Study for the numerical resolution of conservation equations of mass, momentum and energy to be applied to solar thermal collectors

Bachelor's thesis

Author

Daniel Yago Llamas

Director

Carlos David Pérez Segarra

Co-director

Francesc Xavier Trias Miquel

Bachelor's Degree in Aerospace Technology Engineering

Universitat Politècnica de Catalunya

Escola Tècnica Superior d'Enginyeries Industrial i Aeronàutica de Terrassa

June, 2015

Agradecimientos

Quisiera agradecer el apoyo que he recibido por parte de los diferentes profesores e investigadores que conforman el CTTC (Centre Tecnològic de Transferència de Calor i Massa) los cuales me han proporcionado apoyo en el aspecto técnico del proyecto con el asesoramiento y resolución de dudas permitiendo así la finalización del TFG.

En especial, querría agradecer el apoyo mostrado por los profesores y doctores Assensi Oliva Llena y Carles David Pérez Segarra, así como a los investigadores Jordi Chiva y Francesc Xavier Trias Miquel, los cuales han estado siempre dispuestos a ayudar.

También me gustaría recalcar el apoyo recibido por parte de mi familia a lo largo de la realización de dicho proyecto, haciendo más ameno los momentos más difíciles.

Por último, recalcar la ayuda ofrecida por mi compañero Josep Molins Farrés en la revisión y corrección del trabajo, permitiendo ofrecer un trabajo de calidad.

Resumen

Este proyecto tiene como objetivo la creación de un software en lenguaje C++ que permita resolver las ecuaciones de Navier-Stokes para casos con geometrías sencillas y presuponiendo ciertas hipótesis, así como el estudio y comprensión de las ecuaciones que rigen la dinámica de fluidos. Aunque también se introduce la metodología aplicada a casos de conducción y radiación.

La finalidad principal es llegar a simular el comportamiento fluido-dinámico y térmico de casos ya conocidos y así verificar los resultados. Principalmente se estudiarán los problemas: “Smith-Hutton problem”, “Driven Cavity problem” así como “Differentially heated cavity problem”.

Finalmente se procede a introducir la fenomenología que caracteriza la turbulencia así como la implementación teórica y práctica de casos 1D.

Abstract

The aim of this project is to create a software in C++ language that allows to solve numerically the Navier-Stokes equations for cases with simple geometries and presupposing certain hypotheses, as well as the study and comprehension of the equations that govern the fluid dynamics. Furthermore, the procedure applied to cases of conduction and radiation is also introduced.

The main purpose is to simulate the fluid-dynamic and thermal behaviour of already well-known cases as a way to verify the results. Principally, The studied problems are: "Smith-Hutton problem”, “Driven Cavity problem” as well as “Differentially heated cavity problem”.

Finally, an introduction is done about the phenomenology that characterizes the turbulence as well as the theoretical and practical implementation of 1D cases.

Contents

List of Figures

List of Tables

Part I.

Introduction

1. Introduction

1.1. Objective

The main objective of this bachelor's thesis is the achievement of enough knowledge to be able to understand the physics involved in the CFD&HT (Computational fluid dynamics and heat transfer), and the development of a self-built CFD&HT code using the C++ language. At this point of the project, it will be possible to solve the Navier-Stokes equations for most of the cases by solving the laminar flow.

A numerical solution will be achieved for a practical case with the knowledge obtained about CFD&HT numerical simulation and the built code. This solution will be obtained effectively with few computational resources to be able to run this code in any laptop. After that, the codes will be validated to verify their thermophysical reliability from known conditions and cases or from experimental results and advanced simulations obtained by the CTTC or other researchers.

1.2. Scope

The tasks to complete the project are listed below:

- State of the Art study of numerical simulation in the field of fluid dynamics and heat and mass transfer.
- Study and comparison of different numerical methods capable of transforming systems of differential equations with partial derivatives into system of algebraic equations.
- Study of the equations that govern the heat conduction in solids with different materials and unsteady conditions.
- Study of the equations that govern the laminar convection of fluids with different boundary conditions and unsteady conditions.
- Study of the equations that govern the turbulent convection of fluids with different boundary conditions and unsteady conditions if there is enough time.

- Study of the equations that govern radiation in the infrared and solar range.
- Selection of a case that implements the physical equations that govern the CFD&HT in the field of thermal energy accumulators.
- Study of the equations that govern the physics of the selected case of study.
- Creation of an own code in C++ language.
- Verification and validation of the code from experimental results or advanced simulations made by researchers.
- Code optimization in C++.
- Obtention of the results of the selected case.
- Writing of possible improvements and future expansion.
- Writing of the project planning and budget.

1.3. Requirements

The requirements for the study are listed below:

- The programming language used to develop the code will be C++.
- A new non-commercial code is created.
- The Navier-Stokes equations will be solved with certain assumptions and simplifications.
- The mathematical method used to solve the differential equations is the Finite Volumes method.
- The domain and the geometries must be 2D.
- The mesh used for the discretization of the domain will be structured, and the possibility of applying an unstructured mesh.
- The software must be able to solve transitory cases.
- The case of study will be considered on a scale large enough to assume continuity hypothesis of matter.
- The flow must have constant physical properties.
- The flow is considered to be incompressible.
- The fluid must be pure and Newtonian.
- Turbulence will be considered if its implementation is possible.

1.4. Justification

In nowadays, industrial companies invest large amounts of money and time on prototyping to simulate and test their products in order to improve and optimize them. These products are tested in wind tunnels in the case for aerodynamic products or in laboratories with high-tech sensors to measure temperature and other physical properties for thermal products.

These procedures to test the products are, in some cases, too expensive for small businesses, which cannot afford them. Moreover, the computational power of today's computers has suffered an abrupt improvement. For that reason, CFD&HT programs are increasing in strength as they are a viable and cheaper option for companies.

These CFD software allow to solve complex cases with different geometries, although in some cases they may not be efficient enough or optimized for a specific use. In these cases, specific codes are developed for each particular case in order to solve the problem efficiently and obtain precise results.

A self-built software code was chosen because the project's purpose is not to solve any complex case, whatever the complexity, but to truly understand the physics and phenomenology of fluid mechanics, mathematics and programming behind the CFD&HT software. This option allows to identify the appropriateness and usefulness of solving the equations of conservation of energy, mass and momentum (Navier-Stokes equations) in the field of aerodynamics and heat transfer.

About this area in particular, there will be an application or analysis based on own code developed, although if it is too exhaustive for the duration of the bachelor's thesis, a prospective and planning on how the project could be expanded to go in depth into the theme will be done.

1.5. Background

Scientists have been interested in fluid mechanics from antiquity to ancient Greece, the Roman Empire or even the first inventors "engineers" of the Renaissance. But it was not until the 17th century, when Isaac Newton (1643-1727) developed the Newtonian physics equations, which attempted to predict fluid flow by determining the concept of Newtonian viscosity.

Later, in the 18th century, the mathematical model describing the flow of fluid was improved thanks to the contributions of Daniel Bernoulli (1700-1782) with the equation of Bernoulli and Leonhard Euler (1707-1783) with the Euler equations, which described conservation of mass and momentum for inviscid fluids.

Finally, in the 19th century, the Navier-Stokes equations were deduced independently by George Gabriel Stokes (1819-1903) in England and Henry Claude Louis Marie Navier (1785-1836) in France. The equations include the viscosity effects to the Euler equations which describe how the speed (three components), pressure, temperature and density are related to the fluid motion.

The Navier-Stokes system is a very complex system of differential equations that although it can be solved analytically for very simple cases using different mathematical methods, they require numerical methods for solving the approaches of this set of equations using techniques such as finite differences, finite volume and finite element among others. This field is called "Computational Fluid Dynamics" or commonly CFD.

The system of equations is formed by the equation of conservation of mass, which is time dependent, three equations of momentum conservation (one for each direction), time dependent, and a final equation of energy conservation, also time dependent. Finally, we must add a state equation of the fluid to solve the system of 6 equations and 6 unknowns.

Over the following centuries, this field has been investigated in order to incorporate the boundary layer phenomena and fluid turbulence. The most important researchers are Ludwig Prandtl (1875-1953), Theodore Von Karman (1881-1963), etc.

But it was not until the 50s, when the first computer numerical analysis was performed. Subsequently, NASA developed numerous numerical methods, which are currently used in CFD codes and software. Years later, commercial CFD programs went on sale, and it was at this point when companies and researchers invested in CFD software and computers or "clusters" to solve the equations numerically.

Currently, any personal computer is able to run CFD software in a short time by reducing the need for expensive laboratory experiments. It is for this reason that CFD software are increasingly more important for companies engaged in this area, even taking it to specialized departments. Still, at certain cases, larger computational capacity is needed to solve the phenomena of small-scale turbulence.

Part II.

Numerical analysis

2. Introduction to numerical methods

Computational fluid dynamics carries out solving flows and related phenomena which can be described by partial differential equations, and commonly they cannot be solved analytically. Thereupon, this science tries to obtain an approximated solution numerically by using discretization methods to approximate differential equations by a system of algebraic equations. These discretized equations are applied to small domains in space in order to obtain results at discrete locations in space over time.

It is true that CFD can deal with complex problems and obtain an accurate solution, but it must not be forgotten that it is only an approach because errors arise from each part of the numerical method. There are errors in the differential equations which may contain idealizations or approximations, in the discretization process, in the iterative solver, etc.

The numerical solution method consists of several components that are listed below [13]

- Mathematical Model: it is the most important part of a numerical method since the physical problem is simplified to a set of partial differential equations and boundary conditions. These equations are based on assumptions and simplifications in order to solve easily and realistically the target case. It will be explained in the different sections of heat transfer.

- Discretization method: this part approximates the differential equations by a system of algebraic equations for the variables at discrete locations in space and time. The most important approach methods are: finite difference (FD), finite volume (FV) and finite element (FE) methods. They almost achieve the same solution if the grid is very fine. The chosen method for this study will be a finite volume method.

- Coordinate and basis vector systems: the differential equations can be written in many different forms, depending on the coordinate system (Cartesian, cylindrical, spherical or curvilinear coordinate systems) and the basis vector used which may be fixed or moving. The chosen coordinate system is a fixed, Cartesian, orthogonal system.

