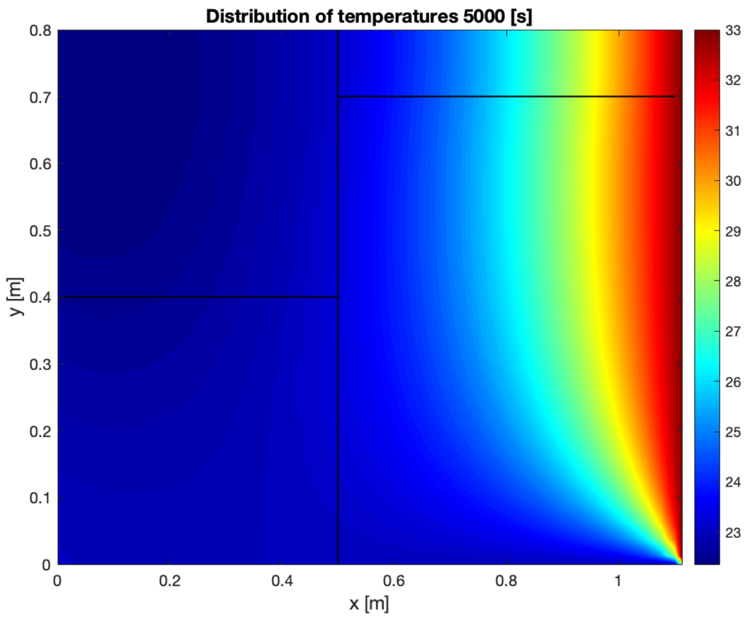


Figure 4.18: Temperatures distribution case 1 at t=5000s

The difference between 4.16 and 4.18 is the distribution of the temperatures in the left and right walls. The fourth material (top-right of the wall) starts very cool, for the graphic 4.17 from less than 10ºC and passes through 23ºC in 4.18 to finally obtain 25ºC.

On the other hand, the right wall starts cooler too but in the mid-time, figure 4.18 it starts to become the hotter part of the problem, as in the final time. Also, it can be seen that Material M2 and Material M4 change the temperature almost at the same time in each graphic, this can be because they have the same conductivity although they have different specific heat.

4.4.6 Verification of the code

In order to know if the code is correct, the CTTC has provided one map of temperatures at a concrete time, this is t=5000s. The comparison of the graphics is shown below:

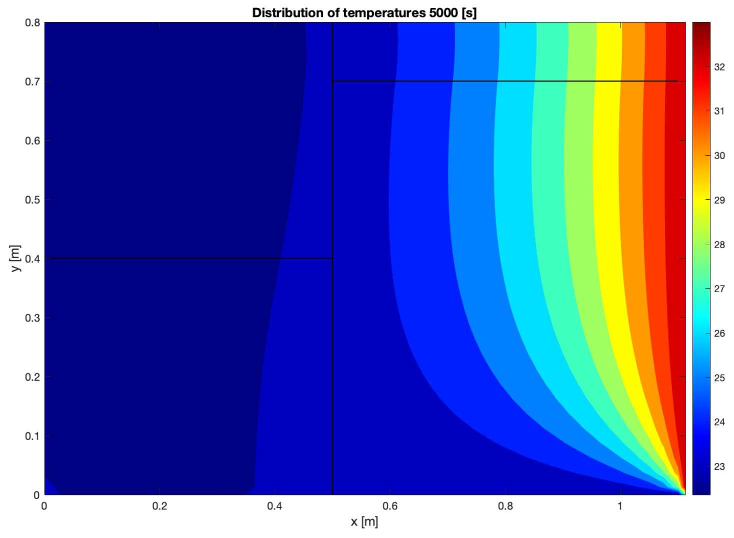


Figure 4.19: Temperatures distribution case 1 at t=5000s

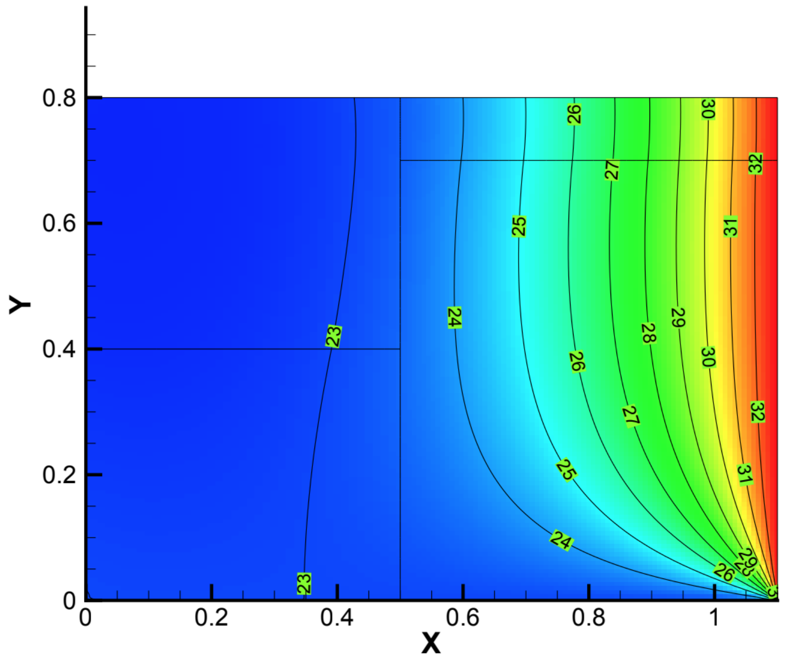


Figure 4.20: Solution of case 1 provided by CTTC

It can be analyzed that both graphics had the same temperature distribution at this time. Also, the graphic used to represent the specific time is less accurate than the showed in the previous section to observe better the difference in the temperatures.

In conclusion, the problem has been verified and it can be said that the code is correct.

4.4.7 Study of two points of the domain

In this section, two case points will be studied to know what happens during the case time in a specific part. The points are:

- $P_A = [0.6\ 0.6]$
- $P_B = [0.5\ 0.7]$

Figure 4.21 is shown the distribution of the points inside the final result of the case.

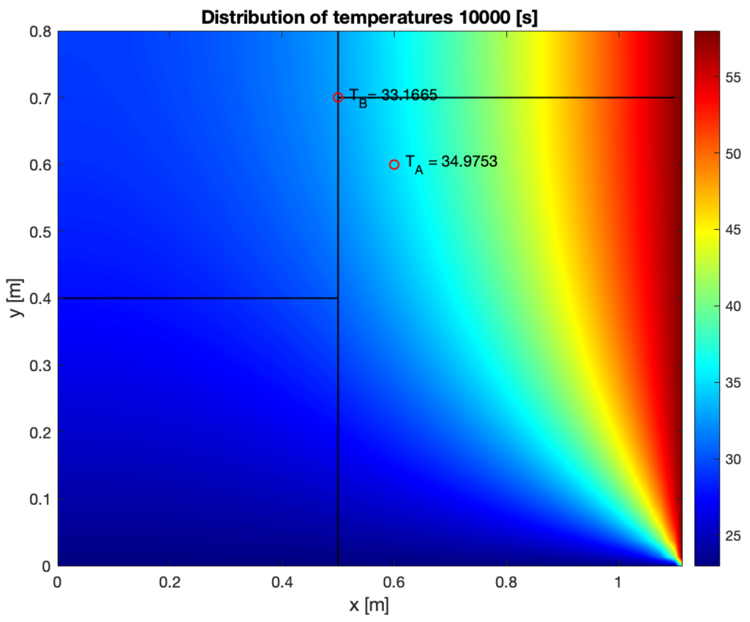


Figure 4.21: Representation of points P_A and P_B on the domain

Point A is represented in the material M2 meanwhile point B is represented in the middle of three different materials. This representation is to perceive the evolution of the different materials inside the domain and what happens in the line of separation of the materials.

In the next figure 4.22 it can be noted the distribution of temperatures in the same point for all the time from 0 seconds to 10000 seconds.

In the first seconds of the simulation, both points are at the same time, initially the case is in T=8ºC. Later, at time t=5000s it starts to become different temperatures. That is because the material properties are different at each point. Even though in Point B there is a property mix it can be analyzed that properties from Material 3, which is the most conductive, are dominant in this point (because the temperatures in A and B are quite different and they are near in the domain).

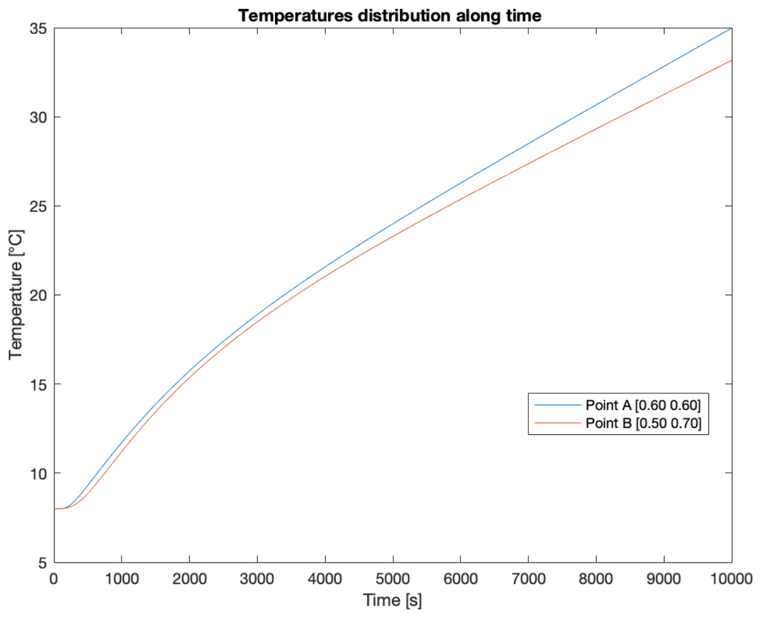
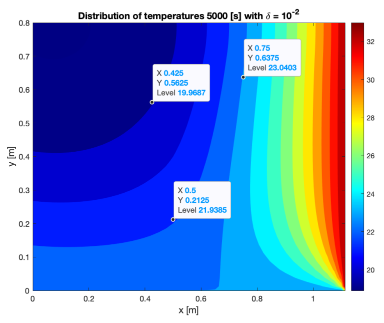


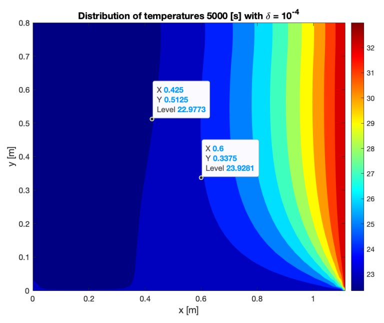
Figure 4.22: Evolution of the temperatures in each point along time in seconds

4.4.8 Numerical parameters

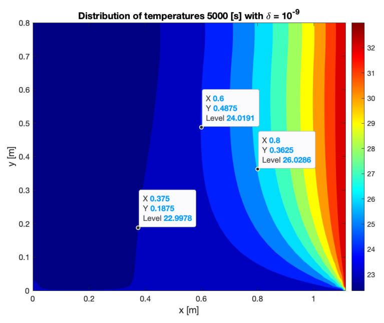
This section is going to be studied the influence of the parameter δ on the computation time. For this reason, three plots of the same case will be computed with different δ to observe the changes in the solution.



(a) $\delta = 10^{-2}$



(b) $\delta = 10^{-4}$



(c) $\delta = 10^{-9}$

Figure 4.23: Results of the case with different δ

As can be analyzed in 5.7 when $\delta < 10^{-4}$ the results can be said as correct. The only parameter to choose δ is now the compilation time. The next figure 4.24, it is shown the difference between different δ and their compilation time.

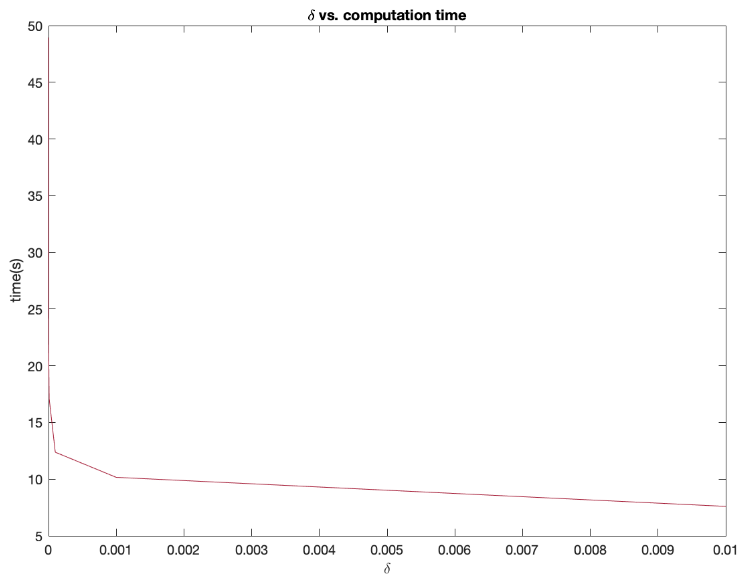


Figure 4.24: Computation time vs. δ

It is obvious that the time of computation increases when the error parameter, in this case, δ, decreases because the code wants more resolution of the solution.

Nevertheless, at one point the computation time increases exponentially. This is when $\delta > 10^{-6}$, which means, this is the limit to obtain a great resolution but with a short computation time. This can be justified also with figures 5.7 because the difference between $\delta = 10^{-4}$ and $\delta = 10^{-9}$ in the temperature evolution are insignificant.

In addition, the parameters N and M have been done a similar study to obtain the minimum number of elements to obtain a good approximate solution but with a short computation time.

4.5 Summary

The discretization of the diffusion equations for use in solving heat conduction issues in solid materials has been the focus of this chapter. The mathematical procedures have been justified because it is the first discretization method used in that thesis. Following the establishment of the theoretical framework, the necessary Matlab functions were created to address two verification instances put forth by the CTTC. A straightforward heat conduction issue and the instance of the four materials.

On the other hand, in the case of the first scenario, the verification process involved putting up certain kinds of boundary conditions in the code to produce a 1D heat conduction profile and verifying it with a heat balance (as in theory class).

Chapter 5

Convection-diffusion equation

The goal of this section is to create a Matlab program, as in the other chapter, that can use FVM to solve a transient, 2D convection-diffusion problem. As in the previous section, a brief theory will be explained and then the equation will be discretized to compute a code that can solve the problem mentioned before. Lastly, by the CTTC, various benchmark scenarios will be proposed to verify the numerical code.

5.1 Convection-diffusion theory

To introduce the Convection - diffusion theory first of all the Navier-Stokes equations for a perfect gas with a constant value of c_v where c_v is the heat capacity of a substance in a constant volume. The equations are written as follows,

Mass conservation equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{5.1}$$

Momentum conservation equation:

$$\frac{\partial (\rho \vec{v})}{\partial t} + \nabla (\rho \vec{v} \vec{v}) = \nabla \cdot (\mu \nabla \vec{v}) + [\nabla \cdot (\vec{\tau} - \mu \nabla \vec{v}) - \nabla \cdot p + \rho g] \tag{5.2}$$

Energy conservation equation:

$$\frac{\partial (\rho T)}{\partial t} + \nabla \cdot (\rho \vec{v} T) = \nabla \cdot (\frac{\lambda}{c_v} \nabla \cdot T + \left(\frac{-\nabla \cdot \dot{q}^r - p \nabla \cdot \vec{v} + \vec{\tau} : \nabla \cdot \vec{v}}{c_v} \right) \tag{5.3}$$

All these equations have a common structure composed of an unsteady term, a convective term, and, finally, a diffusion term among others.

The generic convection-diffusion transport equation can be rewritten with a generic variable (Φ) which can be velocity, temperature, mass, the fraction of species, or entropy...

$$\frac{\partial(\rho\Phi)}{\partial t} + \nabla\cdot(\rho\vec{v}\Phi) = \nabla\cdot(\Gamma_\Phi\nabla\Phi) + \dot{S}_\Phi \tag{5.4}$$

where $\dot{S}_\Phi$ is the extra source/sink terms and Γ_Φ the diffusion coefficient.

Using the mass conservation equation 5.1 and the previous equation 5.4 it can be written an equivalent convection-diffusion equation:

$$\rho\frac{\partial\Phi}{\partial t} + \rho\vec{v}\cdot\nabla\Phi = \nabla\cdot(\Gamma_\Phi\nabla\cdot\Phi) + \dot{S}_\Phi \tag{5.5}$$

As a result, the table 5.1 shows the generic convection-diffusion form for the mentioned Navier-Stokes equations.

Equation	$\boldsymbol{\Phi}$	$\boldsymbol{\Gamma_\Phi}$	$\dot{\mathbf{S}}_{\boldsymbol{\Phi}}$
Mass [5.1]	1	0	0
Momentum [5.2]	$\vec{v}$	μ	$\nabla\cdot(\vec{\tau} - \mu\nabla\vec{v}) - \nabla\cdot p + \rho g)$
Energy [5.3]	T	$\frac{\lambda}{c_v}$	$\frac{-\nabla\cdot\dot{q}^r - p\nabla\cdot\vec{v} + \vec{\tau}:\nabla\cdot\vec{v}}{c_v}$

Table 5.1: Solutions to the different Navier-Stokes equations extracted from [9]

5.2 Equation Discretization

In this part of the thesis, different numerical methods will be explained in order to compute the convection-diffusion equation (meaning transforming from a non-linear equation to linear equations that can be programmed). Before that, the spatial and time discretization will be mentioned in other sections.

5.2.1 The generic convective-diffusion equation

Some assumptions have to be done before discretizing the equation. For this reason, it would be considered a 2D problem and it would be integrated over a ΔV and a Δt. Numerical implicit approximations of the different terms are shown [9]:

$$\int_{t^n}^{t^{n+1}}\int_{V_p}\frac{\partial(\rho\phi)}{\partial t}dVdt \approx V_p(\rho_P\Phi_P - \rho_P^0\Phi_P^0)$$

$$\int_{t^n}^{t^{n+1}}\int_{V_P}\nabla\cdot(\rho\vec{v}\Phi)dVdt \approx (\dot{m}_e\Phi_e - \dot{m}_w\Phi_w + \dot{m}_n\Phi_n - \dot{m}_s\Phi_s)\Delta t$$

$$\int_{t^n}^{t^{n+1}}\int_{V_P}\nabla\cdot(\Gamma_\Phi\nabla\cdot\Phi)dVdt = \int_{t^n}^{t^{n+1}}\int_{S_f}\Gamma_\Phi\nabla\cdot\Phi\cdot\vec{n}dSdt \approx$$

$$\approx (\Gamma_e \frac{\Phi_E - \Phi_P}{d_{PE}} S_e - \Gamma_w \frac{\Phi_P - \Phi_W}{d_{PW}} S_w + \Gamma_n \frac{\Phi_N - \Phi_P}{d_{PN}} S_n - \Gamma_s \frac{\Phi_P - \Phi_S}{d_{PS}} S_s)\Delta t$$

$$\int_{t^n}^{t^{n+1}} \int_{V_P} \dot{S}_\Phi dV dt \approx (S_C^\Phi + S_P^\Phi) V_p \Delta t$$

Notice that super-index n, corresponding to the previous time step is substituted by 0, and super-index (n+1) is eliminated from the equations. Moreover, the source term is linearized to make the numerical methods easier to compute.

Secondly, all the terms showed before can be rearranged in the convective-diffusive equation 5.5:

$$\frac{\rho_P \phi_P - \rho_P^0 \Phi_P^0}{\Delta t} V_p + \dot{m}_e \Phi_e - \dot{m}_w \Phi_w + \dot{m}_n \Phi_n - \dot{m}_s \Phi_s =$$

$$= D_e(\Phi_E - \Phi_P) - D_w(\Phi_P - \Phi_W) + D_n(\Phi_N - \Phi_P) - D_s(\Phi_P - \Phi_S) + (S_C^\Phi + S_P^\Phi \Phi_P) V_p$$

where $D_k = \frac{\Gamma_k S_k}{d_{PK}}$, $k \in \{e, s, n, w\}$ and $K \in \{E, S, N, W\}$.

Notice that a simplification can be done in the volume V_p. As in the other section, see equation 4.4, it can be expressed as $V_p = \Delta x \cdot \Delta y \cdot 1$. Now the terms can be rearranged as:

$$\frac{\rho_P \phi_P - \rho_P^0 \phi_P^0}{\Delta t} \Delta x \cdot \Delta y + J_e - J_w + J_n - J_s = (S_C + S_P \phi_P) \Delta x \Delta y$$

where the J_k are the diffusive terms rearranged in one coefficient. This term can be written also as follows:

$$J = \rho u \Phi - \Gamma \frac{\partial \Phi}{\partial t} \tag{5.6}$$

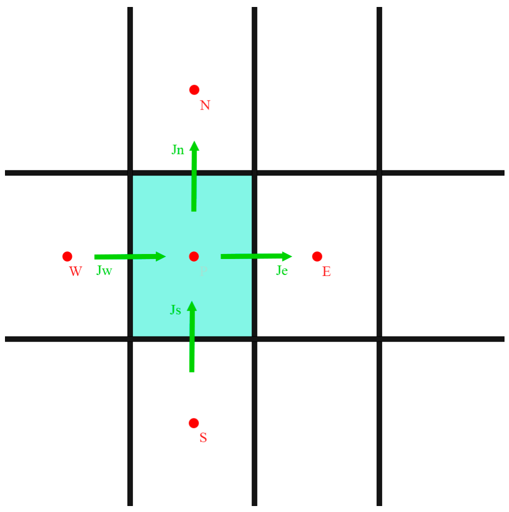


Figure 5.1: Example of J_k inside a control volume, VC

Before continuing, some variables have to be defined, the convective force, F, and the diffusivity conductance D.

$$F = \rho u$$

$$D = \frac{\Gamma}{\delta}$$

From this two-equation a relationship between the convective force and the diffusivity conductance can be done, this is called the Péclet number [10]. If this number tends to zero is a convective pure problem, on the other way around, if the number tends to 1 is a diffusivity conductance pure problem. This number will be important in the next section, 5.2.2.

$$Pe = \frac{F}{D} = \frac{\rho u \delta}{\Gamma} \tag{5.7}$$

On the other hand, the discretized mass conservation equation can be used to find an equivalent form of the above equation.

$$\begin{aligned} &\rho_P^0 \frac{\phi_P - \phi_P^0}{\Delta t} V_p + \dot{m}_e(\phi_e - \phi_p) - \dot{m}_w(\phi_p - \phi_w) + \dot{m}_n(\phi_n - \phi_p) - \dot{m}_s(\phi_p - \phi_s) = \\ &D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W) + D_n(\phi_N - \phi_P) - D_s(\phi_P - \phi_S) + (S_C^\Phi + S_P^\Phi)V_p \end{aligned} \tag{5.8}$$

As can be seen the diffusion and source terms in this implicit form of the convective-diffusive equation are second-order accurate. The major issue at hand is how to describe the terms of convective distribution in terms of numerical methods.

5.2.2 Numerical schemes

When the values of a property, in this case, Φ, at the neighboring nodes are known, numerical schemes have been devised to interpolate the value at the faces of these nodes. Numerical schemes may be divided into two categories: low-order schemes and high-order schemes [9].

In addition, when the flow field is perpendicular to the grid lines and the dependent variable displays a nonzero gradient in the normal direction of the flow, first-order methods, such as the upwind scheme, are numerically stable but extremely diffusive. Researchers have created a number of high-order schemes, such as the QUICK scheme or the second-order upwind scheme, to address this weakness and improve the accuracy of the projected outcomes [11].

Central Difference Scheme (CDS)

The arithmetic mean of the Φ variable from the two near nodal values is calculated.

$$\Phi_e = \frac{1}{2}(\Phi_P + \Phi_E) \tag{5.9}$$

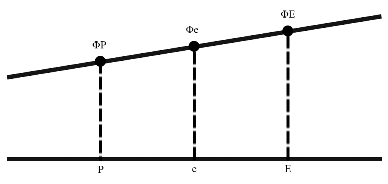


Figure 5.2: Example of the Central Difference Scheme

Upwind Difference Scheme (UDS)

According to this technique, the variable Φ at the cell face equals the value of the closest upwind node (in incompressible fluids the convective terms are more influenced by upstream than downstream conditions). The First-order scheme describes it [9].

$$\Phi_e - \Phi_P = f_e(\Phi_E - \Phi_P) \tag{5.10}$$

if $f_e = 0$ when $\dot{m}_e > 0$ and if $f_e = 1$ when $\dot{m}_e < 0$. This expression can be written as [9],

$$\dot{m}_e(\Phi_e^{UDS} - \Phi_P) = \frac{\dot{m}_e - |\dot{m}_e|}{2}(\Phi_E - \Phi_P)$$

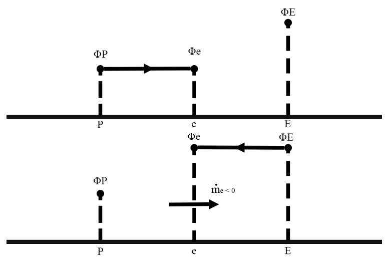


Figure 5.3: Example of the Upwind Difference Scheme

Exponential Difference Scheme (EDS)

The basic principle of the exponential-difference scheme is the assumption of a Φ distribution between nodal sites based on a convection-diffusion equation that has been simplified and does not include the source term. For instance, for the east face [9]:

$$\Phi_e - \Phi_P = f_e(\Phi_E - \Phi_P) \tag{5.11}$$

where $f_e = \frac{e^{P_e d_{pe}/d_{PE}} - 1}{e^{P_e}} - 1$ and the Peclet number, P_e is

$$\frac{\rho_e v_{xe} d_{PE}}{\Gamma_e}$$

5.2.3 Coefficients discretized

Now that equation 5.8 has been discretized it can be expressed with coefficients as the heat conduction problem.

$$a_P\Phi_P = a_W\Phi_W + a_E\Phi_E + a_S\Phi_S + a_N\Phi_N + b_P \tag{5.12}$$

where

$$\begin{gathered}
a_E = D_e A(|Pe_e|) + max(-Fe, 0) \\
a_W = D_w A(|Pe_w|) + max(Fw, 0) \\
a_N = D_n A(|Pe_n| + max(-Fn, 0)) \\
a_S = D_s A(|Pe_s|) + max(Fs, 0) \\
a_P = a_E + a_W + a_N + a_S + \frac{\rho_P^0 \Delta x \Delta y}{\Delta t} - S_p \Delta x \Delta y \\
b_P = S_C \Delta x \Delta y + a_P^0 \Phi_P^0
\end{gathered}$$

and,

$$
\begin{aligned}
D_e &= \frac{\Gamma_e \Delta y}{d_{PE}} \\
D_w &= \frac{\Gamma_w \Delta y}{d_{PW}} \\
D_n &= \frac{\Gamma_n \Delta x}{d_{PN}} \\
D_s &= \frac{\Gamma_s \Delta x}{d_{PS}} \\
F_e &= (\rho u)_e \Delta y \\
F_w &= (\rho u)_w \Delta x \\
F_n &= (\rho u)_n \Delta x \\
F_s &= (\rho u)_s \Delta x
\end{aligned}
$$

The Péclet number, Pe_k can be expressed as, mentioned before in 5.7, $Pe_k = \frac{F_k}{D_k}$, $k \; \epsilon \; \{e, s, n, w\}$. The coefficient $A(|Pe|)$, is useful because it is an approximation of the Peclet number and can be used for expressing different numerical schemes. The function of this coefficient will be determined with table 5.2, which is extracted from [10].

Scheme	**Formula for** $A(\lvert Pe\rvert)$
CDS	$1 - 0.5\lvert Pe\rvert$
UDS	1
HDS	$max(0, 1 - 0.5\lvert Pe\rvert)$
Power law (PLDS)	$max(0, (1 - 0.1\lvert Pe\rvert)^5)$
Exponential (exact)	$\frac{\lvert Pe\rvert}{e^{Pe}-1}$

Table 5.2: Function $A(|Pe|)$ for different schemes [12]

5.2.4 Boundary conditions

Discretization coefficients for internal nodes were explained in section 5.2.1. Nevertheless, when expressing those coefficients for border nodes, extra considerations must be made to obtain a correct solution.