- Numerical grid: this part defines the discrete locations at which the variables are to be calculated. Therefore, it divides the solution domain into small subdomains. The chosen numerical grid is a structured, orthogonal, uniform or non-uniform mesh. It will be explained in the following sections.
- Finite approximations: the approximations to be used in the discretization process must be selected depending on the type of grid and the discretization method: the approximation of the derivatives at the grid points for finite difference method, the approximation of volumetric and surface integrals, and the shape functions and weighting functions for finite element method. It will be explained in the different sections of this study.
- Solution method: a solver is needed to achieve a solution of the large system of non-linear algebraic equations for the discrete locations in space over time until the convergence is achieved.
- Convergence criteria: the accuracy of the solution depends on the convergence criteria. For that reason, it is important to specify when to stop the iterative process on each time step, and when the solution is steady.

2.1. Finite Volume Discretization

The FVM consists in the division of the domain in a finite number of control volumes which depends on the desired precision or the complexity of the problem to study. This method locates a node at the centroid of each control volume and assumes that the fluid has constant properties inside each control volume. In other words, it has the same pressure, temperature, density and velocity and all these values are assigned to the centre of the finite volume. For that reason, interpolation is used to express variable values at the control volume surfaces in terms of the nodal values.

2.1.1. Domain discretization

The FVM structural discretization of the domain for Cartesian coordinate system is shown in Figure 2.1.1. The control volume is shaded, and has a $\Delta x \cdot \Delta y$ dimension. For structured, uniform, orthogonal meshes, there are 4 neighbouring nodes (N, S, W and E) which are $(\delta x)_{nb}$ or $(\delta y)_{nb}$ from the central node.

The control volume faces e, w, n and s are placed between the grid point P and its neighbours. There is no need for the faces to be located right between them. Therefore, $(\delta x)_e$ and $(\delta x)_w$ can be different. The same happens with $(\delta x)_n$ and $(\delta x)_s$. This can be appreciated in Figure 2.1.2.

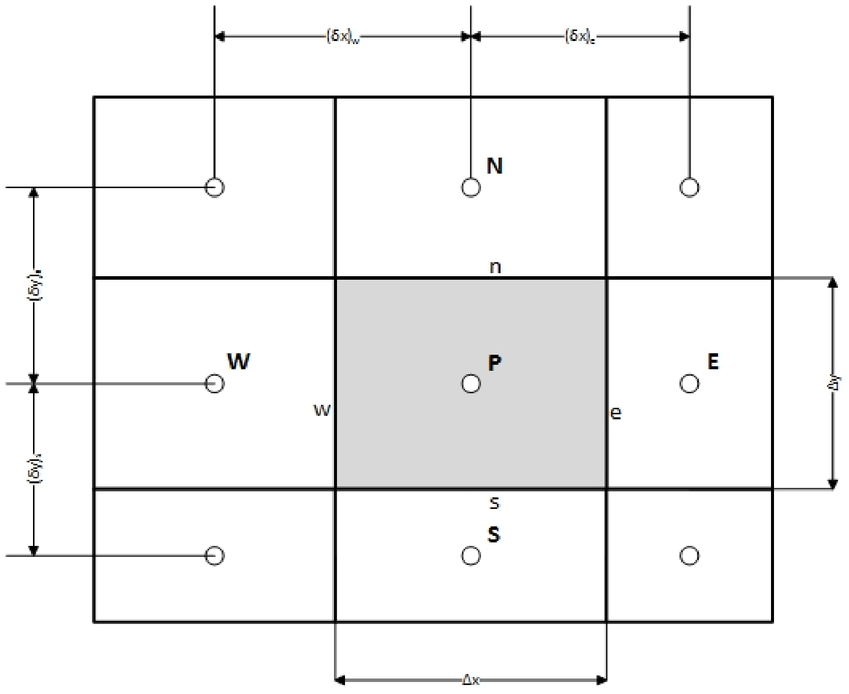


Figure 2.1.1.: Control volume for the two-dimensional situation.

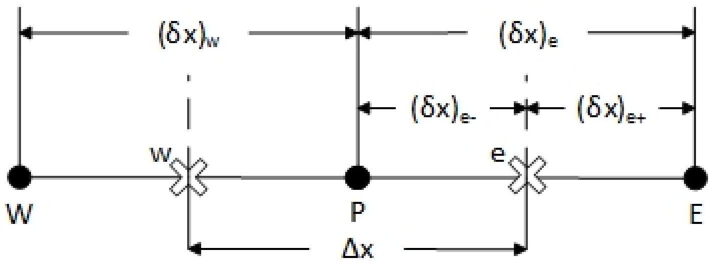


Figure 2.1.2.: Control volume for one-dimensional situation.

2.1.2. Mesh

There are two types of meshes according to the treatment for the convection-diffusion equation [13, 23]:

- Co-located: the convection-diffusion equation is evaluated in the same grid for all the ϕ variables at the centre of each grid cell. This type of mesh has a disadvantage because the velocity and the pressure values have to be referenced/interpolated to the central node of the cell. This can be appreciated in Figure 2.1.3.

- Staggered: the convection-diffusion equation is evaluated in a different grid for each ϕ variable. This method implies an advantage due to the fact that each variable is obtained at the point of interest to solve N-S Equations. Therefore, the pressure is calculated at the central node of the main grid, although the

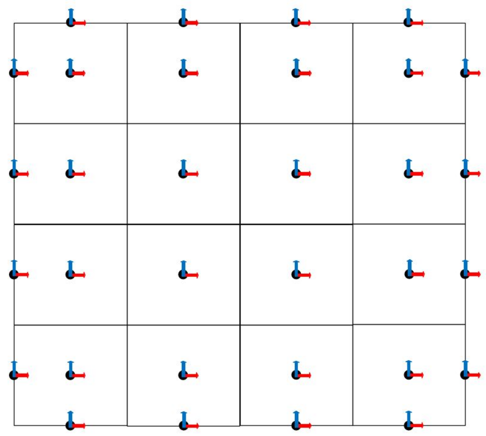


Figure 2.1.3.: Co-located mesh

velocity is calculated at the interface of the main grid with a displaced grid. For that reason, there are three different meshes: one for central node properties, another one for the x-axis velocity component, which is displaced in x direction, and another grid for the y-axis velocity component, which is displaced in y direction. This can be appreciated in Figure 2.1.4.

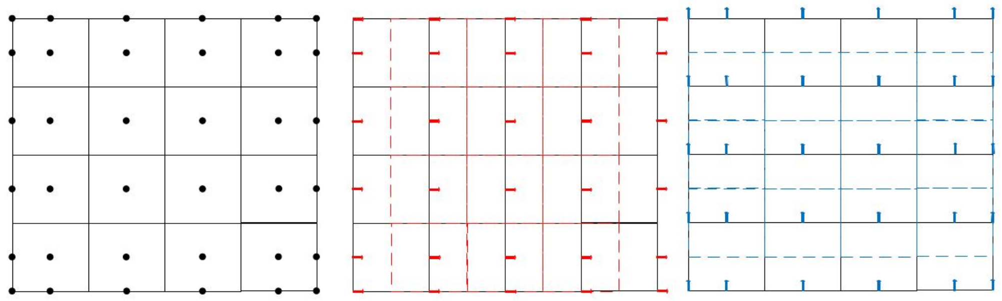


Figure 2.1.4.: Staggered mesh (central grid, x-staggered and y-staggered mesh)

The types of spatial meshes are listed below (for more information, see [23]) :

- Structured: this mesh is characterized by a regular connectivity for either two-dimensional array or three-dimensional array. This restricts the elemental choices to quadrilaterals in 2D and hexahedron in 3D. Structured meshes allow programming codes more efficiently than unstructured meshes, although they can only be used for geometrically simple solution domains.
    - Uniform: all elements have the same dimension, they are equally separated.
    - Non-Uniform: the distance between elements is variable.
    - Orthogonal: this mesh can be uniform or non-uniform, but it has all the interfaces orthogonal, which makes it easier to obtain the flow through the faces.

  · Non-orthogonal: this mesh fixes better to the boundary geometries and it can be uniform or non-uniform as well as the orthogonal mesh. In this case, the flow through the faces is not normal to the control volume's faces.

- Unstructured: this mesh is characterized by irregular connectivity. This type of mesh allows better adaptation to complex boundaries. However, the equations to solve the problems are more complex and it needs more computational requirements. The common used elements in unstructured meshes are triangles or quadrilaterals for 2D and tetrahedra or hexahedra in 3D. The solvers for the unstructured grids are slower than those for structured grids, because the matrices of the algebraic equation systems are no longer as regular or diagonal as structured matrices.
- Hybrid: this mesh contains structured portions and unstructured portions.

The meshes used in the project will be orthogonal, and uniform or non-uniform depending on each problem. However, unstructured meshes will be used if there is some extra or free time by the end of the bachelor's thesis.

2.2. Temporal discretization schemes

2.2.1. Explicit scheme

This scheme is used to obtain the temperature Tp^1 in state t+Δt only with t values, Tp^0, in a direct method due to the fact that f is equal to 0. No iteration is needed to obtain the nodal temperatures in the next time step, hence it is a faster method than the Crank-Nicolson scheme or the implicit scheme.

According to Scarborough criterion, every coefficient in Equation 3.7.1 must be positive. Nevertheless, the coefficient of Tp^0 could become negative when using explicit scheme, thus providing a wrong or unrealistic result. A minor step must be used to avoid it in conduction heat transfer.

$$\Delta t < \frac{\rho C_p (\Delta x)^2}{2\kappa} \tag{2.2.1}$$

Therefore, the time step must be necessarily small in order to increase the low stability that shows this scheme. On the other hand, this scheme requires a reduced computational costs.

2.2.2. Crank-Nicolson scheme

This scheme is used to obtain the temperature Tp^1 in state t+Δt given the t values, Tp^0, and t+Δt guess values, Tp^{1^*}, in an iterative method due to the fact that f is equal to 0.5. A linear variation between Tp^0 and Tp^1 is assumed.

This scheme is commonly described as unconditionally stable for any time step. Nonetheless, it returns an oscillatory solution, but technically speaking, this oscillation will vanish gradually and it will return a solution that is not always realistic. That happens for the same reason as in the explicit scheme.

2.2.3. Fully Implicit scheme

This scheme is used to obtain the temperature Tp^1 in state t+Δt only with this new step, in an iterative method due to the fact that f is equal to 1. Because of this, no coefficient can be negative. In this way, the implicit scheme returns a physically satisfactory behaviour.

Therefore, this scheme shows a high stability, although it entails a high computational cost.