Dirichlet boundary conditions

Dirichlet boundary conditions apply to nodes with predetermined property values. Where X is the border node's closest node and Φ_{wall} its value, discretization coefficients are [9]:

$$a_W = a_E = a_S = a_N = 0$$
$$a_P = 1$$
$$b_P = \Phi_{wall}$$

To make it clear the figure below shows this condition:

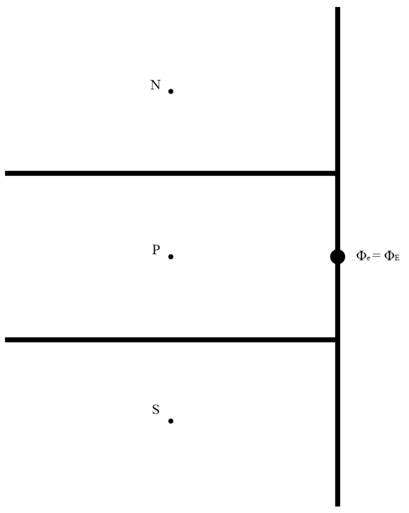


Figure 5.4: Dirichlet boundary condition

Neumann boundary conditions

In Neumann boundary conditions the value known is the derivative of the predetermined property, which in this case is the heat flux. Considering X the nearest node of the boundary and j the value is known, discretized coefficient are [9]:

$$a_{noX} = 0$$
$$a_X = 1$$
$$a_P = 1$$
$$b_P = j\frac{d_{PX}}{\Gamma_P}$$

Once again, to make the concept clear a figure is shown:

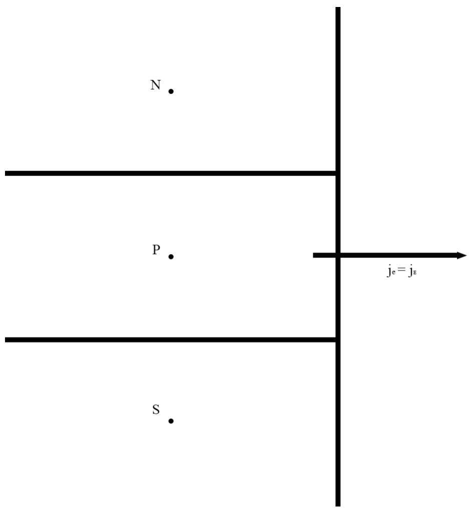


Figure 5.5: Neumann boundary condition

5.2.5 Algorithm

The code will follow the same structure explained in 3.4.

1. Input data: a) Physical data (geometry, materials, boundary conditions...) b) Numerical data (mesh, time step, convergence criteria...).
2. Previous calculations (Δx, Δy, $S[i][j]$, $r_x[i][j]$, $r_y[i][j]$).
3. Velocity map calculations ($v_w[i][j]$, $v_e[i][j]$, $v_s[i][j]$, $v_n[i][j]$).
4. Material properties map: $\rho[i][j] = \rho(x, y)$ and $\Gamma[i][j] = \Gamma(x, y)$
5. Peclet number map calculations: $Pe_e[i][j], Pe_w[i][j], Pe_n[i][j], Pe_s[i][j]$
6. Numerical scheme assignation: $A(|Pe_e|), A(|Pe_w|), A(|Pe_n|), A(|Pe_n|)$
7. Initial map ($t^n = 0$). $\Phi^n[i][j] = \Phi_0[i][j]$
8. Computation of the next time step, $t^{n+1} = t^n + \Delta t$
   (a) Estimation of $\Phi^{n+1}[i][j]$
   (b) Computation of the discretization coefficients $a_W[i][j], a_E[i][j], a_S[i][j], a_N[i][j]$ and $b_P[i][j]$
   (c) Use line-by-line method explained in section 3.3
   (d) Max$|\Phi^{n+1}[i][j] = \Phi^n[i][j]| < \delta$?
      - If the answer is **Yes** go to step 9.
      - If the answer is **No** go to step 8 and $\Phi^{n+1}[i][j] = \Phi^{n+1}[i][j]$
9. New time step?
   (a) If the answer is **Yes** go to step (a) and $\Phi^{n+1}[i][j] = \Phi^n[i][j]$.
   (b) If the answer is **No** go to step 10.

10. Print the results and show the conclusions.

5.2.6 Developed code

The code's structure is similar to the one used in the heat conduction problem, it can be seen in section 4.2.5. This code is divided into functions to make the code easier to understand and to be organized.

In order to find if this code is correct or not a verification has to be made. In this case, the CTTC has provided some benchmark cases [9]. These cases are the diagonal flow and the Smith-Hutton flow. Although it exists an analytical solution for the first one, there is no one for the Smith-Hutton flow. However, some experimental solutions are used to verify the code.

Notice that the cases of horizontal and vertical flow are a simplified version of the diagonal flow so they are not shown in this thesis.

5.3 Diagonal flow

The verification case is a rectangular domain, which dimensions are a square of 1x1, and inside there is a diagonal flow. The flow has these velocity components:

$$U = V_0 \cdot cos(\alpha)$$
$$V = V_0 \cdot sin(\alpha)$$

where α is the angle formed by the direction of the flow and the x-axis. In terms of boundary conditions, walls above the diagonal have a variable value of Φ_1, while those below the diagonal have a variable value of Φ_2. The figure below is shown the conditions of the problem:

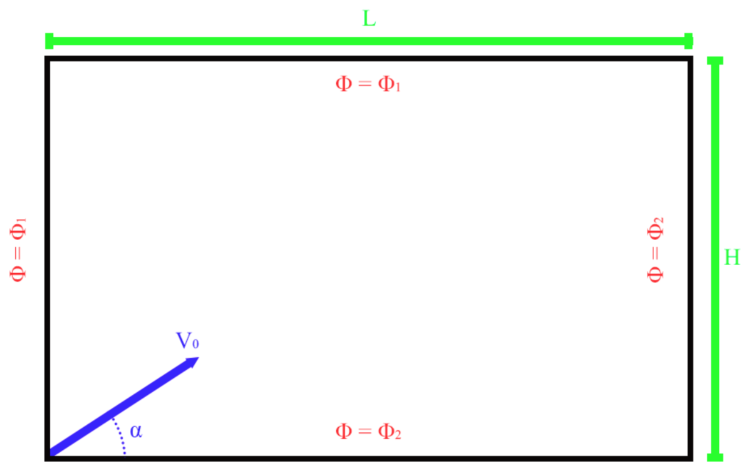


Figure 5.6: Diagonal Flow verification problem [9]

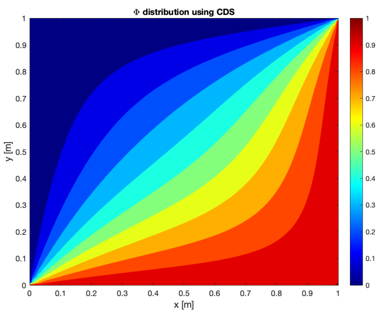
Talking about the analytical solution only exists for an infinite Peclet number value. However, numerous simulations with rising Peclet numbers were run. The property boundary value was set to $\phi_1 = 0$ above the diagonal and $\phi_2 = 1$ below the diagonal (as mentioned before). In terms of mesh size, this will be L= 1 m and H = 1 m. The outcomes are displayed below.

On the other hand, to compare the first-order scheme with the second-order scheme the problem will be solved using EDS, UDS, CDS, and QUICK schemes to compare the differences. Moreover, in this case, will be using Gauss-Seidel 3.3.

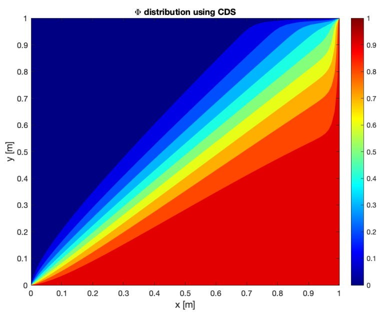
5.3.1 Results

This section will be showing the different numerical schemes with the scenario explained before. To see the influence of the Peclet number will be analyzed this scenario in different circumstances.

First of all, will be showing the solution using CDS for low, medium, and high Peclet numbers.



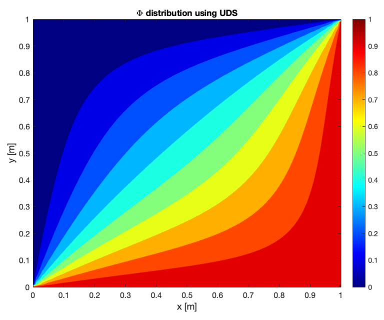
(a) Lower Peclet Number ($P_e \approx 1$)



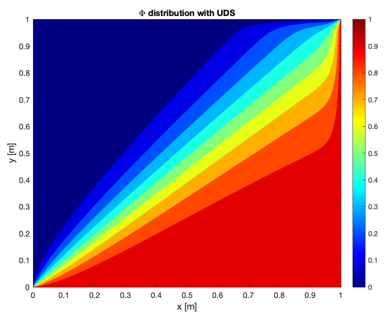
(b) Medium Peclet Number ($P_e \approx 100$)

Figure 5.7: Results of the case with different Peclet Number using CDS

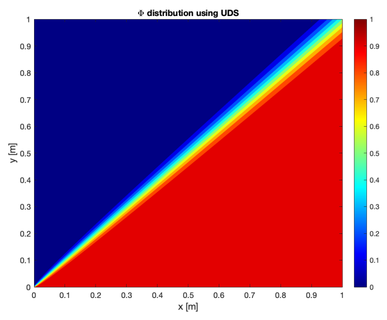
The code did not converge for high Peclet numbers. This can be because it is the most simple first-order scheme and the solver can make an exact solution. That is one of the reasons to use another numerical scheme in the next problem, which has more complications than this. Secondly, the same scenario but using UDS is represented.



(a) Lower Peclet Number ($P_e \approx 1$)



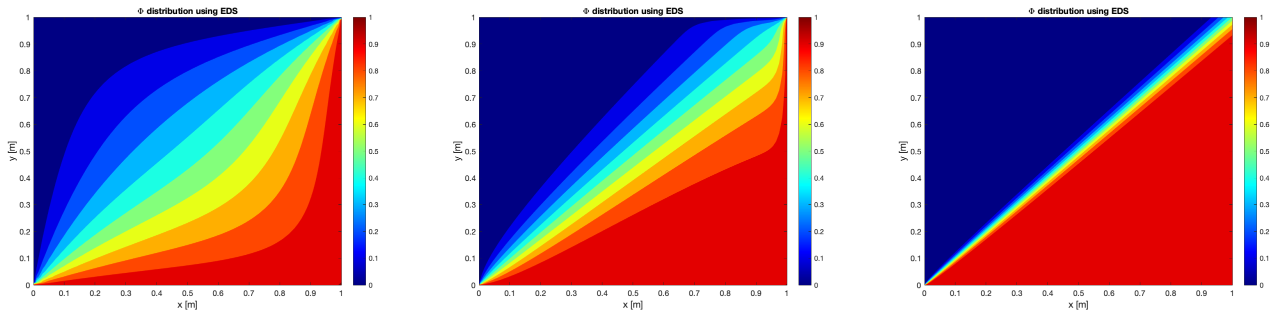
(b) Medium Peclet Number ($P_e \approx 100$)



(c) High Peclet Number ($P_e \approx 10^6$)

Figure 5.8: Results of the case with different Peclet Number using UDS

For the last first-order scheme, the scenario will be solved using EDS,

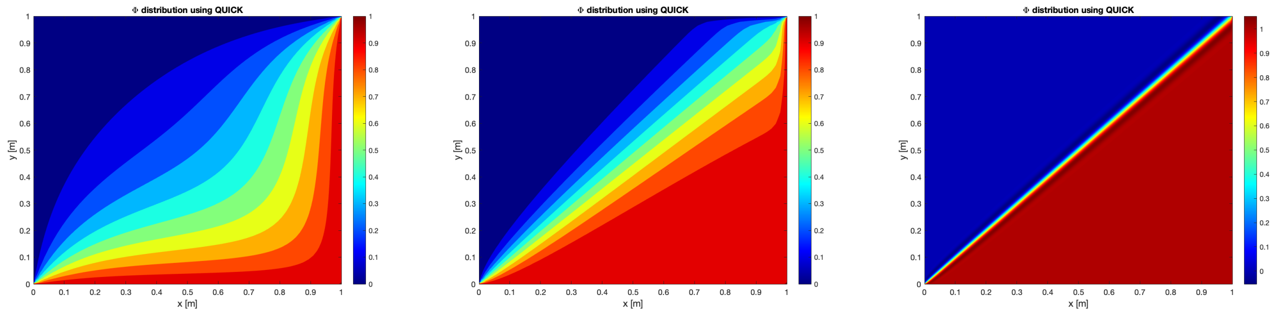


(a) Lower Peclet Number ($P_e \approx 1$) (b) Medium Peclet Number ($P_e \approx 100$) (c) High Peclet Number ($P_e \approx 10^6$)

Figure 5.9: Results of the case with different Peclet Number using EDS

All this will be compared with the QUICK scheme to analyze if there's any difference between using a second-order scheme or using a first-order scheme (this last one is faster to implement because it has less computation time).

The results using QUICK are deployed below:



(a) Lower Peclet Number ($P_e \approx 1$) (b) Medium Peclet Number ($P_e \approx 100$) (c) High Peclet Number ($P_e \approx 10^6$)

Figure 5.10: Results of the case with different Peclet Number using QUICK

As can be seen for the lower Peclet Number using a high-order scheme only increases the computation time but the results are less accurate (because it has to be used fewer elements because the computation time increases exponentially) than the ones obtained with the first-order schemes. Nevertheless, for high Peclet numbers, the most accurate one is the QUICK scheme which is also the faster one.

Overall, if it has to be chosen one of these four schemes the Upwind or the Exponential would be selected because they represent very well the distribution of Φ in any scenario (this means for any Peclet number) and the computation time is short.

5.3.2 Verification of the code

It can be noted that the transition zone between the high and low values gets narrower as the number of Peclet increases. Therefore, if the tendency holds, it can be said that for an infinite ratio number, the computed property over the diagonal would be upper the diagonal equal to 0 and below the diagonal equal to 1. The property fluctuation along the horizontal mid-plane is depicted in the following figure 5.11.

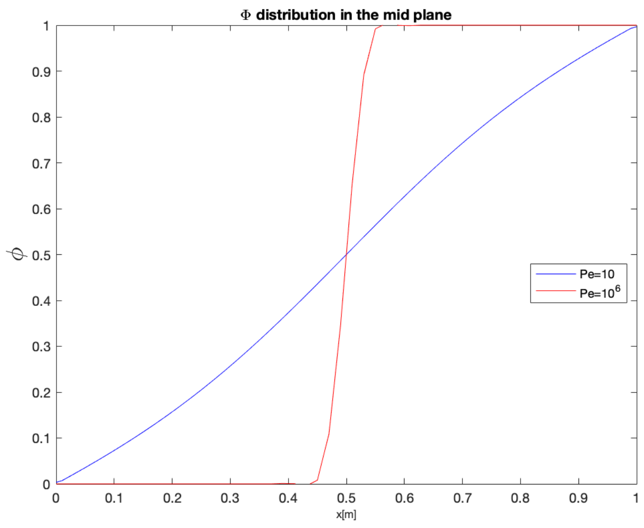


Figure 5.11: ϕ distribution with diagonal flow along the x mid-plane

5.4 Smith-Hutton flow

The last verification of the code consisted of a rectangular domain of dimensions L = 2m and H = 1m. The condition of this case is that the right half of the lower wall served as an outflow, while the left half was an inlet. Figure 5.12 show the problem.

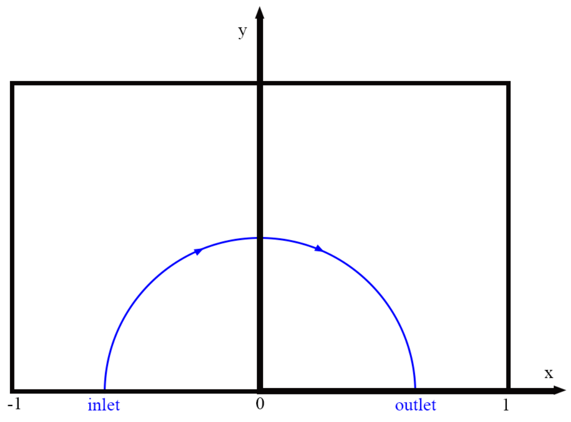


Figure 5.12: Smith-Hutton flow verification problem [9]

The velocity components are:

$$u(x,y) = 2y(1-x^2)$$
$$v(x,y) = -2x(1-y^2)$$

Furthermore, the Φ value in the inlet part is $\Phi = 1 + tanh[10(2x+1)]$ while the outlet part is zero. On the other hand, in the rest of the domain, the variable value is $\Phi = 1 - tanh(10)$.

To verify this case there are no analytical answers to the problem. Nevertheless, the CTTC [9] has provided a few reference values for different ρ/Γ ratios.

x[m]	$\rho/\Gamma = 10$	$\rho/\Gamma = 10^3$	$\rho/\Gamma = 10^6$
0	1.989	2.000	2.000
0.1	1.402	1.999	2.000
0.2	1.146	1.999	2.000
0.3	0.946	1.9850	1.999
0.4	0.775	1.841	1.964
0.5	0.621	0.951	1.000
0.6	0.480	0.154	0.036
0.7	0.349	0.001	0.001
0.8	0.227	0	0
0.9	0.111	0	0
1.0	0	0	0

Table 5.3: Values to verify the problem extracted from [9]

5.4.1 Results

The results are presented in this section. First, the distribution of Φ for different Peclet numbers is shown as with the diagonal flow in section 5.3.1.

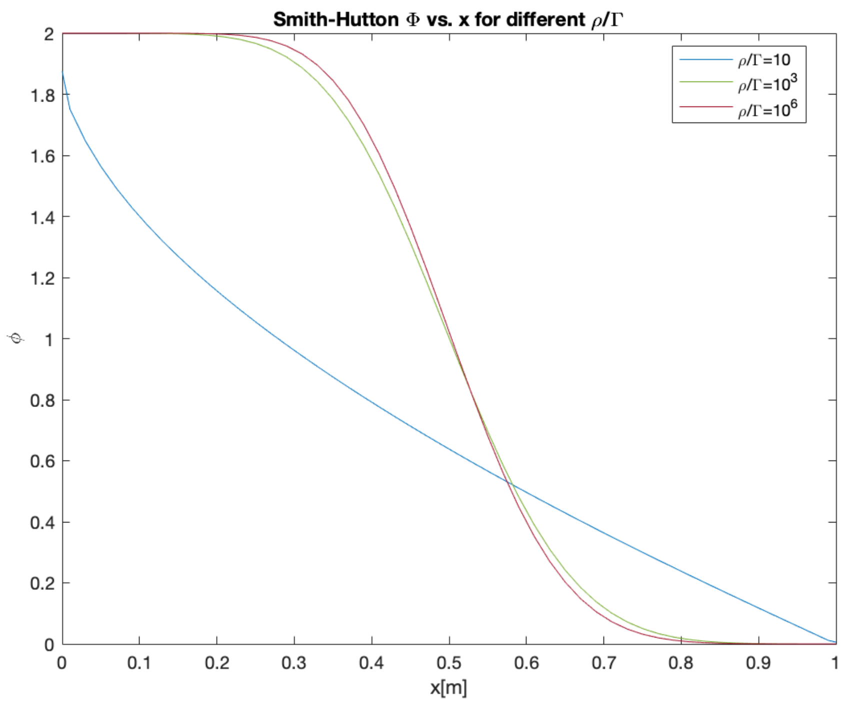


Figure 5.13: Smith-Hutton flow distribution with different ρ/Γ

The following figures are shown the distribution of the flow for different scenarios.

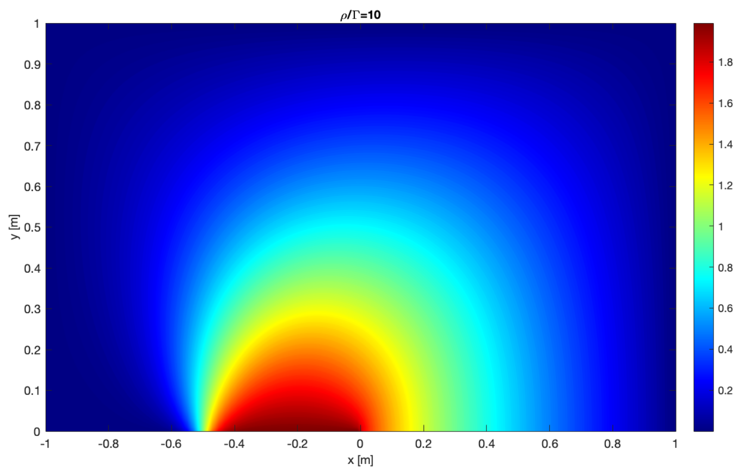


Figure 5.14: Φ distribution with Smith-Hutton flow in $\rho/\Gamma = 10$ scenario

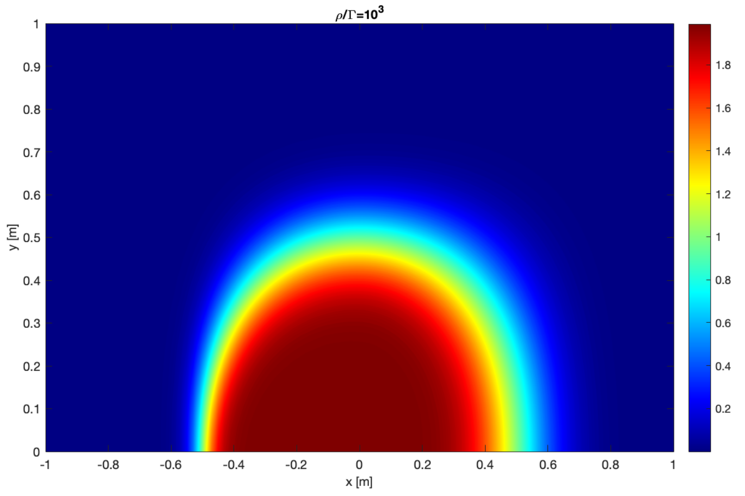


Figure 5.15: Φ distribution with Smith-Hutton flow in $\rho/\Gamma = 10^3$ scenario

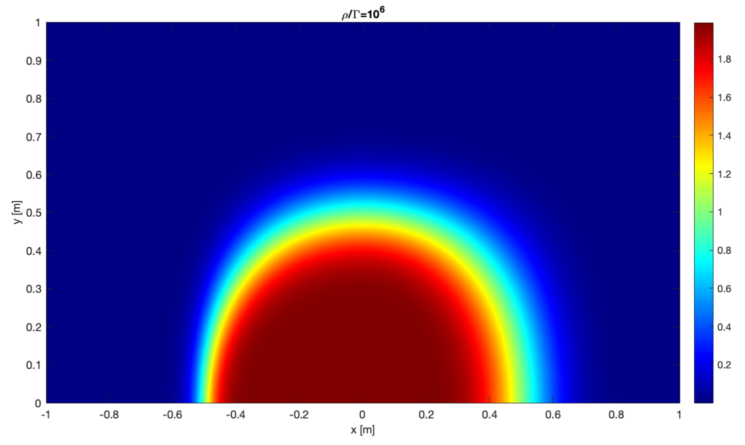


Figure 5.16: Φ distribution with Smith-Hutton flow in $\rho/\Gamma = 10^6$ scenario

5.4.2 Verification of the problem

In order to verify the case it has to be taken the 5.3 table and the figure shown in the figure 5.13.

The case has been solved using the *Upwind* numerical scheme, which has been shown to be the best to fit this case. As can be seen, the results of the Smith-Hutton problem match exactly with the table. For instance, when x is 0.5m and ρ/Γ ratio is 10^6 it matches the point 1.000.

In this case, the problem has only been solved using the Upwind solution but can be also used in the QUICK scheme. Using this last scheme only increases the computation time but the accuracy of the solution is similar.

5.5 Summary

For different CTTC's recommended verification situations, the Convection-Diffusion equation has been numerically solved throughout this chapter. The Convection-Diffusion equations were first transformed into a set of algebraic equations to build the mathematical foundation. Finally, the created code was used to solve various verification issues that the CTTC had suggested.

All the cases consisted of a flow, in which a property Φ varied in different directions, at the first one it varied in a diagonal way, and in the second one like a half circle. The aim of the chapter was to compute its variation for different P_e numbers and numerical schemes and see which one fits better in this type of problem. Concerning the influence of the non-dimensional parameter, it has been noticed that as it incremented, the variation of the calculated property was produced closer to the right boundary in a more abrupt way as the addictive transport mechanism gained more relevance over the diffusive term.

On the other hand, all numerical schemes computed accurate results, especially the EDS. Nonetheless, it has been noticed that the UDS, in the diagonal flow, was the most stable one (which means it has fewer errors in order to compute the code).

Finally, talking about the Smith-Hutton problem has been solved using the UDS, thanks to this stability talked before.

Chapter 6

Case resolution using FSM

The objective of this chapter is to create a Matlab code that uses the Fractional Step Method, FSM, to solve a problem in 2D, transient, and incompressible Navier-Stokes equations with an orthogonal structured mesh 3.1. The FSM will be introduced first in this chapter. Then, the equations will be discretized to support all the actions and presumptions made. Finally, three benchmark cases suggested by the CTTC will be used to verify the implemented code. This case will be the Lid-driven cavity for different Reynolds numbers.

6.1 The Fractional Step Method (FSM)

A numerical technique for solving partial differential equations is the FSM. It entails dissecting a PDE into more manageable subproblems that may be resolved independently and in order.

The PDE's solutions are broken down into several parts using the FSM, each of which entails solving an equation with regard to just one or a small group of variables. To separate the pressure and velocity fields in the solution, the approach frequently alternates between solving equations for the velocity and pressure fields. It may be seen as a projection onto a space of velocity without divergence, hence it is a projection method.

Its essence is the **Gauss-Ostrogradsky theorem**, which states

$$\int_{\Omega} \nabla \cdot \overrightarrow{F} dV = \int_{\delta\Omega} \overrightarrow{F} \cdot \overrightarrow{n} dS \tag{6.1}$$

In order to use this method the two functions that will be used are going to be orthogonal, as can be seen in the figure 6.1 below,

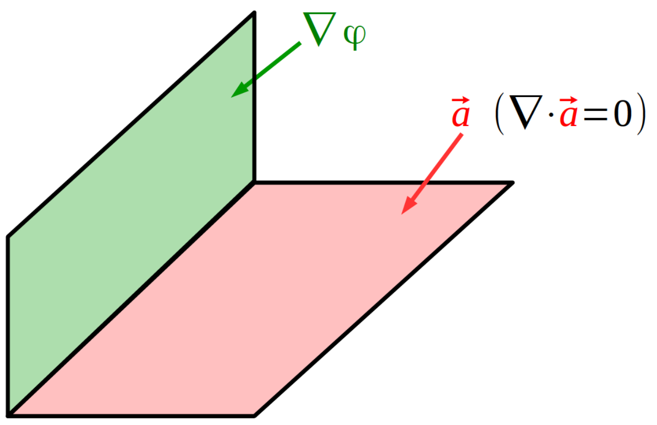


Figure 6.1: Fractional Step Method [13]

6.2 Discretization of the equations

Before using the FSM the first step is to discretize the domain of the problem using a numerical grid. Moreover, time-step stability, which is the second step, and boundary conditions are discussed in this section.