2.3. Numerical schemes for interpolation

The approximations of the integrals require the values of the variables at the interfaces of the computational nodes. For that reason, the values of these points need to be interpolated using control volume nodes. The interpolation for the face e is going to be shown for different interpolation schemes.

2.3.1. Central Difference Scheme (CDS)

It is a second order scheme [31], so the variable at the cell face is calculated as an arithmetic mean when it is right in the middle of the grid nodes.

$$\phi_{nf} = \frac{1}{2}\left(\phi_P + \phi_{nb}\right) \tag{2.3.1}$$

In other cases, the variable should be interpolated between grid node values.

$$\phi_{nf} = f_{nf}\phi_P + \left(1 - f_{nf}\right)\phi_{nb} \tag{2.3.2}$$

Where $\mathrm{f_{nf}}$ is the ratio defined in terms of the distances shown in Figure 2.1.2:

$$f_{nf} = \frac{(\delta x)_{nf+}}{(\delta x)_{nf}} \tag{2.3.3}$$

The assumption of linear approximation can be also applied to gradients, which yields

$$\left(\frac{\partial \phi}{\partial x}\right)_e = \frac{\phi_E - \phi_P}{x_E - x_P} \tag{2.3.4}$$

This can be also extrapolated to the other interface nodes.

2.3.2. Upwind Differencing Scheme (UDS)

It is a first order scheme, where the value of ϕ at the cell face is equal to the value of ϕ at the grid point on the upwind side of the face [13]. In other words, if υ is positive, the value of ϕ at a face cell will be the value of ϕ at the left grid point. However, if υ is negative, the value of ϕ at a face cell will be the value of ϕ at the right grid point. It will be the same for ν. An example will be:

$$\phi_e = \phi_P \quad if \quad (\overrightarrow{u} \cdot \overrightarrow{n})_e > 0 \tag{2.3.5}$$

$$\phi_e = \phi_E \quad if \quad (\overrightarrow{u} \cdot \overrightarrow{n})_e < 0 \tag{2.3.6}$$

This scheme requires fine grids in order to obtain accurate solutions because of the truncation error is a first order error.

2.3.3. Exponential Difference Scheme (EDS)

It is a second order scheme, in which the evaluation of the variables at the cell faces comes from the exact solution of the Equation 5.2.5 for steady one-dimensional problem without source term [5]. The solution for the one-dimensional problem is:

$$\frac{\phi_e - \phi_P}{\phi_E - \phi_P} = \frac{exp\,(Pe\,x_e/L) - 1}{exp\,(Pe) - 1} \tag{2.3.7}$$

Where Pe is the Peclet number; x_e is the position of the e interface node; and L is the distance of the domain.

This scheme gives an exact solution for 1D problems for any Peclet number, although it is not exact for two- or three- dimensional situations. Another disadvantage is the extra time it takes to compute the solution with exponential functions.

2.3.4. Quadratic Upwind Interpolation for Convective Kinematics (QUICK)

It is a third order scheme and the cell face value of ϕ is approximated by a parabola rather than a straight line. Therefore, as three points are needed to construct a parabola, they will be W, P and E (UU, U and D, respectively) if the flow goes in the positive direction or P, E and EE (D, U and UU, respectively) if the flow goes in the inverse direction. In other words, there is one point downstream and two points upstream. The equation to obtain the e interface value is shown below [13, 5]

$$\phi_e = \phi_U + g_1 \left(\phi_D - \phi_U\right) + g_2 \left(\phi_U - \phi_{UU}\right) \tag{2.3.8}$$

where

$$g_1 = \frac{\left(x_e - x_U\right)\left(x_e - x_{UU}\right)}{\left(x_D - x_U\right)\left(x_D - x_{UU}\right)}; \qquad g_2 = \frac{\left(x_e - x_U\right)\left(x_D - x_e\right)}{\left(x_U - x_{UU}\right)\left(x_D - x_{UU}\right)} \tag{2.3.9}$$

For uniform grids, Equation 2.3.8 can be simplified as

$$\begin{cases} \phi_e = \frac{6}{8}\phi_P + \frac{3}{8}\phi_E - \frac{1}{8}\phi_W & if \quad \left(\overrightarrow{u} \cdot \overrightarrow{n}\right)_e > 0 \\ \phi_e = \frac{6}{8}\phi_E + \frac{3}{8}\phi_P - \frac{1}{8}\phi_{EE} & if \quad \left(\overrightarrow{u} \cdot \overrightarrow{n}\right)_e < 0 \end{cases} \tag{2.3.10}$$

2.4. Boundary conditions

It is necessary to establish certain boundary conditions in order to obtain the solution of the desired case. The different boundary conditions are listed below:

- Dirichlet boundary condition: a certain variable ϕ has a prescribed value at the boundary nodes.

$$\phi = K \tag{2.4.1}$$

  Generally, it is used to establish a certain pressure, temperature or velocity at the boundary. It is also used to fix the no-slip condition in the wall by defining K equals 0 for velocity.

- Neumann boundary condition: a certain variable ϕ has a prescribed gradient normal to the boundary.

$$\frac{\partial \phi}{\partial n} = K \tag{2.4.2}$$

  Generally, it is used to establish a heat flow through the boundary walls. It can be fixed a certain value or established K to 0 for adiabatic boundary.

2.5. Solvers

It is necessary to solve a system of algebraical equations in order to obtain the field of the variable. The convergence is achieved when the values of the variable are obtained with a specified error.

$$A \cdot x = b \tag{2.5.1}$$

where A is the matrix with the dependent coefficients, x is the vector of the variable of study and b is the non-dependent coefficient.

The easiest mathematical way to solve the system shown in Equation 2.5.1 would be the inversion of the matrix A, although it is unfeasible due to the computational cost. For that reason, other ways to solve that system must be used. The velocity of convergence, the memory used, the parallelism and the simplicity of the code will depend on the chosen solver.

Most of the parameters of the matrix A are equal to 0 because each node is connected to 4 other nodes in a structured mesh. Therefore, there are some no-null diagonals depending on the scheme used to solve the system and on the numeration of the nodes. The construction of the matrix A is going to be explained with an example of a 3x3 grid (Figure 2.5.1). Each node only has five parameters: a_p, a_n, a_s, a_e and a_w.

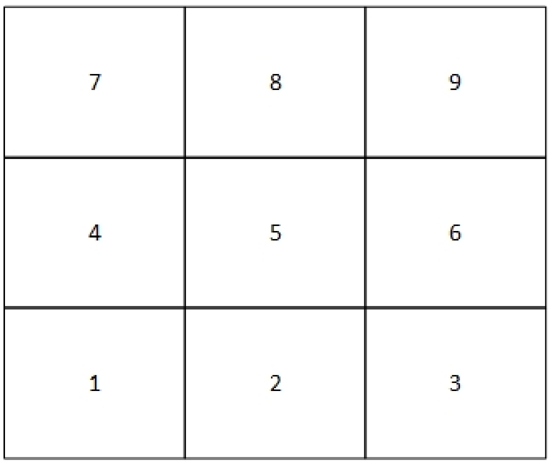


Figure 2.5.1.: 3x3 mesh to explain the construction of the matrix A

$$A=\begin{pmatrix} a_{p1} & a_{e1} & 0 & a_{n1} & 0 & 0 & 0 & 0 & 0 \\ a_{w2} & a_{p2} & a_{e2} & 0 & a_{n2} & 0 & 0 & 0 & 0 \\ 0 & a_{w3} & a_{p3} & 0 & 0 & a_{n3} & 0 & 0 & 0 \\ a_{s4} & 0 & 0 & a_{p4} & a_{e4} & 0 & a_{n4} & 0 & 0 \\ 0 & a_{s5} & 0 & a_{w5} & a_{p5} & a_{e5} & 0 & a_{n5} & 0 \\ 0 & 0 & a_{s6} & 0 & a_{w6} & a_{p6} & 0 & 0 & a_{n6} \\ 0 & 0 & 0 & a_{s7} & 0 & 0 & a_{p7} & a_{e7} & 0 \\ 0 & 0 & 0 & 0 & a_{s8} & 0 & a_{w8} & a_{p8} & a_{e8} \\ 0 & 0 & 0 & 0 & 0 & a_{s9} & 0 & a_{w9} & a_{p9} \end{pmatrix}; x=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix}; b=\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{pmatrix} \tag{2.5.2}$$

The solvers can be listed in two big groups: the direct solvers and the iterative solvers.

- Direct Solvers: this group of solvers works all the equations out directly. It will find the exact solution (there will be a rounding error in the solution) in a finite number of steps. In general, these types of solvers are complex to program and require more memory than the iterative solvers, although they are fast algorithms. Some of these direct solvers are listed below:

  – LU: this method is based on the decomposition of the matrix A in two different matrices using the multipliers from Gaussian elimination which define the L matrix (Lower triangular matrix) and the U matrix (Upper triangular matrix).

  $$LU = A \Longrightarrow LU\,x = b \tag{2.5.3}$$

  First of all, it will solve $Ly = b$ to find y, and then it will use that result to obtain the x vector according to $Ux = y$.

  – Band-LU: this method is based on LU decomposition, however it is used on banded matrices, most of the diagonal terms of which are 0. This situation fits the problem involving the current bachelor's thesis.

  – TDMA (Tri-Diagonal Matrix Algorithm): This method is a simplified form of Gaussian elimination that can be used to solve a tridiagonal system of equations:

  $$a_i x_i + a_E x_{i+1} + a_W x_{i-1} = b_i, \qquad i = 1 \div n \tag{2.5.4}$$

  It consists of two parts: the first one is a forward elimination phase, and the second one is a backward substitution phase. The algorithm consists in finding the coefficients γ_i and β_i in order to solve the next equation:

$$x_{i-1} = \gamma_i x_i + \beta_i \tag{2.5.5}$$

where

$$\gamma_{i+1} = \frac{-a_E}{a_W\gamma_i + a_i}, \qquad \beta_{i+1} = \frac{b_i - a_W\beta_i}{a_W\gamma_i + a_i} \tag{2.5.6}$$

Algorithm:

1. Set $\gamma_1 = \beta_1 = 0$.
2. Evaluate Equation 2.5.6 for i=1 to n-1.
3. Set $x_{n+1} = 0$.
4. Find Equation 2.5.5 for i=n+1 to 2.

The TDMA is only applicable to matrices that are diagonally dominant, i.e., $|a_E| > |a_i| + |a_W|$ for i equals to 1 to n.