6.2.1 Helmholtz-Hodge theorem (HHT) applied in FSM

According to theory classes [13], to find a solution for the incompressible Navier-Stokes equations the Helmholtz-Hodge has to be used. These equations, as mentioned before in 2, represent the conservation of fluid mass, momentum, and energy. As the equations used are for incompressible fluid the viscosity remains constant, μ.

$$\nabla \cdot \overrightarrow{u} = 0 \tag{6.2}$$

$$\rho \frac{\partial \overrightarrow{u}}{\partial t} + (\rho \overrightarrow{u} \cdot \nabla) \overrightarrow{u} = -\nabla p + \mu \nabla^2 \overrightarrow{u} \tag{6.3}$$

where the fluid density is ρ, the pressure is represented with a p, the viscosity is μ and the velocity field is $\overrightarrow{u}$. Now a constant will be used in order to make the equation above easier. This constant is called $\overrightarrow{R}(\overrightarrow{u})$ and it is defined as,

$$\overrightarrow{R}(\overrightarrow{u}) := -(\overrightarrow{u} \cdot \nabla) \overrightarrow{u} + \mu \nabla^2 \overrightarrow{u} \tag{6.4}$$

The equations must be discretized in time and space before beginning the resolution technique, which is analogous to the one used in earlier chapters. As in theory [13], the discretization is

$$\frac{\overrightarrow{u}^{n+1}-\overrightarrow{u}^{n}}{\Delta t}=\frac{3}{2}\overrightarrow{R}(\overrightarrow{u}^{n})-\frac{1}{2}\overrightarrow{R}(\overrightarrow{u}^{n-1})-\nabla p^{n+1} \tag{6.5}$$

$$\nabla\cdot\overrightarrow{u}^{n+1}=0 \tag{6.6}$$

The first step is to calculate the predictor vector $\overrightarrow{u}^{p}$. Therefore,

$$\frac{\overrightarrow{u}^{p}-\overrightarrow{u}^{n}}{\Delta t}=\frac{3}{2}\overrightarrow{R}(\overrightarrow{u}^{n})-\frac{1}{2}\overrightarrow{R}(\overrightarrow{u}^{n-1}) \tag{6.7}$$

On the second hand, the Poisson equations must be solved for p^{n+1}.

$$\overrightarrow{u}^{n+1}=\overrightarrow{u}^{p}-\Delta t\nabla p^{n+1} \tag{6.8}$$

Finally, a velocity correction should be added.

$$\overrightarrow{u}^{n+1}=\overrightarrow{u}^{p}-\Delta t\nabla p^{n+1} \tag{6.9}$$

In short, the computation process entails calculating the predictor velocity. After that, the velocity will be rectified, followed by the pseudo-pressure field using a linear solution. The process is continued until the desired end time is reached. After the spatial discretization, as seen before, a linear system of equations must be solved in order to obtain these velocities for a temporal mesh. The equation states:

$$\frac{\overrightarrow{u}^{n+1}-\overrightarrow{u}^{n}}{\Delta t}=\frac{3}{2}\overrightarrow{R}(\overrightarrow{u}^{n})-\frac{1}{2}\overrightarrow{R}(\overrightarrow{u}^{n-1})-\nabla p^{n+1} \tag{6.10}$$

$$\nabla\cdot\overrightarrow{u}^{n+1}=0 \tag{6.11}$$

6.2.2 The checkerboard problem

On a node P, finite differences may be used to calculate the velocity at moment n+1 as,

$$u^{n+1}=u^{P}-\frac{\Delta t}{\rho}\left(\frac{p_E^{n+1}-p_W^{n+1}}{2\Delta x}\right) \tag{6.12}$$

As mentioned before, the velocity and the pressure are no longer coupled non-physical pressure fields can result in converged velocity fields because the discrete approximation of Δp^{n+1} at node P does not depend on p_P^{n+1}. Because of this, a different methodology, such as the staggered-meshes method, must be used.

6.2.3 The staggered mesh approach

The staggered meshes will be utilized to address the spatial discretization problem in the preceding section.

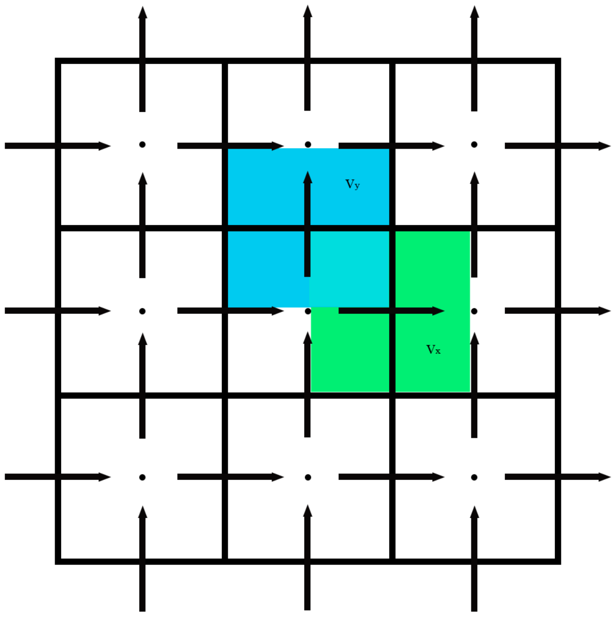


Figure 6.2: Staggered mesh approach

Looking at the figure 6.2 the horizontal component of the velocity is computed using the staggered-x mesh, and the vertical component of the velocity is computed using the y direction. As the structure mesh that is going to be used is $(N_x + 2) \cdot (N_y + 2)$, the staggered-x mesh has a domain of $(N_x + 1) \cdot (N_y + 2)$ and the staggered-y mesh is $(N_x + 2) \cdot (N_y + 1)$. Thus, both of these meshes are subjected to the parallel application of the calculation steps shown in section 6.2.

Staggered-x mesh predictor velocity calculation

The method used to determine the prediction of velocity in the horizontal direction is,

$$u^P = u^n + \frac{\Delta t}{\rho}\left[\frac{3}{2}R(u^n) - \frac{1}{2}R(u^{n-1})\right] \tag{6.13}$$

where R(u) as mentioned in the theory class, for a generic time t [13]:

$$R(u^t) = -(\rho u^t \cdot \nabla)u^t + \mu\nabla u^n$$

The $R(u)$ term is spatially integrated across a standard staggered-x mesh with volume control, VC, inside a domain $\partial\Omega_X$. A volume integral of a divergence field variable is converted into the finite sum of the variable across the control surface using the Gauss theorem [14].

$$\int_{\Omega_x} R(u)d\Omega_x = diffusion - convection$$

The terms *convection* and *diffusion* were used in the previous section which means they have the same discretization as before, leading to,

$$diffusion = \mu_e \frac{u_E - u_P}{d_{EP}} A_e - \mu_w \frac{u_P - u_W}{d_{WP}} A_w + \mu_n \frac{u_N - u_P}{d_{NP}} A_n - \mu_s \frac{u_P - u_S}{d_{SP}} A_s$$

$$convection = \dot{m}_e u_e - \dot{m}_w u_w + \dot{m}_n u_n - \dot{m}_s u_s$$

where $\dot{m}_k$ $k \in s, e, w, n$ is the fluid flow and u_k is the velocity at the control volume face, which is calculated with the numerical schemes showed in section 6.2. On the other hand, the flow rate is computed in the horizontal direction, u, as an interpolation, which in the vertical direction, v, is the contribution of vertical mass flow. This can be expressed as,

$$\dot{m}_e = \frac{1}{2}[(\rho u)_E + (\rho u)_P]A_e$$
$$\dot{m}_w = \frac{1}{2}[(\rho u)_W + (\rho u)_P]A_w$$
$$\dot{m}_n = \frac{1}{2}[(\rho v)_A A_{An} + (\rho v)_B A_{Bn}]$$

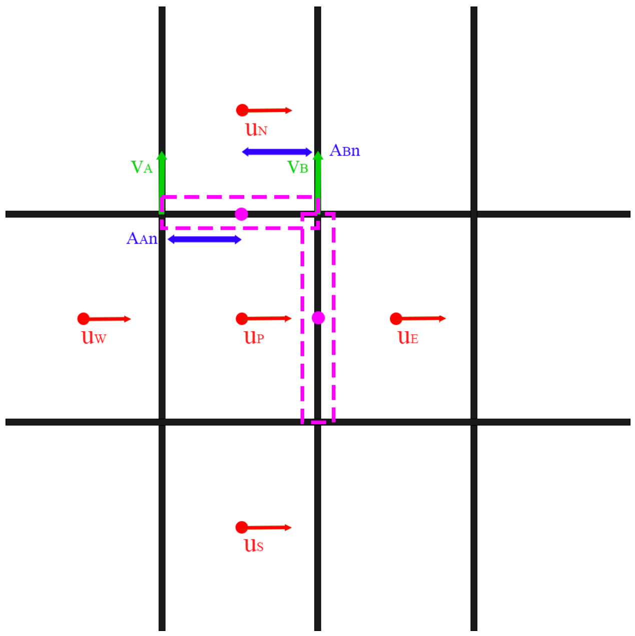


Figure 6.3: Cell face body velocity and mass fow

Staggered-y mesh predictor velocity calculation

The velocity predictor in the vertical direction is calculated by, as before in 6.2.3

$$v^P = v^n + \frac{\Delta t}{\rho}\left[\frac{3}{2}R(v^n) - \frac{1}{2}R(v^{n-1})\right] \tag{6.14}$$

where R(u) as mentioned in the theory class, for a generic time t [13]:

$$R(v^t) = -(\rho v^t \cdot \nabla)v^t + \mu\nabla v^n$$

As in section 6.2.3 the Gauss Theorem [14] is used and this lead to the same expression but in the vertical direction as the previous discretization, 6.2.3.

$$diffusion = \mu_e\frac{v_E - v_P}{d_{EP}}A_e - \mu_w\frac{v_P - v_W}{d_{WP}}A_w + \mu_n\frac{v_N - v_P}{d_{NP}}A_n - \mu_s\frac{v_P - v_S}{d_{SP}}A_s$$

$$convection = \dot{m}_e v_e - \dot{m}_w v_w + \dot{m}_n v_n - \dot{m}_s v_s \tag{6.15}$$

The same happens to calculate the flow rate for each node and the face of the volume of control. As previously mentioned is an interpolation.

$$\begin{aligned}\dot{m}_e &= \frac{1}{2}[(\rho v)_E + (\rho v)_P]A_e\\ \dot{m}_w &= \frac{1}{2}[(\rho v)_W + (\rho v)_P]A_w\\ \dot{m}_n &= \frac{1}{2}[(\rho u)_A A_{An} + (\rho u)_B A_{Bn}]\end{aligned}$$

6.2.4 Pressure calculation in the general mesh

The pressure Poisson equation [15] can be solved after computing the predictor velocity components. The pressure equation is integrated over a standard main grid VC, much like it is done with other differential equations.

$$\int_\Omega \Delta p^{n+1} d\Omega = \frac{\rho}{\Delta t}\int_\Omega \nabla \cdot v_P d\Omega$$

Once again the Gauss theorem [14] is applied,

$$\int_\partial \Omega\nabla p^{n+1} \cdot ndS = \frac{\rho}{\Delta t}\int_\partial \Omega v_P \cdot ndS$$

The discretized equation is then represented using the discretization coefficients, as was done in earlier sections.

$$a_{PP_P}^{n+1} = a_{EP_E}^{n+1} + SP_S{}^{n+1} + NP_N{}^{n+1} + WP_W{}^{n+1}$$
$$a_E = A_e/d_{EP}$$
$$a_S = A_s/d_{SP}$$
$$a_N = A_n/d_{NP}$$
$$a_W = A_w/d_{WP}$$
$$a_P = a_E + a_S + a_W + a_N$$
$$b_P = -\frac{1}{\Delta t}[(\rho u^P)_e A_e + (\rho u^P) n A_n - (\rho u^P)_n A_n - (\rho u^P)_s A_s]$$

A Gauss-Seidel or Line-by-line linear solver, which can be seen in more detail in 3.3, is used to resolve the linear system of equations as in previous sections.

6.2.5 Pressure calculation in the staggered mesh

Calculating the velocity at the posterior time step is possible after obtaining the pressure field. Equation 6.9 is discretized to approximate the derivative terms as finite differences in order to achieve this.

$$u_P^{n+1} = u_P^P - \frac{\Delta t}{\rho}\frac{p_B^{n+1} - p_A n + 1}{d_{BA}}$$
$$v_P^{n+1} = v_P^P - \frac{\Delta t}{\rho}\frac{p_D^{n+1} - p_C^{n+1}}{d_{CD}}$$

where the most significant mesh nodes A and B are located on the left and right side of the staggered-x grid, as seen in figure 6.3 respectively, while C and D are located on the bottom and the top face of the staggered-y grid, respectively.

6.3 Boundary conditions

The above-described computation methods are used for generic nodes, or nodes that are not subject to a specific restriction (boundary condition). It is necessary to define the boundary conditions for the velocity and pressure fields before computing them. Prescribed value, or a Dirichlet condition 3.1.2, or prescribed gradient, or a Neumann condition 3.1.2, can be boundary conditions.

- **Pressure boundary conditions** are wall conditions and the prescribed value of the problem itself.
- **Wall boundary condition**: the pressure variations in the normal direction that are thought to be zero at walls. The mathematical formulation of such circumstance is,

$$\frac{\partial p}{\partial n} \approx \frac{P_P - Pnb}{d_{Pnb}} = 0$$

If it has to be discretized

$$a_{p_{boundary}} = 0$$
$$a_{boundary} = 1$$
$$a_P = a_{p_{boundary}} + a_{boundary} b_P = 0$$

- **Prescribed pressure**: if a specific value is set to a pressure node, its discretization coefficients are

$$a_E + a_W + a_N + a_S = 0$$
$$a_P = 1$$
$$b_P = pressure\ value$$

6.3.1 Boundary conditions of the velocity

It is possible to apply a set velocity value or gradient. The predictor and corrected velocity must also be subjected to the same boundary constraints. Since discretization coefficients are not taken into account while computing the velocity field, such requirements must be set as an exception.

6.4 Time step selection

The time integration method utilized is a semi-explicit one, as was indicated in section 3.2. The time step selection must thus satisfy the stability criteria, as seen in theory class [15]. As a result, the CFL condition determines the least convective and diffusive time step required, and in addition, the time increment utilized is the smallest of those quantities. As per theory class [15] the formulas to obtain this time step are,

$$\Delta t_c = min\left(0.35\frac{\Delta x}{|v|}\right) \tag{6.16}$$

$$\Delta t_d = min\left(0.2\frac{\rho \Delta x^2}{\mu}\right) \tag{6.17}$$

6.5 Global Algorithm

The code will follow the same structure explained in 3.4.

1. Input data: a) Physical data (geometry, materials, boundary conditions...) b) Numerical data (mesh, time step, convergence criteria...).
2. Geometry discretization: the main mesh can be written as two vectors $r_x[j]$ and $r_y[i]$, and the staggered-x mesh as $r_{xx}[j]$ and staggered-y mesh $r_{yy}[i]$
3. Initial map of velocities, pressure and $R(v^{n-1})$
4. Computation of the time step, Δt according to section 6.4
5. Computation at the next time step started: $t^{n+1} = t^n + \Delta t$
6. $R(v^n)$ factor calculation
7. Computation of the predictor velocity, v^P
8. Pressure discretization coefficients: $a_E[i][j], a_S[i][j], a_N[i][j], a_W[i][j], a_P[i][j], b_P[i][j]$
9. Solve the set of equations (using Gauss-Seidel 3.3)
10. $max(|P^{n+1}[i][j] - P*^{n+1}[i][j]|) < \delta)$?
    - If the answer is **Yes**. Go to 11
    - If the answer is **No**. $P*^{n+1}[i][j] = P^{n+1}[i][j]$ and go to step 9
11. Correction of velocity at time instant $t^{n+1} \Rightarrow v^{n+1}$
12. Temperature at time instant $t^{n+1} \Rightarrow T^{n+1}$
13. It is $|v^{n+1} - v^n| < \delta$ and $|T^{n+1} - T^n| < \delta$
    - If the answer is **Yes**, go to step 14
    - If the answer is **No**, $t = t + \Delta t, P* = P^{n+1}, v^n = v^{n+1}, T^n = T^{n+1}$ and $R(v^n) = R(v^{n+1})$
14. Show results and conclusions

Notice that every time that we do a step-time it has to be calculated again the delta t, according to section 6.4. That is the reason why in the section of results, this is section 6.8 of this chapter, there will be different simulation times for different Reynolds numbers.

On the other hand, the only resolution scheme used in this case is the CDS because it was the faster and the most accurate one (comparing the computation time with the accuracy shown in other sections before).

6.6 Lid-Driven Cavity

The well-known Lid-Driven Cavity, a popular code verification tool, was the first issue to be resolved. To verify this code reference value used can be found in [15].

The problem consists of a square domain of dimensions LxL with solid walls at the other boundaries and at the upper wall that moves with a constant horizontal velocity, u_{ref}. The case is depicted in the picture below.

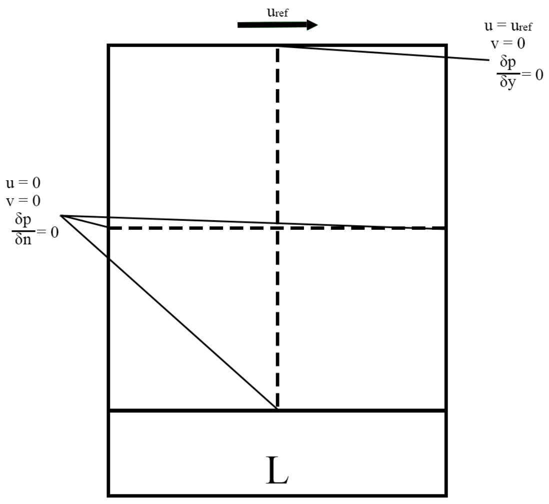


Figure 6.4: Lid-driven cavity boundary and initial conditions

6.7 Resolution scheme

The resolution scheme is comparable to the theoretical explained in section 6.5. Boundary conditions, nevertheless, must be considered for this case differently.

$$walls : u = v = 0, \frac{\partial P}{\partial n} = 0 \tag{6.18}$$

$$upperwalls : u = u_{ref}, v = 0, \frac{\partial P}{\partial y} = 0 \tag{6.19}$$

The Reynolds number [16], which is the ratio of inertial and viscous forces, is the parameter that describes the flow behavior. The definition of the non-dimension magnitude [16] is

$$Re = \frac{\rho L u_{ref}}{\mu} \tag{6.20}$$

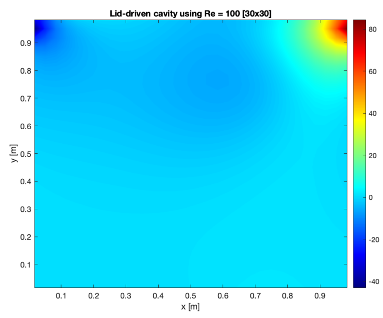
where ρ is the density of the fluid, L is the reference dimension of the particular case, u_{ref} is the characteristic velocity, μ the kinematic viscosity, and ν the dynamic viscosity of a fluid. It L, μ and u_{ref} are all set to 1 in the simulations that were run in this thesis, and the value of ρ is calculated using equation 6.20 and the given Reynolds number.

6.8 Results

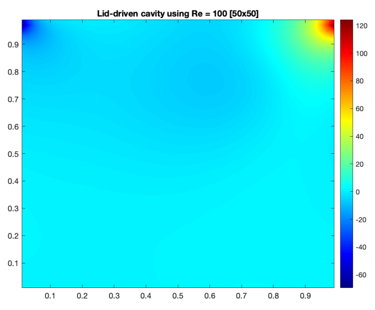
In this section, the results of this problem will be analyzed. This is essential to the report due to the first contact with Navier Stokes's problem. For this case, different scenarios have been shown with different numbers of elements to discuss when the solution is the most accurate with a reasonable computing time.

In addition, as mentioned before, the developed code is only using the numerical scheme CDS, which is explained in 5.2.2.

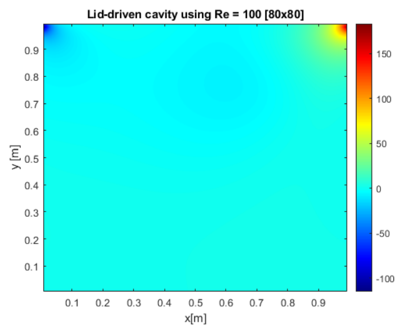
For case Re = 100,



(a) Mesh 30x30

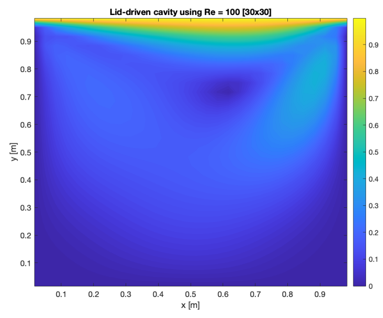


(b) Mesh 50x50

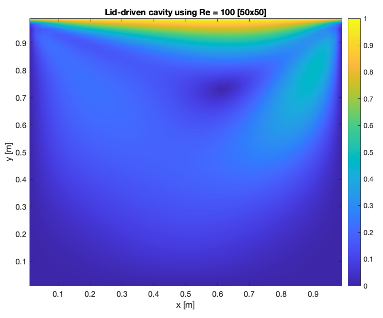


(c) Mesh 80x80

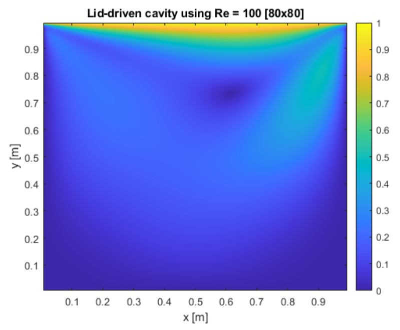
Figure 6.5: Lid-driven cavity using Re = 100 pressure distribution



(a) Mesh 30x30



(b) Mesh 50x50



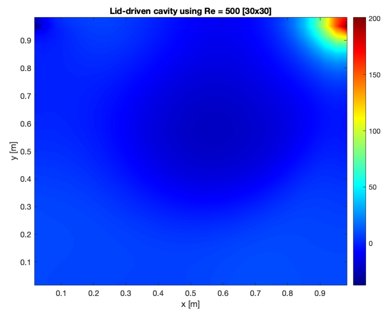
(c) Mesh 80x80

Figure 6.6: Lid-driven cavity using Re = 100 average velocity distribution

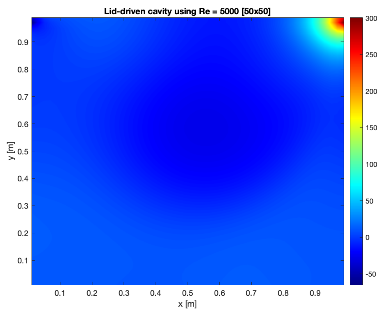
It is observable that the behavior of the pressure distribution is similar in the three of them. But as can be noticed, the pressure increases when the number of elements increases too. That is because when growing the number of the control volume, VC, the solution becomes more accurate to the exact solution.

On the other hand, the average velocity does not seem to change a lot during the increment of elements. It looks similar in shape and in value. In the verification section, it will be analyzed if this is true or if it is because as it has a little value, it varies from 0 to 1, and the error is not appreciable in the color-map distribution.

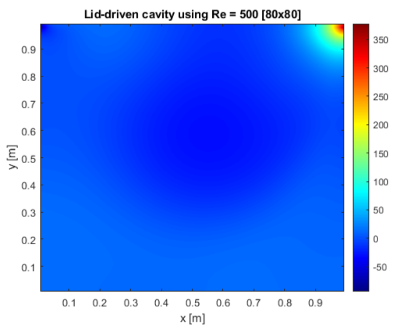
For Re = 500,



(a) Mesh 30x30

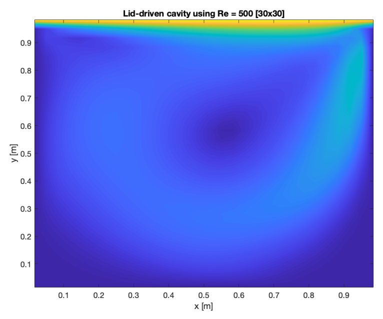


(b) Mesh 50x50

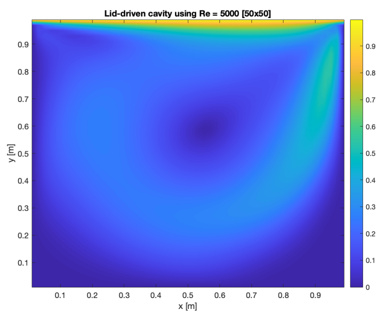


(c) Mesh 80x80

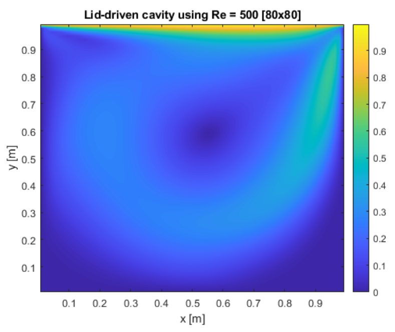
Figure 6.7: Lid-driven cavity using Re = 500 pressure distribution



(a) Mesh 30x30



(b) Mesh 50x50



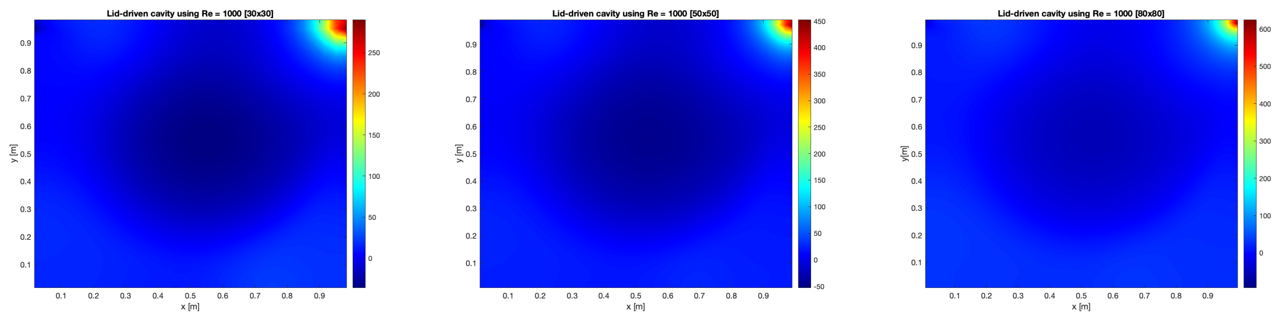
(c) Mesh 80x80

Figure 6.8: Lid-driven cavity using Re = 500 average velocity distribution

In case Re = 500 the pressure distribution looks a little bit darker than in the Re = 100 because the pressure distribution is huge when augmenting the Reynolds number, Re. However, the difference between the meshes is quite similar to the one previously seen with lower Reynolds.

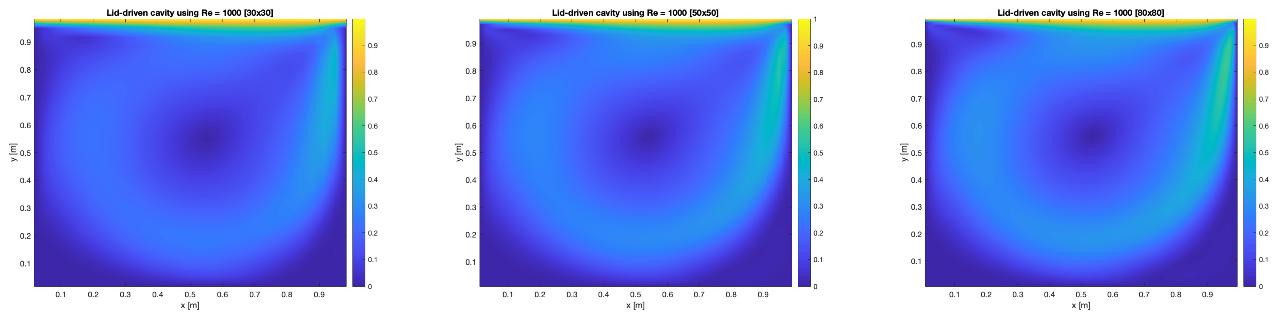
Secondly, in the velocity distribution, there is a difference between the previous one and this one, the lower corners of the map have become darker and the makes like a drop form with the initial point in the top right corner. The part that is common in all the graphics is that the highest velocity is it found on the top of the map because there is the initial condition $u_{ref} = 0$.

For Re = 1000,



(a) Mesh 30x30 (b) Mesh 50x50 (c) Mesh 80x80

Figure 6.9: Lid-driven cavity using Re = 1000 pressure distribution



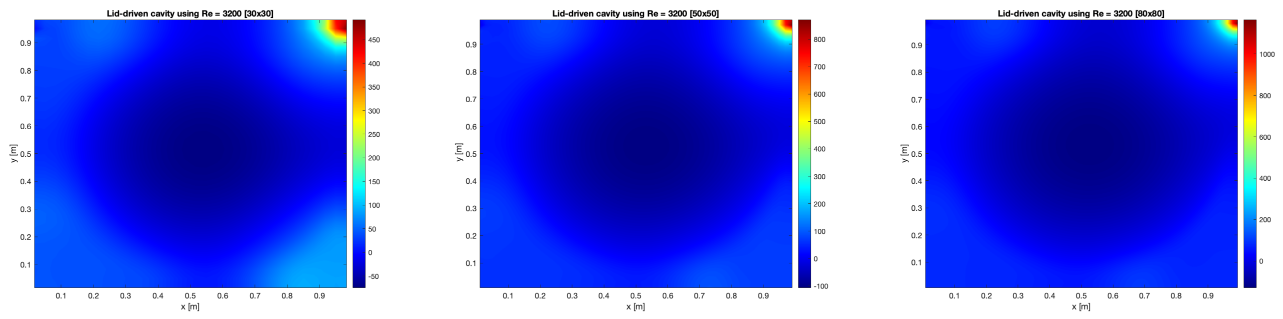
(a) Mesh 30x30 (b) Mesh 50x50 (c) Mesh 80x80

Figure 6.10: Lid-driven cavity using Re = 1000 average velocity distribution

In this scenario, it can be appreciated it a darker hole in the middle of the pressure distribution. In this hole, the pressure has a negative magnitude. The pressure distribution looks similar to the other ones but has the highest values on the corners.