- Iterative Solvers: this group of solvers works the equations out iteratively, so it attempts to solve the system of equations by successive approximations to the solution starting from an initial estimated value. It will find an approximate solution (there will be a convergence error in the solution) in a huge number of iterations. In general, these methods are useful for solving large matrix, where direct methods would be prohibitively expensive. Some of these iterative solvers are listed below:

  – Jacobi: this method is the easiest iterative method for solving a matrix equation on a matrix that has no zeros along its main diagonal. The Jacobi method is easily derived by examining each of the n equation in the linear system of Equation 2.5.1 in isolation. Thus, for a linear system $a_{i,j}x_i + \sum_{nb} a_{i,j}x_i^* = b_i$, the solution will be

$$x_i = \frac{b_i - \sum_{nb} a_{i,j}x_i^*}{a_{i,j}} \tag{2.5.7}$$

  It can be extrapolated to matrices where the matrix A can be split in three matrices.

$$A = D + U + L \Longrightarrow Dx = b - (U + L)\,x^* \tag{2.5.8}$$

  where D is the diagonal matrix, L is the lower matrix and U is the upper matrix. The algorithm is:

1. Solve Equation 2.5.7 with the values of the previous time step.
2. Iterate Equation 2.5.7 with the values of the previous iteration until the convergence is achieved.

– Gauss-Seidel: this method is based on the Jacobi algorithm but it uses the new values of each x_i as soon as they are known. That is, once x_1 has been determined from the first equation, its value is then used in the second equation to obtain the new x_2. Similarly, the new x_1 and x_2 are used in the third equation to obtain the new x_3, and so on. Thus, for a linear system $a_{i,j}x_i + \sum_{nb}^{known} a_{i,j}x_i + \sum_{nb}^{unknown} a_{i,j}x_i^* = b_i$, the solution will be

$$x_i = \frac{b_i - \sum_{nb}^{known} a_{i,j}x_i - \sum_{nb}^{unknown} a_{i,j}x_i^*}{a_{i,j}} \tag{2.5.9}$$

It can be extrapolated to matrices where the matrix A can be split in three matrices.

$$A = D + U + L \Longrightarrow (D + L)\,x = b - Ux^* \tag{2.5.10}$$

The algorithm is:

1. Solve Equation 2.5.9 with the values of the previous time step.
2. Iterate Equation 2.5.9 with the values of the previous iteration until the convergence is achieved.

It is also true that the Gauss-Seidel method does not always converge. However, the Scarborough criterion guarantees the convergence of a system of equations when it is satisfied.

$$\frac{\sum |a_{nb}|}{|a_p|} \begin{cases} \leq & for\ all\ equations \\ < & for\ at\ least\ one\ equation \end{cases} \tag{2.5.11}$$

The Gauss-Seidel method has an enormous disadvantage due to the fact that the boundary-condition information is transmitted at a rate of one grid each iteration. Therefore, the convergence is slow.

– TDMA-GS: this method is used when the A matrix has more than 3 no-null diagonals due to the fact that a TDMA cannot be used. For a 2D case, there are 5 or more no-null diagonals depending on the numerical scheme used. Nevertheless, for a 3D case, there are 7 or more no-null diagonals depending on the numerical scheme used. A 2D or 3D case can be converted to a 1D case solved by a TDMA algorithm, assuming the neighbouring temperatures at the neighbouring lines from the previous step or iteration in each one-dimensional problem. The line-by-line method has a faster convergence because the boundary condition information is transmitted at once to the interior of the domain in each line solve with the direct method.

$$a_{i,j}x_{i,j} + a_W x_{i,j-1} + a_E x_{i,j+1} + a_N x_{i+1,j} + a_S x_{i-1,j} = \bar{b}_{i,j};\ i,j = 1\ to\ n \tag{2.5.12}$$

This previous equation can be rewritten as

$$a_{i,j}x_{i,j} + a_W x_{i,j-1} + a_E x_{i,j+1} = b_{ij} \tag{2.5.13}$$

where

$$b_{ij} = \bar{b}_{i,j} - a_N x^*_{i+1,j} - a_S x_{i-1,j} \tag{2.5.14}$$

or

$$a_{i,j}x_{i,j} + a_N x_{i+1,j} + a_S x_{i-1,j} = b_{ij} \tag{2.5.15}$$

where

$$b_{i,j} = \bar{b}_{i,j} - a_W x_{i,j-1} - a_E x^*_{i,j+1} \tag{2.5.16}$$

Algorithm solving line-by-line the mesh:

1. Solve horizontally line-by-line the TDMA algorithm for i=1 to n.
2. Set $x^{1*} = x^1$.
3. Solve vertically line-by-line the TDMA algorithm for j=1 to n.
4. Check the convergence, if it is not achieved set $x^{1*} = x^1$ and go to 1. If it is achieved continue the code.

– Conjugate gradient: this method is used to find the nearest local minimum of a function of n variables which presupposes that the gradient of the function can be computed [16]. The matrix A must be symmetric and positive-definite[1]. It uses conjugate directions instead of the local gradient for going downhill due to the fact that the minimum is reached in far fewer steps. The equation to optimize is

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b, \quad where\ b, x \in \mathbb{R}^n\ and\ A \in \mathbb{R}^{nxn} \tag{2.5.17}$$

The minimizer x^* of the function ϕ is given as the point where the gradient

[1]A symmetric matrix A is SPD (symmetric positive definite) if $x^T Ax > 0\ \forall x \in \Omega$, or equivalently if all the eigenvalues of A are positive.

of the function is equal to zero. Algebraic calculation leads to

$$\nabla\phi\left(x^*\right) = Ax^* - b = 0 \quad or \quad Ax^* = b \tag{2.5.18}$$

Consequently, the x^* variable minimizes the $\phi\left(x\right)$ function. For that reason, a solver is tried to be found that minimizes the $\phi\left(x\right)$ function instead of looking for a direct solver to solve the algebraic system. This method will find unique and global minimum in n steps by successively minimizing it along each of the linearly independent conjugate directions $p_i \, and \, p_j \; for \, \forall i \neq j$[2]. The difference between the first guess x_0 and the exact solution x^*can be computed as

$$x^* = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + ... + \alpha_{n-1} p_{n-1} \tag{2.5.19}$$

where α_k is the step length for each search direction using the residual value r_k.

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k} \tag{2.5.20}$$

The A-conjugate search directions may be the eigenvector of A, although finding the eigenvalues requires high computational power. For that reason, a conjugate direction method is used in order to find those search directions such that each new p_k can be found only by using the previous direction. That happens due to the fact that each new direction vector is a linear combination of the negative residual $-r_k = Ax_0 - b$ and the previous search vector p_{k-1}.

$$p_k = -r_k + \beta_k p_{k-1} \tag{2.5.21}$$

where β_k is found by imposing the A-conjugancy condition

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}} \tag{2.5.22}$$

The algorithm is

1. Compute $r_0 = Ax_0 - b$ and $p_0 = -r_0$.
2. Iterate for k=1,2... until convergence:
   a) Evaluate Equation 2.5.20 for α_{k-1}.
   b) Evaluate Equation 2.5.19 until the k component: $x_k = x_{k-1} +$

[2]A-conjugancy is defined as a conjugancy of a set of non-zeros vectors $\{p_0, p_1, ..., p_{n-1}\}$ with the A matrix where the p_K is the search direction. That is $p_i^T A p_j \; \forall i \neq j$.

$\alpha_{k-1} p_{k-1}$.

c) Evaluate the new residual by $r_k = r_{k-1} + \alpha_{k-1} A p_{k-1}$.

d) Evaluate Equation 2.5.22 for β_k.

e) Evaluate Equation 2.5.21 for p_k.

f) Evaluate the convergence with the residual value.

It is true that the direct solvers are more accurate and faster than iterative solver[3]. However, iterative solvers are easier to program than direct solvers. This is the main reason this type of solvers has been used. Moreover, a 10^{-6} precision is enough to obtain a good macroscopic result.

In this study, a Jacobi Solver, GS solver, TDMA-GS solver, Conjugate gradient solver and a biconjugate gradient stabilized solver have been programmed. The time required for the first two solvers is almost the same, but the TDMA-GS or the Conjugate gradient solver are significantly faster for fine meshes. For that reason, a TDMA-GS or a Conjugate gradient solver has been used.

2.6. Resolution algorithm

The global resolution algorithm is shown in Figure 2.6.1. This algorithm consists of four parts rather differentiable:

- Problem definition: it consists of the definition of the physical problem with mathematical equations and of the boundary conditions. The type of mesh is selected as well as the numerical time scheme and the numerical spatial discretization.
- Data input and initial calculations: all the required parameters by the program are declared. Apart from the required parameters, the physical properties of the fluids and materials, the physical dimensions of the domain and convergence parameters are defined. The mesh used by the program and some other calculation such as constant coefficients are also defined .
- Iterative block: it solves the system of equations until the convergence is achieved defined by the convergence criterion.
- Final calculations and result visualization: once the convergence is achieved, the final calculations are done in order to extract the values and save them in a file. Finally, the results are plotted using a commercial software like Matlab.

[3]Note that it is only true when a high precision is required.

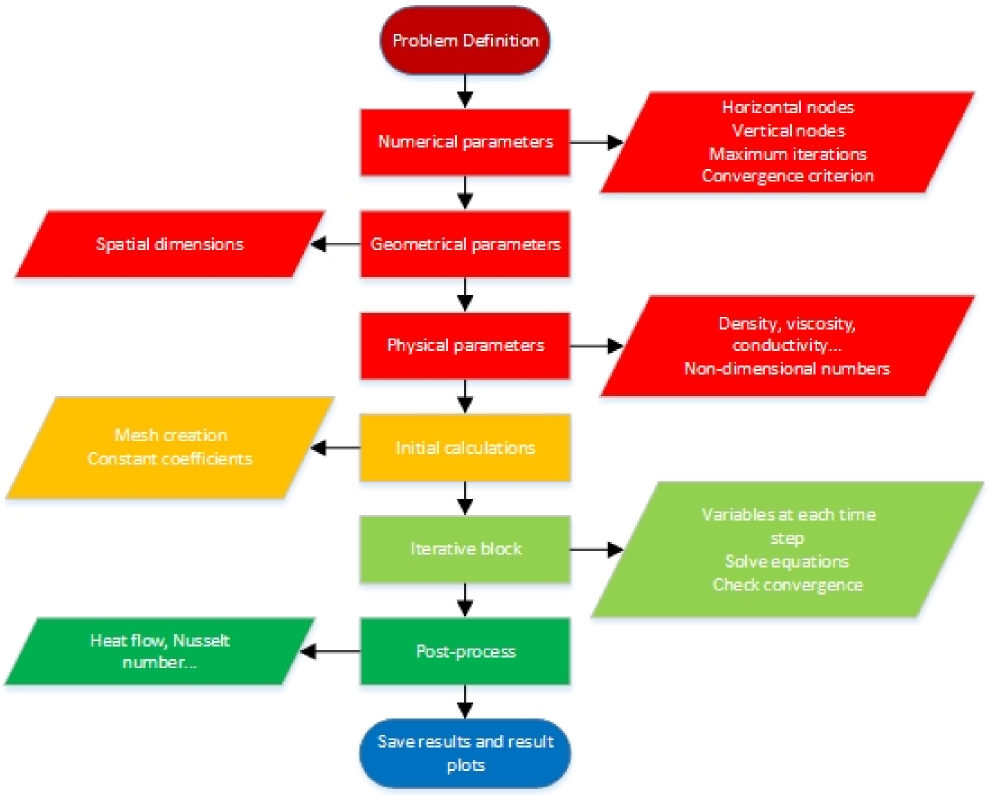


Figure 2.6.1.: Global resolution algorithm

Part III.

Heat conduction transfer

3. Heat Conduction

3.1. Introduction

Heat transfer conduction is the transfer of heat from one solid to another in contact that is at a different temperature, or just from one part of the body (a control volume) to another part in contact of the same body. The transferred heat will depend on the difference between these control volumes and the physical properties of both of them.

3.2. The governing equations

The equation for steady conduction in two dimensions is shown in Equation 3.2.1. For more information, see [31].

$$\frac{\partial}{\partial x}\left(\kappa\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\kappa\frac{\partial T}{\partial y}\right)+S=0 \tag{3.2.1}$$

Where $\varkappa$ is the thermal conductivity and S is the rate of heat generation per unit volume, which is defined in Equation 3.2.2:

$$S=S_c+S_pT_p \tag{3.2.2}$$

The equation for unsteady conduction in two dimensions is shown in Equation 3.2.3:

$$\rho C_p\frac{\partial T}{\partial t}=\frac{\partial}{\partial x}\left(\kappa\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\kappa\frac{\partial T}{\partial y}\right)+S \tag{3.2.3}$$

Where $\varkappa$ is the thermal conductivity, ϱ is the material density, $\mathrm{C_p}$ is the specific thermal capacity and S is the rate of heat generation per unit volume, which is defined in Equation 3.2.4:

$$S=S_c+S_pT_p \tag{3.2.4}$$

3.3. Spatial discretization

Equation 3.2.3 is integrated to obtain a discretization equation over the control volume shown in Figure 2.1.1 and over time from t to t+Δt.

$$\int_V \int_t^{t+\Delta t} \rho C_p \frac{\partial T}{\partial t} dt dV = \int_t^{t+\Delta t} \int_w^e \frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right) e\, dx dy dt + \int_t^{t+\Delta t} \int_s^n \frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{\partial y}\right) e\, dy dx dt \tag{3.3.1}$$

Where

$$\int_V \int_t^{t+\Delta t} \rho C_p \frac{\partial T}{\partial t} dt dV = \bar{\rho}\bar{C}_p \left(T_p^1 - T_p^0\right) V_p \tag{3.3.2}$$

$$\int_t^{t+\Delta t} \int_w^e \frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right) e\, dx dy dt = \int_t^{t+\Delta t} \left[\frac{\kappa_e \left(T_E - T_P\right)}{\left(\delta x\right)_e} S_e - \frac{\kappa_w \left(T_P - T_W\right)}{\left(\delta x\right)_s} S_s\right] dt \tag{3.3.3}$$

$$\int_t^{t+\Delta t} \int_s^n \frac{\partial}{\partial y}\left(\kappa \frac{\partial T}{\partial y}\right) e\, dy dx dt = \int_t^{t+\Delta t} \left[\frac{\kappa_n \left(T_N - T_P\right)}{\left(\delta y\right)_n} S_n - \frac{\kappa_s \left(T_P - T_S\right)}{\left(\delta y\right)_s} S_s\right] dt \tag{3.3.4}$$

3.4. Time discretization

The time integration is shown below:

$$\int_t^{t+\Delta t} \left[\frac{\kappa_e \left(T_E - T_P\right)}{\left(\delta x\right)_e} S_e - \frac{\kappa_w \left(T_P - T_W\right)}{\left(\delta x\right)_s} S_s\right] dt = \Delta t \left(f \cdot \left[\frac{\kappa_e \left(T_E^1 - T_P^1\right)}{\left(\delta x\right)_e} S_e - \right.\right.$$
$$\left.\left. - \frac{\kappa_w \left(T_P^1 - T_W^1\right)}{\left(\delta x\right)_s} S_s\right] + (1-f) \cdot \left[\frac{\kappa_e \left(T_E^0 - T_P^0\right)}{\left(\delta x\right)_e} S_e - \frac{\kappa_w \left(T_P^0 - T_W^0\right)}{\left(\delta x\right)_s} S_s\right]\right) \tag{3.4.1}$$

$$\int_t^{t+\Delta t} \left[\frac{\kappa_n \left(T_N - T_P\right)}{\left(\delta y\right)_n} S_n - \frac{\kappa_s \left(T_P - T_S\right)}{\left(\delta y\right)_s} S_s\right] dt = \Delta t \left(f \cdot \left[\frac{\kappa_n \left(T_N^1 - T_P^1\right)}{\left(\delta y\right)_n} S_n - \right.\right.$$
$$\left.\left. = \frac{\kappa_s \left(T_P^1 - T_S^1\right)}{\left(\delta y\right)_s} S_s\right] + (1-f) \cdot \left[\frac{\kappa_n \left(T_N^0 - T_P^0\right)}{\left(\delta y\right)_n} S_n - \frac{\kappa_s \left(T_P^0 - T_S^0\right)}{\left(\delta y\right)_s} S_s\right]\right) \tag{3.4.2}$$

Where f is a weighting factor between 0 and 1, explicit and implicit schemes, respectively.

Combination of Equation 3.3.2, Equation 3.4.1 and Equation 3.4.2 yields

$$\frac{\bar{\rho}\bar{C}_p V_p}{\Delta t}\left(T_p^1 - T_p^0\right) = f \cdot \left[\frac{\kappa_e \left(T_E^1 - T_P^1\right)}{(\delta x)_e} S_e - \frac{\kappa_w \left(T_P^1 - T_W^1\right)}{(\delta x)_s} S_s\right] + (1-f) \cdot \left[\frac{\kappa_e \left(T_E^0 - T_P^0\right)}{(\delta x)_e} S_e - \frac{\kappa_w \left(T_P^0 - T_W^0\right)}{(\delta x)_s} S_s\right] + f \cdot \left[\frac{\kappa_n \left(T_N^1 - T_P^1\right)}{(\delta y)_n} S_n - \frac{\kappa_s \left(T_P^1 - T_S^1\right)}{(\delta y)_s} S_s\right] + (1-f) \cdot \left[\frac{\kappa_n \left(T_N^0 - T_P^0\right)}{(\delta y)_n} S_n - \frac{\kappa_s \left(T_P^0 - T_S^0\right)}{(\delta y)_s} S_s\right] \tag{3.4.3}$$

3.5. Discretization of the domain

An FVM orthogonal, structural discretization is used, see subsection 2.1.1. For that reason, uniform properties in control volume faces have to be determined considering a linear variation of magnitude between P and one of the neighbouring points by using a CDS scheme. Then,

$$\phi_{nf} = f_{nf}\phi_P + (1 - f_{nf})\,\phi_{nb} \tag{3.5.1}$$

Where $\mathrm{f_{nf}}$ is equal to $f_{nf} = \frac{(\delta x)_{nf+}}{(\delta x)_{nf}}$.

The previous method is useful although there is an alternative to achieve a good representation for the flux $\mathrm{q_{nf}}$ at the interface of a neighbour from the discretization of Equation 3.2.1.

$$q_{nf} = \frac{\kappa_{nf}(T_P - T_{nb})}{(\delta x)_{nf}} \tag{3.5.2}$$

The previous equation can also be expressed with nodal properties instead of interface properties.

$$q_{nf} = \frac{T_P - T_{nb}}{{(\delta x)_{nf-}}/{\kappa_P} + {(\delta x)_{nf+}}/{\kappa_{nb}}} \tag{3.5.3}$$

Combination of Equation 3.5.2 and Equation 3.5.3 yields

$$\kappa_{nf} = \left(\frac{1 - f_{nf}}{\kappa_P} + \frac{f_{nf}}{\kappa_{nb}}\right)^{-1} \tag{3.5.4}$$

This methodology can be easily extrapolated to two or three dimensions, and other properties.

3.6. Boundary conditions

Typically, three different boundary conditions are defined in heat conduction. They are listed below:

- Boundary temperature: temperature is established as a neighbouring temperature with a $(\delta x)_{nb} = (\delta x)_{nf}$ in nodal centred method.
- Boundary flux heat: a neighbouring coefficient is established by the flux heat value given.
- Boundary heat flux specified via a heat transfer coefficient and the temperature of the surrounding fluid.

3.7. Implicit discretization equation for two dimensions

Equation 3.2.3 can be easily turned into the discretization equation

$$a_p T_p = a_E T_E + a_W T_W + a_N T_N + a_S T_S + b \tag{3.7.1}$$

where

$$a_E = \frac{\kappa_e S_e}{(\delta x)_e} \tag{3.7.2}$$

$$a_W = \frac{\kappa_w S_w}{(\delta x)_w} \tag{3.7.3}$$

$$a_N = \frac{\kappa_n S_n}{(\delta y)_n} \tag{3.7.4}$$

$$a_S = \frac{\kappa_s S_s}{(\delta y)_s} \tag{3.7.5}$$

$$a_p^0 = \frac{\rho C_p V_p}{\Delta t} \tag{3.7.6}$$

$$b = S_c V_p + a_p^0 T_p^0 \tag{3.7.7}$$

$$a_p = a_E + a_W + a_N + a_S + a_p^0 - S_p V_p \tag{3.7.8}$$

Equation 3.2.1 can be easily turned into the previous system with some changes

$$a_p T_p = a_E T_E + a_W T_W + a_N T_N + a_S T_S + b \tag{3.7.9}$$

where

$$a_E = \frac{\kappa_e S_e}{(\delta x)_e} \tag{3.7.10}$$

$$a_W = \frac{\kappa_w S_w}{(\delta x)_w} \tag{3.7.11}$$

$$a_N = \frac{\kappa_n S_n}{(\delta y)_n} \tag{3.7.12}$$

$$a_S = \frac{\kappa_s S_s}{(\delta y)_s} \tag{3.7.13}$$

$$b = S_c V_p \tag{3.7.14}$$

$$a_p = a_E + a_W + a_N + a_S - S_p V_p \tag{3.7.15}$$

3.8. Algorithm

Equation 3.7.1 and Equation 3.7.9 are linear algebraic equations. However, some parameters or properties of the coefficients $\mathrm{a_{nb}}$ may depend on temperature, for example density, conductivity and so on. For that reason, these equations have to be calculated through iteration. The general algorithm is shown in Figure 3.8.1.