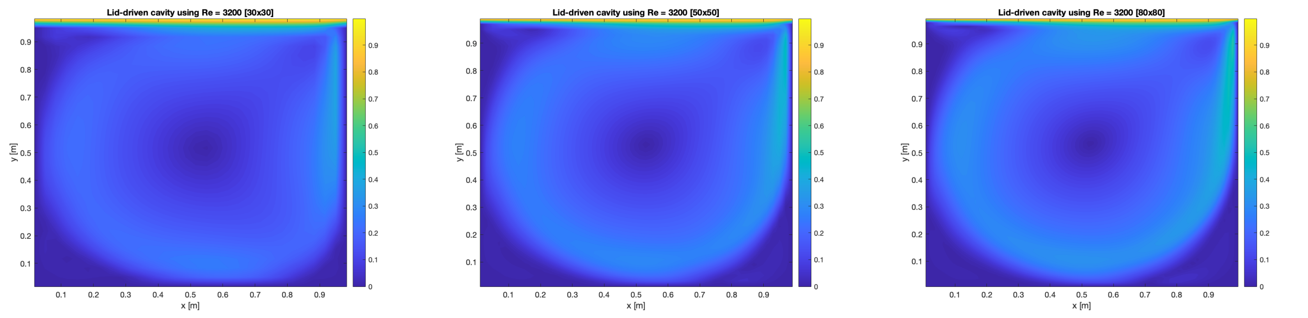
Looking now into the velocity distribution is quite similar to the distribution seen before, with Re = 500 but it looks like the form of a drop is more intensified in this case.

For Re = 3200,



(a) Mesh 30x30 (b) Mesh 50x50 (c) Mesh 80x80

Figure 6.11: Lid-driven cavity using Re = 1000 pressure distribution



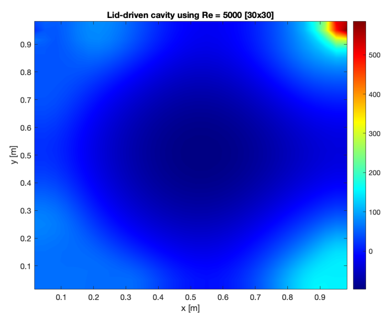
(a) Mesh 30x30 (b) Mesh 50x50 (c) Mesh 80x80

Figure 6.12: Lid-driven cavity using Re = 3200 average velocity distribution

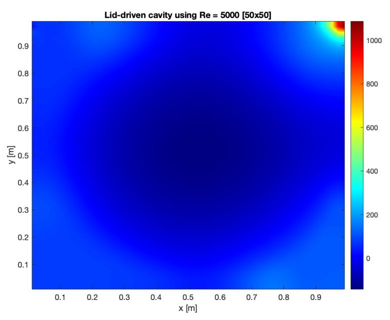
The lower value follows the greatest pressure in increasing with the number of Reynolds (in absolute value). Moreover, the hole in the middle has become bigger and darker than before meaning the pressure gradient increases with the number of Reynolds, which is obvious because the fluid becomes more turbulent.

Nevertheless, the velocity distribution looks similar to the previous one only the corners have become darker, which means the velocity in these points are more close to 0 than to 1 (as opposed to the velocity distribution when the Reynolds is 100).

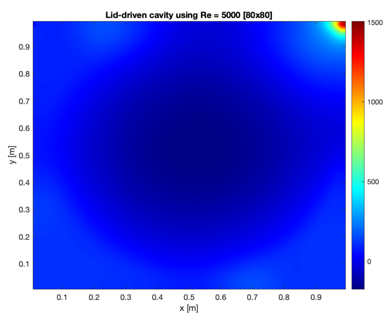
For Re = 5000,



(a) Mesh 30x30

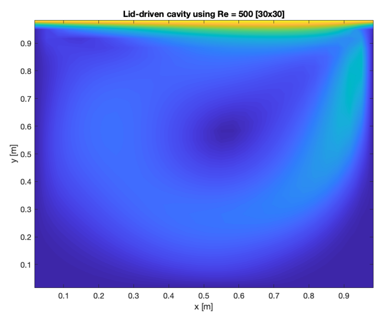


(b) Mesh 50x50

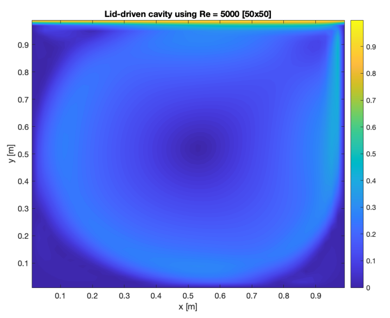


(c) Mesh 80x80

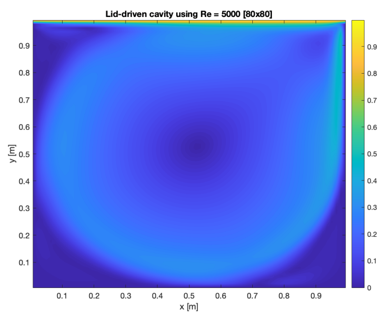
Figure 6.13: Lid-driven cavity using Re = 5000 pressure distribution



(a) Mesh 30x30



(b) Mesh 50x50



(c) Mesh 80x80

Figure 6.14: Lid-driven cavity using Re = 5000 average velocity distribution

Finally, the last simulation is Re = 5000 this is when the fluid is turbulent. The hole has become greater than before but the pressure distribution map looks quite similar to the previous one.

In the velocity distribution, there is a big difference between the mesh with 30x30 and the mesh with 50x50. The corners are more defined than in the other Reynolds when the mesh has more elements in it. That is because, as has been said before, the approximate solution would have less error while the elements inside the mesh increase (because it calculates more points in the domain with less distance between them).

6.8.1 Computation and simulation time

In order to obtain a better analysis of this case, which is the last one, a table of computation and simulation time is written.

	30x30		50x50		80x80	
Reynolds	**Execution time [s]**	**Simulation time [s]**	**Execution time [s]**	**Simulation time [s]**	**Execution time [s]**	**Simulation time [s]**
100	111.463484	18.7244	325.814569	18.4021	4104.698123	17.2469
500	436.683136	67.4193	1085.195047	61.8790	6596.886128	58.1494
1000	618.805340	141.3691	2344.949194	132.9661	20223.021224	123.4901
3200	752.546323	486.4325	8477.770128	479.0619	59631.826588	463.5068
5000	1189.68454	773.5674	13429.944329	771.0469	955678.301642	747.9657

Table 6.1: Simulation and computation time for different Reynolds and elements

The simulation times for the same Reynolds are pretty comparable among them, as can be seen in this table. However, it is noted that the execution time increases while the simulation time lowers when the number of components is increased.

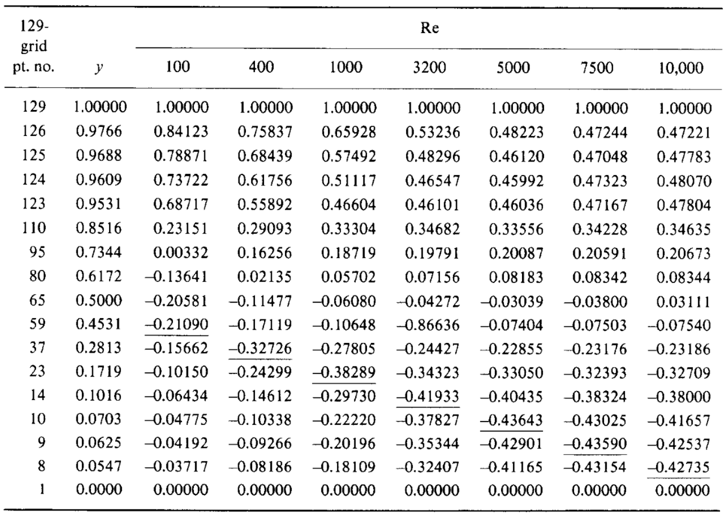
Moreover, the highest simulation is when the Reynolds is 5000 which makes sense because it is the simulation with more turbulence in it. Take note that the execution time of the highest mesh with this Reynolds is extremely large (more than 20h). The following part will evaluate the various outcomes and determine if it is profitable to use this mesh rather than the other one with fewer components.

6.9 Verification of the case

To ensure that the code and the solutions are correct the CTTC has provided a few keys to prove it. These are the horizontal velocity, u, in the middle vertical symmetry plane, the vertical velocity, v, in the middle horizontal symmetry plane, and, finally, the velocities distribution map.

6.9.1 Comparison of u-velocity

First of all, it is going to be compared to the horizontal velocity, u, in the middle vertical plane. The solution provided is:



129-grid pt. no.	y	Re						
		100	400	1000	3200	5000	7500	10,000
129	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
126	0.9766	0.84123	0.75837	0.65928	0.53236	0.48223	0.47244	0.47221
125	0.9688	0.78871	0.68439	0.57492	0.48296	0.46120	0.47048	0.47783
124	0.9609	0.73722	0.61756	0.51117	0.46547	0.45992	0.47323	0.48070
123	0.9531	0.68717	0.55892	0.46604	0.46101	0.46036	0.47167	0.47804
110	0.8516	0.23151	0.29093	0.33304	0.34682	0.33556	0.34228	0.34635
95	0.7344	0.00332	0.16256	0.18719	0.19791	0.20087	0.20591	0.20673
80	0.6172	−0.13641	0.02135	0.05702	0.07156	0.08183	0.08342	0.08344
65	0.5000	−0.20581	−0.11477	−0.06080	−0.04272	−0.03039	−0.03800	0.03111
59	0.4531	<u>−0.21090</u>	−0.17119	−0.10648	−0.86636	−0.07404	−0.07503	−0.07540
37	0.2813	−0.15662	<u>−0.32726</u>	−0.27805	−0.24427	−0.22855	−0.23176	−0.23186
23	0.1719	−0.10150	−0.24299	<u>−0.38289</u>	−0.34323	−0.33050	−0.32393	−0.32709
14	0.1016	−0.06434	−0.14612	−0.29730	<u>−0.41933</u>	−0.40435	−0.38324	−0.38000
10	0.0703	−0.04775	−0.10338	−0.22220	−0.37827	<u>−0.43643</u>	−0.43025	−0.41657
9	0.0625	−0.04192	−0.09266	−0.20196	−0.35344	−0.42901	<u>−0.43590</u>	−0.42537
8	0.0547	−0.03717	−0.08186	−0.18109	−0.32407	−0.41165	−0.43154	<u>−0.42735</u>
1	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Figure 6.15: Results for u-velocity along Horizontal Line through Geometric Center of Cavity [17]

It is going to be compared with the solutions of Meshes 50x50 and 80x80 to analyze if it is necessary to compute a long mesh with a computation time very high. It will be only compared the Reynolds number:

$$Re = \{100, 3200, 5000\}$$

The next table represents the values for different longitudes with a mesh of 50x50.

Can be seen in table 6.2 that the results have a similar tendency as in table 6.16 but there are not equal. Then it has been compared with the mesh of 80x80.

50x50	**Reynolds**		
y	**500**	**3200**	**5000**
1	1	1	1
0.97	0.75869400	0.506790675	0.4557936132
0.95	0.55796524	0.310840141	0.2854460755
0.85	0.25627278	0.289315908	0.2329639937
0.73	0.15132344	0.1605242325	0.1366906005
0.61	0.029122152	0.056411956	0.05601043530
0.51	-0.068998915	-0.010972314	-0.0057917317
0.17	-0.217850127	-0.247857517	-0.2204099633
0.07	-0.146833009	-0.232829989	-0.2747127368
0.05	-0.12127678	-0.1983718204	-0.2872547447
0	0	0	0

Table 6.2: Results for u-velocity with a Mesh of 50x50

To find better the errors of the solutions are correct it will be chosen the same y longitudes as the mesh with 50x50 elements.

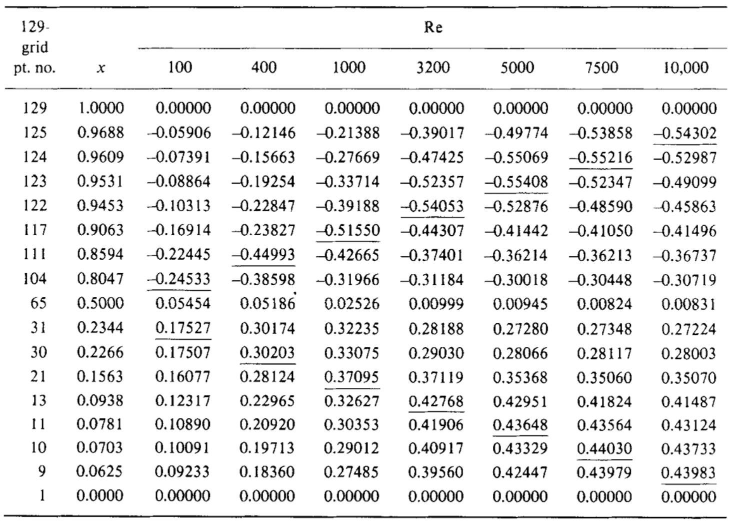
80x80	**Reynolds**		
y	**500**	**3200**	**5000**
1	1	1	1
0.97	0.82428195	0.6117565229	0.546524965
0.95	0.53127585	0.3312280389	0.312607150
0.85	0.27749957	0.269708935	0.26789153
0.73	0.17705773	0.173116374	0.16465405
0.61	0.05562248	0.079167669	0.06728023
0.51	-0.06135623	-0.0063083312	-0.0019548
0.17	-0.25572732	-0.2455674278	-0.2253510
0.07	-0.13201886	-0.3170262233	-0.3119490
0.05	-0.09194626	-0.2740962611	-0.29667994
0	0	0	0

Table 6.3: Results for u-velocity with a Mesh of 80x80

Comparing all the results it can be analyzed that the Reynolds that adjusted better to the theoretical one is Re = 500 because it represents laminar flow, which is way easier to represent than critical (Re = 3200) or turbulent flow (Re = 5000).