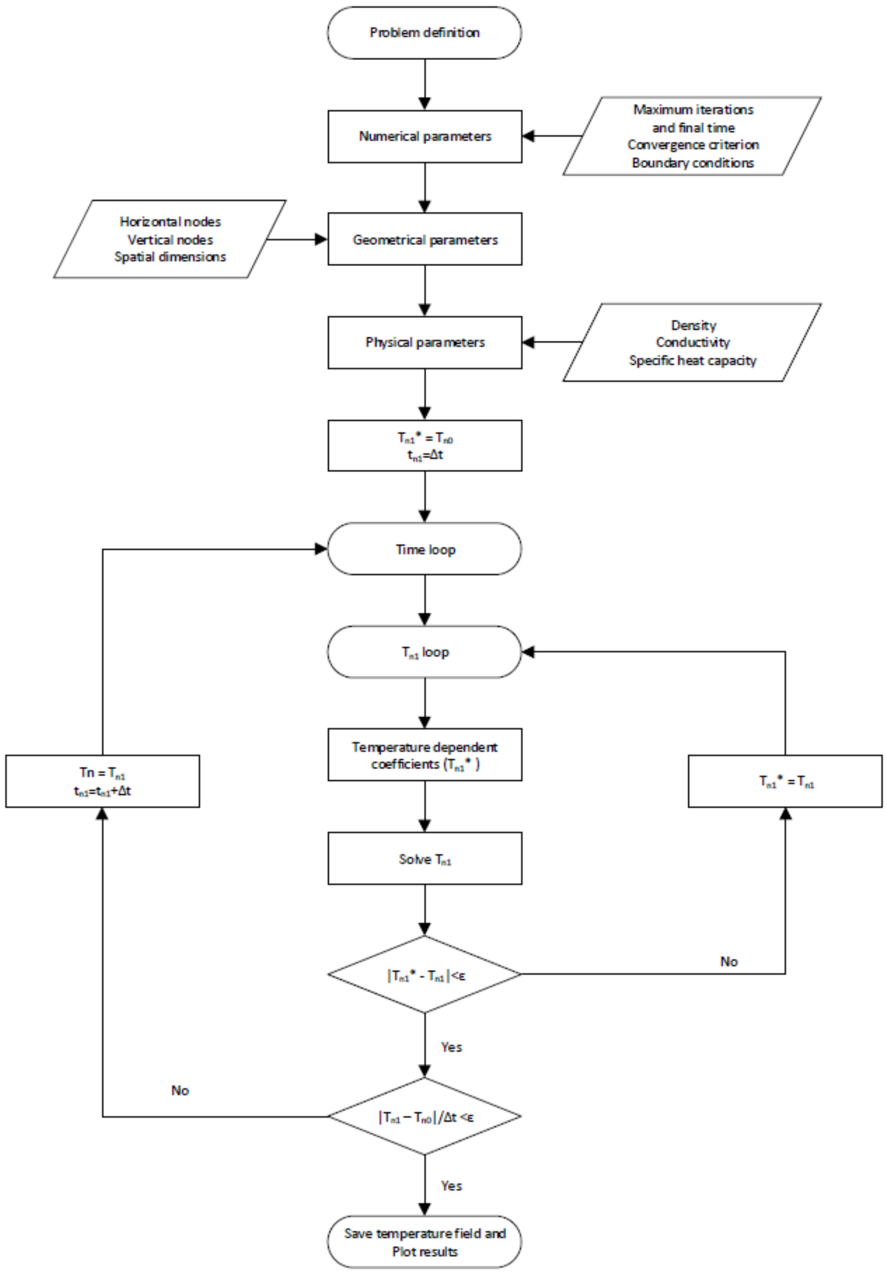

Problem definition
Numerical parameters
Maximum iterations and final time Convergence criterion Boundary conditions
Horizontal nodes Vertical nodes Spatial dimensions
Geometrical parameters
Physical parameters
Density Conductivity Specific heat capacity
$T_{n1}^* = T_{n0}$ $t_{n1}=\Delta t$
Time loop
T_{n1} loop
Temperature dependent coefficients (T_{n1}^*)
Tn = T_{n1} $t_{n1}=t_{n1}+\Delta t$
$T_{n1}^* = T_{n1}$
Solve T_{n1}
$|T_{n1}^* - T_{n1}|<\varepsilon$
No
Yes
No
$|T_{n1} - T_{n0}|/\Delta t < \varepsilon$
Yes
Save temperature field and Plot results

Figure 3.8.1.: Conduction algorithm

4. A Two-dimensional Transient Conduction Problem

4.1. Introduction

The case chosen to be solved and to prove the previous equations is an unsteady case with constant properties that consists of four different materials. See Figure 4.1.1.

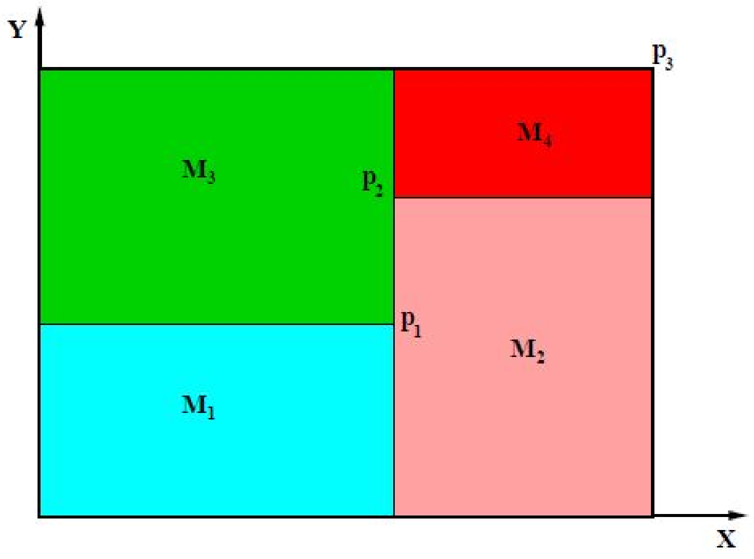


Figure 4.1.1.: Heat conduction problem

4.2. Spatial discretization

There are four different materials defined by the points in Table 4.2.1. The material's values, otherwise, are shown in Table 4.2.2.

The spatial discretization is shown in Table 4.2.3. It has been used a uniform, structured, orthogonal mesh.

	$x\,[m]$	$y\,[m]$
p_1	0.5	0.4
p_2	0.5	0.7
p_3	1.1	0.8

Table 4.2.1.: Problem coordinates for heat conduction problem

	$\rho\,[kg/m^3]$	$C_p\,[J/kgK]$	$\kappa\,[W/mK]$
M_1	1500.0	750.0	170.0
M_2	1600.0	770.0	140.0
M_3	1900.0	810.0	200.0
M_4	2500.0	930.0	140.0

Table 4.2.2.: Physical properties for heat conduction problem

Spatial property	L	H	$\mathrm{N_x}$	$\mathrm{N_y}$	Δx	Δy
Value	1.1	0.8	55	40	0.02	0.02

Table 4.2.3.: Spatial discretization for heat conduction problem

4.3. Boundary conditions

The boundary conditions are defined below

- Initial temperature field: $T = 8.00^{\circ}C$.
- Left boundary condition: in contact with a fluid at $T_g = 33.00^{\circ}C$ and a heat transfer coefficient $h = 9.00W/m^2K$.
- Right boundary condition: uniform temperature at $T = 8.00 + 0.005t\ ^{\circ}C$.
- Bottom boundary condition: isotherm boundary at $T = 23.00^{\circ}C$.
- Top boundary condition: uniform heat flow $Q = 60.00W/m$

4.4. The governing equations

The equations used to solve the problem are shown in Equation 3.7.1. The scheme chosen to solve the conductive heat problem is an explicit scheme, because it does not iterate to obtain the next step value. However, this scheme needs a minor step settled in 0.01 seconds, which will be small enough to get the convergence of the equations.

An iterative solver has been used to obtain the temperature field, particularly a Gauss-Seidel method due to its simplicity.

4.5. The program

The program creates a structure with all the information related to each grid node. Therefore, it contains the temperature values, the coordinate values of the grid nodes and interface nodes, the type of material, the physical properties and the coefficients of Equation 3.7.1.

The inputs required by the program are the point coordinates, the spatial step and the number of nodes in each direction, the convergence criterion, the relaxation factor, the coefficient to select the scheme, the width of the volume, the heat transfer coefficient for the left boundary condition, the left and right boundary temperatures and finally the initial temperature field. Another requirement needed by the program is the last time to solve the time loop. Moreover, another structure with the variables of density, the specific thermal capacity and heat thermal conductivity is programmed.

The program consists in a variety of files which calculate different parts of the equations. The functions are explained below

- Main file: it contains the general algorithm that calls the different subfiles to perform its function inside the algorithm. See Figure 4.5.1.
- Properties file: it calculates the geometrical properties of each grid node and its interfaces using a uniform orthogonal mesh, and it determines the physical properties for each node. Therefore, it determines the conductivity at the interfaces using the harmonic mean.
- Constant coefficients file: it calculates the constant coefficients taking into account the different conditions of each node. Note that there are 9 different situations.
- Coefficients file: it calculates the temperature dependent coefficient taking into account the different conditions of each node. In this case there are also 9 different situations. This code also solves the equation Equation 3.7.1.
- Temperature field at 5000s file: it stores the entire temperature field in a txt file. These values will be useful to validate the code.
- Index file: it determines the index value for two coordinates points chosen to validate the code.

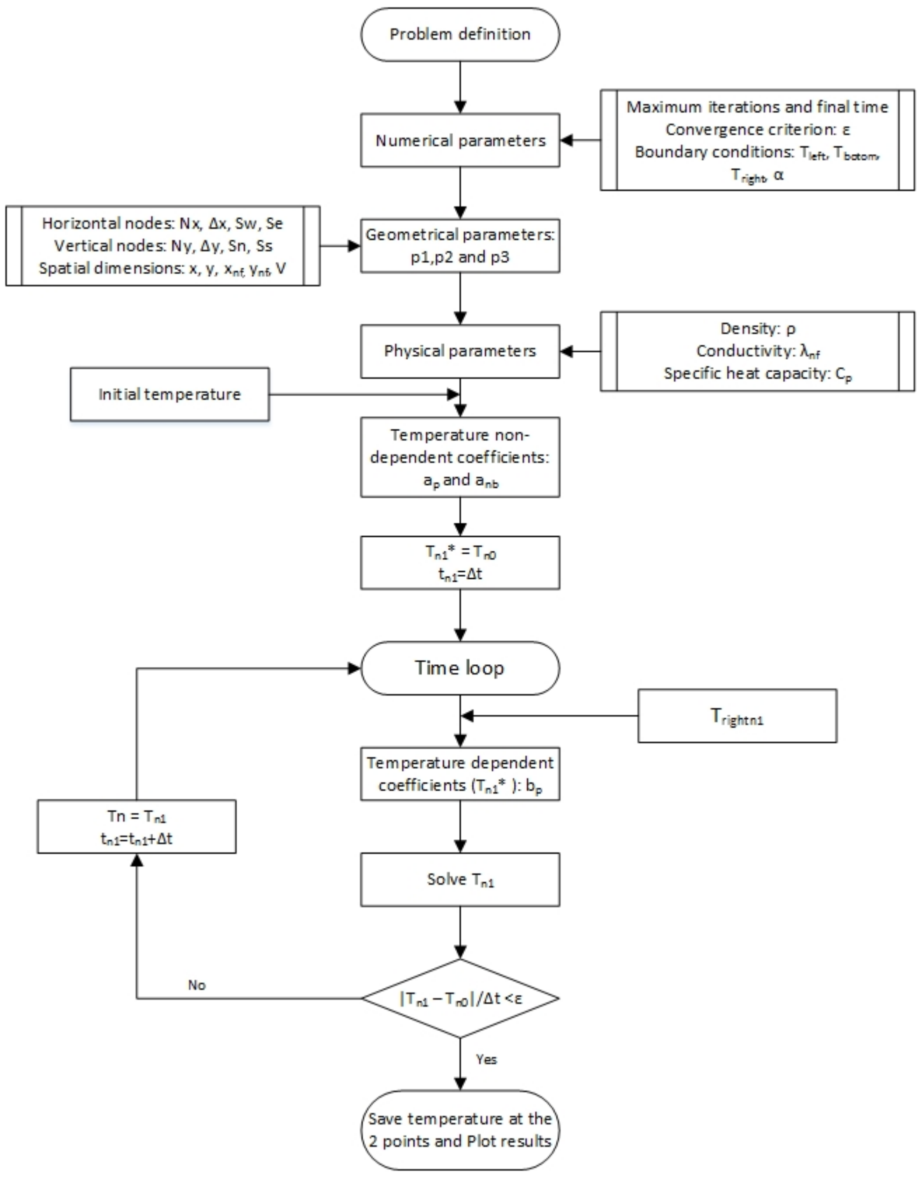

Problem definition
Numerical parameters
Maximum iterations and final time
Convergence criterion: ε
Boundary conditions: T_{left}, T_{botom}, T_{right}, α
Horizontal nodes: Nx, Δx, Sw, Se
Vertical nodes: Ny, Δy, Sn, Ss
Spatial dimensions: x, y, x_{nf}, y_{nf}, V
Geometrical parameters: p1,p2 and p3
Physical parameters
Density: ρ
Conductivity: λ_{nf}
Specific heat capacity: C_p
Initial temperature
Temperature non-dependent coefficients: a_p and a_{nb}
$T_{n1}^* = T_{n0}$
$t_{n1} = \Delta t$
Time loop
$T_{rightn1}$
Temperature dependent coefficients (T_{n1}^*): b_p
Tn = T_{n1}
$t_{n1} = t_{n1} + \Delta t$
Solve T_{n1}
No
$|T_{n1} - T_{n0}|/\Delta t < \varepsilon$
Yes
Save temperature at the 2 points and Plot results

Figure 4.5.1.: Algorithm used in the specific case

4.6. The results

After the code was created, it was validated with the temperature of the two coordinate points to make sure that everything was correct. The results for the temperature field at 5000 seconds are shown below (Figure 4.6.1), and it can be compared with the expected field (Figure 4.6.2).

4.7. Conclusions

It can be concluded that the number of chosen nodes is enough to obtain an accurate solution. This solution has a physical meaning due to the fact that the right side is

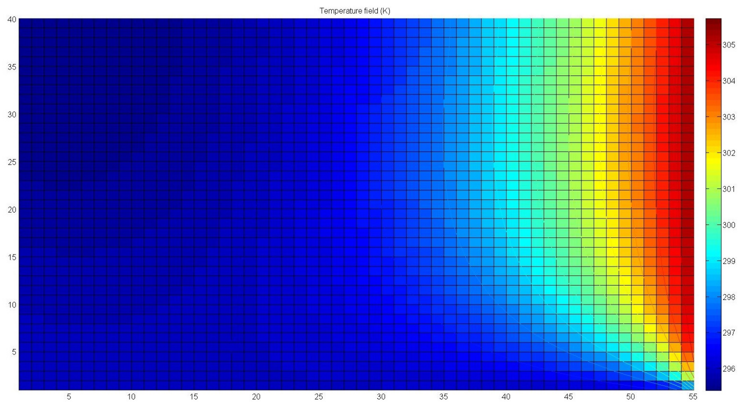


Figure 4.6.1.: Heat conduction solution at 5000s

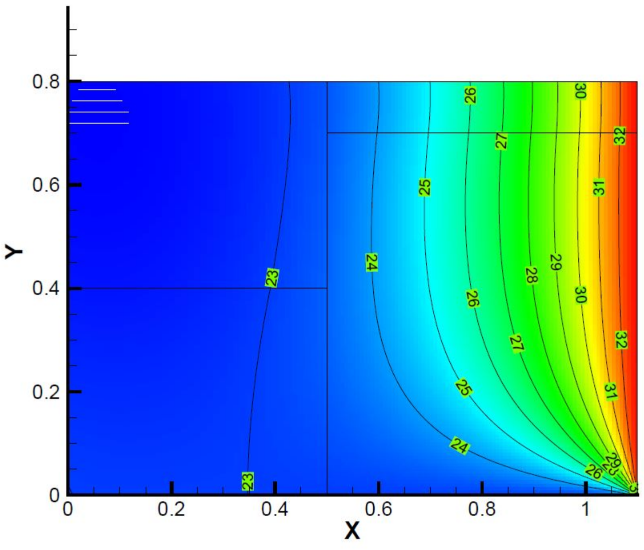


Figure 4.6.2.: Expected heat conduction solution at 5000s

at approximately 33ºC and the bottom boundary at 23ºC.

Inside the domain, a temperature degradation can be observed from a temperature of about 33ºC on the right boundary to a temperature of 23ºC on the left and bottom boundaries.

Part IV.

Laminar heat convection transfer

5. Laminar heat convection: the Navier-Stokes Equations

5.1. The Navier-Stokes Equations

The N-S equations consist of the continuity equation, which represents the mass conservation principle (Equation 5.1.1); the momentum conservation equations, one for each of the problem's dimension (Equation 5.1.2); and the energy conservation equation (Equation 5.1.3). For more information, see [31, 36].

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{5.1.1}$$

$$\frac{\partial}{\partial t}(\rho \vec{v}) + \nabla \cdot (\rho \vec{v}\vec{v}) = -\nabla p + \nabla \cdot (\vec{\tau}) + \rho \vec{g} \tag{5.1.2}$$

$$\frac{\partial}{\partial t}(\rho(u+e_c)) + \nabla \cdot ((u+e_c)\rho\vec{v}) = -\nabla \cdot (p\vec{v}) + \nabla \cdot (\vec{v} \cdot \vec{\tau}) - \nabla \cdot \vec{q} + \rho\vec{g} \cdot \vec{v} + \Phi \tag{5.1.3}$$

Where ρ is the density, t is the time, $\vec{v}$ is the velocity vector, p is the pressure, $\vec{\tau}$ is the total stress tensor and equal to $\vec{\tau} = \mu\left[\left(\nabla\vec{v} + \nabla\vec{v}^{\,T}\right) - \frac{2}{3}\nabla \cdot \vec{v} I\right]$, $\vec{g}$ is the gravitational acceleration vector, u is the internal energy, e_c is the kinetic energy, $\vec{q}$ is the heat flow and Φ is the internal source.

Note that some assumptions have been done in the previous equations. The hypothesis done are: continuity of mater, continuum assumption, relativity effects negligible, inertial reference system, magnetic and electromagnetic body forces negligible.

5.1.1. The continuity conservation equation

This equation defines that:

"The variety of the mass in the control volume has to be equal to the mass flow through its faces".

Therefore,

- The first term of Equation 5.1.1 represents the variation of mass in the control volume in a differential of time.
- The second term of Equation 5.1.1 represents the mass flow through the faces of the control volume.

5.1.2. The momentum conservation equation

This equation defines that:

"The variety of the linear momentum in the control volume plus momentum flux through the faces of its volume has to be equal to the sum of the forces acting on the volume".

Therefore,

- The first term of Equation 5.1.2 represents the variation of the linear momentum in the control volume.
- The second term of Equation 5.1.2 represents the momentum flux through the faces of its control volume.
- The third term of Equation 5.1.2 is the pressure gradient, an axial force that acts on the faces of the control volume.
- The fourth term of Equation 5.1.2 is the total stress tensor. This force acts axially and tangentially on the faces of the control volume. Its value depends on the type of fluid, p.e. a Newtonian fluid, a non-Newtonian fluid ...
- The fifth term of Equation 5.1.2 is the volumetric force. This force may be a gravitational force, electrical force, magnetic force or electromagnetic force.