6.9.2 Comparison of the v-velocity

This section will be comparing the vertical velocity, v, in the middle of the horizontal plane with the solution provided in [17]. The figure above shows these results:



129-grid pt. no.	x	Re						
		100	400	1000	3200	5000	7500	10,000
129	1.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
125	0.9688	–0.05906	–0.12146	–0.21388	–0.39017	–0.49774	–0.53858	–0.54302
124	0.9609	–0.07391	–0.15663	–0.27669	–0.47425	–0.55069	–0.55216	–0.52987
123	0.9531	–0.08864	–0.19254	–0.33714	–0.52357	–0.55408	–0.52347	–0.49099
122	0.9453	–0.10313	–0.22847	–0.39188	–0.54053	–0.52876	–0.48590	–0.45863
117	0.9063	–0.16914	–0.23827	–0.51550	–0.44307	–0.41442	–0.41050	–0.41496
111	0.8594	–0.22445	–0.44993	–0.42665	–0.37401	–0.36214	–0.36213	–0.36737
104	0.8047	–0.24533	–0.38598	–0.31966	–0.31184	–0.30018	–0.30448	–0.30719
65	0.5000	0.05454	0.05186	0.02526	0.00999	0.00945	0.00824	0.00831
31	0.2344	0.17527	0.30174	0.32235	0.28188	0.27280	0.27348	0.27224
30	0.2266	0.17507	0.30203	0.33075	0.29030	0.28066	0.28117	0.28003
21	0.1563	0.16077	0.28124	0.37095	0.37119	0.35368	0.35060	0.35070
13	0.0938	0.12317	0.22965	0.32627	0.42768	0.42951	0.41824	0.41487
11	0.0781	0.10890	0.20920	0.30353	0.41906	0.43648	0.43564	0.43124
10	0.0703	0.10091	0.19713	0.29012	0.40917	0.43329	0.44030	0.43733
9	0.0625	0.09233	0.18360	0.27485	0.39560	0.42447	0.43979	0.43983
1	0.0000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

Figure 6.16: Results for v-velocity along Horizontal Line through Geometric Center of Cavity [17]

It has been chosen the same Reynolds numbers as the previous section and the same longitudes but for the x midplane. The table with the values of the mesh 50x50 is shown.

80x80	**Reynolds**		
y	**500**	**3200**	**5000**
1	0	0	0
0.97	-0.064330228	-0.11573234	-0.1258101
0.95	-0.115192113	-0.18455383	-0.1909330
0.85	-0.23699306	-0.27029152	-0.2475620
0.73	-0.25460535	-0.18095446	-0.16430443
0.61	-0.16574615	-0.09709777	-0.08852608
0.51	-0.06071183	-0.02873286	-0.02653547
0.17	0.36266439	0.231990876	0.216332578
0.07	0.40487757	0.29044414	0.268813189
0.05	0.36653328	0.327800959	0.291896858
0	0	0	0

Table 6.4: Results for v-velocity with a Mesh of 80x80

For mesh 80x80 the table with the different values and Reynolds numbers are been observed below.

80x80	**Reynolds**		
y	**500**	**3200**	**5000**
1	0	0	0
0.97	-0.0518212962	-0.104418391	-0.1170304474
0.95	-0.1318012550	-0.2236081943	-0.2329526207
0.85	-0.245628658	-0.3052830878	-0.2836842118
0.73	-0.275061657	-0.2110332427	-0.1936024852
0.61	-0.195588477	-0.1219338344	-0.1119369507
0.51	-0.070661762	-0.0353065225	-0.0323486143
0.17	0.336316838	0.23448872789	0.2175189219
0.07	0.386041440	0.34789276785	0.3100580213
0.05	0.2777698875	0.42421716328	0.3733983346
0	0	0	0

Table 6.5: Results for v-velocity with a Mesh of 50x50

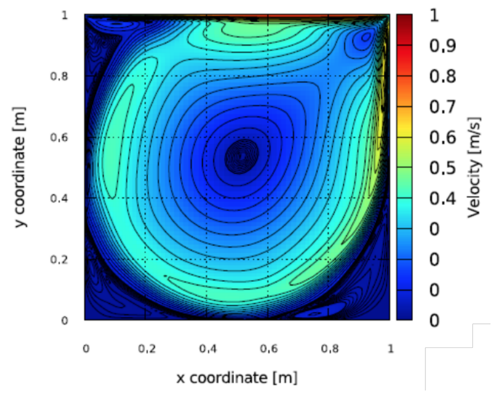
The approximation of the v-velocity is quite similar to the u-velocity and with these tables can be verified that the code is correct even though to be more accurate the number of elements should have been bigger (more or less 130x130).

On the other hand, increasing the number of elements increments exponentially the computation time which is too high and only with a mesh of 80x80 elements.

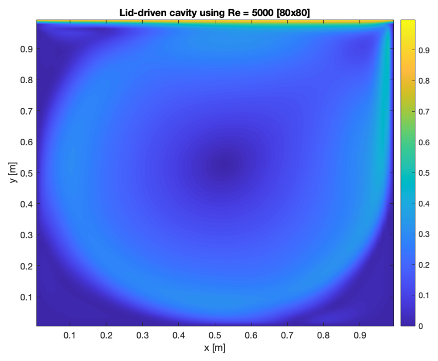
Nevertheless, the difference in results between the two meshes is appreciable so it is completely worth the computation time (so much worth it talking about the pressure distribution which has a huge difference because the numbers are much bigger).

6.9.3 Velocities distribution Map

For this case will be used the reference of [13] which has a figure of streamlines distribution for this special problem when using Re = 5000.



(a) Theoretical velocity distribution [13]



(b) Experimental velocity distribution

Figure 6.17: Comparison of the experimental with the theoretical one lid-driven cavity

The velocity distribution looks quite similar between the two figures above, even the magnitude of the velocities inside the map. The two of them have a drop form and three of the four corners with low velocities (minor of 0.1).

Moreover, in the case of the map the mesh of 80x80, it shows a good resolution of the problem and there is no need of using a higher number of elements (in case of looking for an accurate solution talking about the values of this velocity is better to rise this number of elements).

It is interesting that the maps are quite similar because when putting a high number of Reynolds it becomes a turbulent flow, which is less predictable and more difficult to study, which means, that the methods used to solve this case are good for finding the answer to many Navier-Stokes problems.

6.10 Summary

In this section, the Fractional Step Method has been used to perform a Lid-driven cavity problem.

First of all, it has been studied the pressure distribution along different numbers of Reynolds. From this analysis, it can be said that the pressure in the middle of the domain becomes more strong and negative as the number of Reynolds increases.

In addition, it can also be seen that the pressure gradient becomes bigger too because the flow goes from a laminar to a turbulent form.

On the other hand, the average velocity behavior looks quite similar between Re = 500 and Re = 5000 looking only into the color-map velocity distribution. However, if the values of each number of Reynolds have been observed there is a huge difference between them at some points of the mesh.

Moreover, as mentioned before, the solutions for the velocity are more accurate when talking about lower Reynolds numbers that are because it is a laminar flow (which is more predictable and easy to understand).

Chapter 7

Future considerations

The goal of this thesis was to learn more about the Navier-Stokes equations. However, only one case has been managed that is directly related to these equations. This is because there was a lot of theory to learn before going straight into these equations with no analytical solution, for example, getting the terms convection and conduction right.

Also, this report was started quite late, at the end of February, due to certain problems with the programming language, C++, which finally led to the report being made entirely with Matlab.

For future considerations, A new objective of the project would have been done in a previous C++ course in order to have more time to carry out different cases, and not so much for code problems.

Moreover, C++ is much faster at computing complex codes than Matlab which leads to less computation time and more time to do different cases.

On the other hand, talking about the cases done in this report it can also have improvements, for instance,

- To make bigger meshes, meaning more elements, to make the solutions more accurate.
- To optimize better the code, especially, the Lid-driven cavity for taking less computation time.
- Taking into account turbulence considerations. All the codes that have been developed compute more inaccurate results when the Reynolds number is higher.
- All the code that has been developed considers the flow as incompressible. When having a high fluid speed the results are more accurate if there is a consideration of the density variation.

Chapter 8

Budget summary and Environmental implications

The total budget can be checked in the document called "Budget". The report mentioned has divided the budget into three parts: manpower costs, equipment costs, and electricity costs.

Finally, the total cost of this Bachelor's Final Thesis is **7969.5 €**.

Talking about the environmental implication of this project is only one, the emissions of CO_2 due to energy consumption. As seen in the document "Budget" the essential resource used is Energy, to do the simulations, write the report, to develop the code in MatLab...

In this case, the building used to develop the codes has solar panels which means the emissions of CO_2 are zero, assuming only the hours when the sun does not go down.

On the other hand, to develop the simulations have been done on another type of computer, with more capacity, and this computer was situated in another building which has no solar panels. In Spain, the CO_2 emissions per kWh are $273gCO2/kWh$ [18].

Then, as a summary, and assuming a 20% of the work referring to the development of the codes has been done during night hours,

	Task	**Consumption [kWh]**	**Emissions [gCO_2]**
	Codes development and others	10.8	2948.4
	Simulations of the cases	36	9828
TOTAL		46.8	**12776.4**

Table 8.1: Environmental implications

As can be seen in the table above the total emissions to do this report is 12.77kg CO_2 which is a huge quantity.

Chapter 9

Conclusions

The project's goal was to create MatLab codes for solving the incompressible Navier-Stokes equations. Nevertheless, additional examples were carried out to build expertise before trying the main aim of this thesis, such as the heat conduction cases or the convection-diffusion cases. Talking about the terms of the scope, it can be said that all the goals, except the application of the codes to a particular aeronautical problem, have been done.

On the other hand, a huge number of findings have been realized during the development of this thesis.

First off, it has been noted that many variables have influenced the quality of the results as has been mentioned during the report. For instance, the number of elements and the time in the problems have a significant influence on the solutions. The more elements, the more accurate the problem, but the computation time increases exponentially and, in a huger number of cases, the computation time was too long to have a sufficiently exact solution. That was one of the main reasons why the number of elements in a lot of the cases inside this report has been reduced, the same happens with the time step.

Secondly, the numerical schemes. The low-order schemes have less accurate solutions but less computational time. In the end, it has had to be constantly considering whether it was better to invest more computing time for a more exact solution. However, in many cases, the trend of the solution was already correct and can give you an idea of the behavior of the problem without having such an approximate solution but with a much shorter computation time, as in the case of Lid-driven cavity that Only CDS has been used and the velocity distribution for a turbulent Reynolds already behaved well even with a low-resolution scheme and a mesh with relatively few elements.

Thirdly, the influence of the non-dimensional parameters. For instance, the number of Reynolds influenced in the resolution and the computation time of a problem. For a high number of Reynolds, when the flow becomes turbulent, is less predictable and difficult to have an accurate solution, even though the computation times in this case was more bigger than in other cases.

In conclusion, this work has served to better develop the concepts of the Navier-Stokes equations and the long and complex procedures that have to be followed in order to simply have a non-exact solution to these equations.

Bibliography

1. *What Are Navier-Stokes Equations? Documentation*. 2023. Available also from: `https://www.simscale.com/docs/simwiki/numerics-background/what-are-the-navier-stokes-equations/`. Retrieved on 25th February 2023.
2. NASA. *Navier-Stokes Equations*. [N.d.]. Available also from: `https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html`. Retrieved on 20th February 2023.
3. MORA, Xavier. Navier-Stokes equations, unpredictability even without butterflies? *Mètode Science Studies Journal*. 2018, pp. 43–49. Available from DOI: `https://doi.org/10.7203/metode.8.9415`. Retrieved on 5th March 2023.
4. MOENG, C. H.; SULLIVAN, Patrick F. NUMERICAL MODELS — Large-Eddy Simulation. *Elsevier eBooks*. 2015, pp. 232–240. Available from DOI: `https://doi.org/10.1016/b978-0-12-382225-3.00201-2`. Retrieved 27th February 2023.
5. OHKITANI, Koji. Characterization of blowup for the Navier-Stokes equations using vector potentials. *AIP Advances 7*. 2017. Available from DOI: `https://doi.org/10.1063/1.4975406`. Retrieved on 27th February 2023.
6. HIESTER, H.R.; PIGGOTT, Matthew; FARRELL, P.E.; ALLISON, P. Assessment of spurious mixing in adaptive mesh simulations of the two-dimensional lock-exchange. *Ocean Modelling*. 2014, vol. 73, pp. 30–44. Available from DOI: `https://doi.org/10.1016/j.ocemod.2013.10.003`. Retrieved on 3rd March 2023.
7. ALI HASAN ALI Ahmed Shawki Jaber, Mustafa T. Yaseen. A Comparison of Finite Difference and Finite Volume Methods with Numerical Simulations: Burgers Equation Model. *Hindawi*. 2022, vol. 2022. Available from DOI: `https://doi.org/10.1155/2022/9367638`. Retrieved on 7th March 2023.
8. MARIO STORTI Norberto Nigro, Rodrigo Paz. Dynamic boundary conditions in computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*. 2007, vol. 197. Available from DOI: `https://doi.org/10.1016/j.cma.2007.10.014`. Retrieved on 7th March 2023.
9. CTTC. *Numerical resolution of the generic convection-diffusion equation*. 2022. Course on Numerical Methods in Heat Transfer and Fluid Dynamics. Retrieved on 15th March 2023.
10. CTTC. *Numerical Solution for Convection*. 2022. Retrieved on 20th March 2023.
11. M.S. DARWISH, F.H. Moukalled. Normalized variable and space formulation methodology for high-resolution schemes. *Numerical Heat transfer*. 1994, vol. 26, pp. 79–96. Available from DOI: `https://doi.org/10.1080/10407799408914918`. Retrieved on 17th May 2023.

12. PATANKAR, Suhas V. *Numerical heat transfer and fluid flow*. Hemisphere Publishing Corporation, 2018. Series in Computational Methods in Mechanics and Thermal Science. Retrieved on 21th March 2023.
13. CTTC. *Numerical resolution of the Navier Stokes equations using the Fractional Step Method*. 2022. Course on Numerical Methods in Heat Transfer and Fluid Dynamics. Retrieved on 20th April 2023.
14. SHABBUSHARMA. *What is Gauss Divergence theorem? State and Prove Gauss Divergence Theorem*. 2021. Available also from: `https://physicswave.com/gauss-divergence-theorem/`. Retrieved on 3rd May 2023.
15. CTTC. *Fractional Step Method: Staggered and Collected Meshes*. 2022. Numerical Methods in Heat Transfer and Fluid Dynamics. Retrieved on 5th May 2023.
16. RAPP, Bastian E. *Elsevier eBooks*. Fluids. 2017. Available from DOI: `10.1016/b978-1-4557-3141-1.50009-5`.
17. U-GHIA, K.N. Guia; SHIN, C.T. *High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and Multigrid Method*. University of Cincinnati, Ohio 45221, 1982. Journal of Computational Physics. Retrieved on 20th May 2023.
18. *Factor de emisión de la energía eléctrica: el mix eléctrico*. [N.d.]. Available also from: `https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/index.html`. Retrieved on 10th June 2023.