5.1.3. The energy conservation equation

This equation defines that:

"The variety of internal energy and kinetic energy in a control volume plus the flow of these variables must be equal to the work done on the control volume plus incoming heat flow through the faces of the control volume plus the energy of the sources in the control volume".

Therefore,

- The first term of Equation 5.1.3 represents the variation of the internal and kinetic energy in the control volume.

- The second term of Equation 5.1.3 represents the energy flow of these variables through the faces of its volume.
- The third and fourth term of Equation 5.1.3 is the work done by the superficial forces like $-\nabla \cdot\ (p\overrightarrow{v}) + \nabla \cdot\ (\overrightarrow{v} \cdot \overrightarrow{\tau})$.
- The fifth term of Equation 5.1.3 is the incoming heat flow through the faces of the control volume.
- The sixth term of Equation 5.1.3 is the work done by the volumetric forces. In the case shown, there is only the gravitational work.
- The sixth term of Equation 5.1.3 is the work done by the internal sources.

5.2. The simplified N-S Equations

The N-S Equations can be simplified for specific conditions if some assumptions are done [5]. This hypothesis are:

- Bidimensional model
- Laminar flow
- Incompressible flow
- Newtonian fluid
- Boussinesq hypothesis (it assumes constant physical properties except in the body force terms)
- Negligible viscous dissipation
- Non-participating medium in radiation
- Mono-component and mono-phase fluid

Note that this simplified system of equations uses constant properties of density, thermal conductivity, specific thermal conductivity ... Therefore, it will not be able to solve problems with a huge range in temperatures because all these properties depend on temperature.

The governing equations obtained from simplifying Equation 5.1.1, Equation 5.1.2 and Equation 5.1.3, in Cartesian coordinates, are:

$$\frac{\partial u}{\partial x} + \frac{\partial \nu}{\partial y} = 0 \tag{5.2.1}$$

$$\rho\frac{\partial u}{\partial x} + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} = -\frac{\partial p_d}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{5.2.2}$$

$$\rho\frac{\partial v}{\partial x}+\rho u\frac{\partial v}{\partial x}+\rho v\frac{\partial v}{\partial y}=-\frac{\partial p_d}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+\rho g\beta\left(T-T_\infty\right) \tag{5.2.3}$$

$$\rho\frac{\partial T}{\partial t}+\rho u\frac{\partial T}{\partial x}+\rho v\frac{\partial T}{\partial y}=\frac{\lambda}{C_p}\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)+\frac{\Phi}{C_p} \tag{5.2.4}$$

Where $u\,and\,v$ are the velocity components; $x\,and\,y$ are the spatial coordinates; ρ is the density; p is the dynamic pressure; μ is the dynamic viscosity; g is the gravitational acceleration; β is the volumetric thermal expansion coefficient; T is the temperature; t is the time; $\lambda\ or\ \kappa$ is the thermal conductivity; C_p is the specific thermal heat; and Φ is the heat source.

The 4 unknowns are: pressure, temperature, and the two components of velocity $u\ and\ v$. Furthermore, a boundary condition and an initial condition are required to solve the problem.

If the equation system is closely analysed, it is easy to characterize a strong coupling between them.

- Pressure-velocity: for the previously established conditions, there is no specific pressure equation, but the pressure distribution allows the velocity field to satisfy the mass conservation equation.

- Temperature-velocity: there is only a coupling characterization for natural convection, mixed convection or when the physical properties depend on the temperature. In forced convection and constant physical properties, the velocity field does not depend on temperature field.

All the equations written above (Equation 5.2.1, Equation 5.2.2, Equation 5.2.3 and Equation 5.2.4) can be summarized in the general convection-diffusion equation:

$$\frac{\partial\rho\phi}{\partial t}+\nabla\left(\rho\vec{v}\phi\right)=\nabla\left(\mathit{\Gamma}\nabla\phi\right)+S \tag{5.2.5}$$

The previous equation can be rewritten in Cartesian coordinates for incompressible flow and constant physical properties:

$$\rho\frac{\partial\phi}{\partial t}+\rho v\frac{\partial\phi}{\partial x}+\rho v\frac{\partial\phi}{\partial y}=\Gamma\left(\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}\right)+S \tag{5.2.6}$$

The first term is the accumulation of ϕ along time, which gives information about the temporal variation of ϕ. The second and the third ones are the net convective flow in the control volume, which gives information about spatial transport of ϕ. This sum of these terms has to be equal to the net diffusive flow (fourth term), which represents the transport of ϕ due to the concentration of gradients, plus the generation of ϕ per unit volume (fifth term), which can be a source or a sump.

According to the convection diffusion equation (Equation 5.2.5), it can be possible to write a table with the appropriate parameters of ϕ, Γ and S in order to reproduce the governing equations (Equation 5.2.1, Equation 5.2.2, Equation 5.2.3, Equation 5.2.4). See Table 5.2.1.

Equation	ϕ	Γ	S
Continuity	1	0	0
Momentum in x direction	υ	μ	$-{}^{\partial p_d}/_{\partial x}$
Momentum in y direction	ν	μ	$-{}^{\partial p_d}/_{\partial y} + \rho g \beta \left(T - T_\infty\right)$
Energy (constant C_p)	T	${}^{\lambda}/_{C_p}$	${}^{\Phi}/_{C_p}$

Table 5.2.1.: Parameters to replace in convection-diffusion equation to obtain the governing equations

5.3. Implicit finite-volume discretization of convection-diffusion equations

Integrating the convection-diffusion equation into a rectangular finite volume (FVM)[31] (see Figure 5.3.1), the discretized equation can be written as:

$$\frac{(\rho\phi)_P^{n+1} - (\rho\phi)_P^{n}}{\Delta t} V_p + \left[(\rho\upsilon\phi)_e^{n+1} S_e - (\rho\upsilon\phi)_w^{n+1} S_w\right] + \\ + \left[(\rho\upsilon\phi)_n^{n+1} S_n - (\rho\upsilon\phi)_s^{n+1} S_s\right] = \left[\left(\Gamma\frac{\partial\phi}{\partial x}\right)_e^{n+1} S_e - \left(\Gamma\frac{\partial\phi}{\partial x}\right)_w^{n+1} S_w\right] + \\ + \left[\left(\Gamma\frac{\partial\phi}{\partial y}\right)_n^{n+1} S_n - \left(\Gamma\frac{\partial\phi}{\partial y}\right)_s^{n+1} S_s\right] + S_P^{n+1} V_p \quad (5.3.1)$$

The Equation 5.2.5 can be simplified using the total flux term, which is defined by $J_x = \rho\upsilon\phi - \Gamma\frac{\partial\phi}{\partial x}$ and $J_y = \rho\upsilon\phi - \Gamma\frac{\partial\phi}{\partial y}$. Therefore, it yields

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} = S \quad (5.3.2)$$

Integrating the previous equation into a rectangular finite volume by assuming an implicit scheme for the temporal integration, see Figure 5.3.1:

$$\frac{(\rho\phi)_P^{n+1} - (\rho\phi)_P^{n}}{\Delta t} V_p \quad + \quad \left[J_e - J_w + J_n - J_s\right] e \quad = \quad \left(S_c + S_p \phi_P\right)^{n+1} V_p \quad (5.3.3)$$

The continuity equation is introduced into the system of equations in order to assure convergence:

$$\frac{\partial\rho}{\partial t} + \frac{\partial\rho\upsilon}{\partial x} + \frac{\partial\rho\upsilon}{\partial y} = 0 \quad (5.3.4)$$

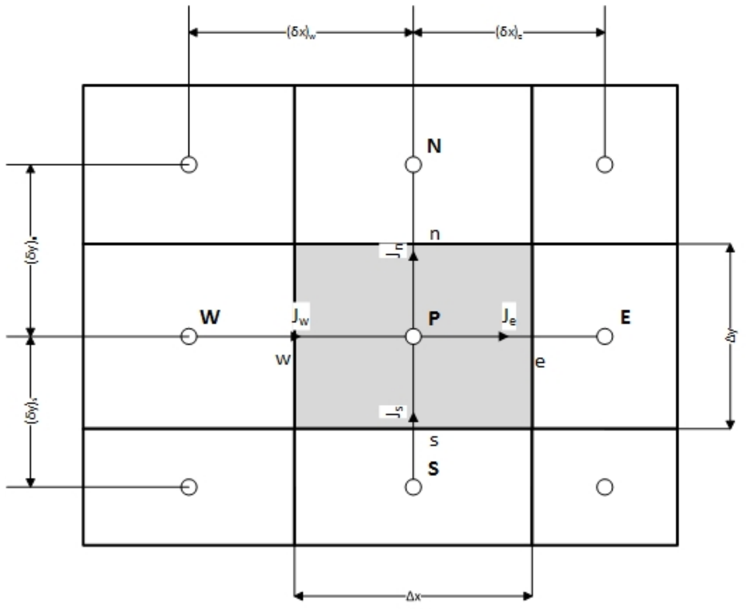


Figure 5.3.1.: Control volume for 2D with J flux

Integrating the previous equation into a rectangular finite volume:

$$\frac{\rho_P^{n+1} - \rho_P^n}{\Delta t} V_p + F_e - F_w + F_n - F_s = 0 \tag{5.3.5}$$

where F indicates the mass flow rate through the faces of the control volume and it can be obtained by $F_w = (\rho v)_w S_w$, $F_e = (\rho v)_e S_e$, $F_s = (\rho v)_s S_s$ and $F_n = (\rho v)_n S_n$.

A combination of equations Equation 5.3.3 and Equation 5.3.5 $\cdot \phi_P$ yields:

$$\frac{(\phi_P^1 - \phi_P^0)\,\rho_P^0}{\Delta t} V_P + (J_e e - F_e \phi_P) - (J_w e - F_w \phi_P) + \\ + (J_n e - F_n \phi_P) - (J_n e - F_n \phi_P) = (S_c + S_p \phi_P)^{n+1} V_p \tag{5.3.6}$$

Note that the hypothesis of uniformity over a control-volume face has been done.

5.4. Numerical Schemes

These numerical schemes are used to evaluate the convective and diffusive terms at the cell faces. The conductive flux $\frac{\partial \phi}{\partial x}$ or $\frac{\partial \phi}{\partial y}$ on any face is calculated as an arithmetic mean between central nodal values:

$$\left(\frac{\partial \phi}{\partial x}\right)_w = \frac{\phi_W - \phi_P}{\delta x_w} \tag{5.4.1}$$

$$\left(\frac{\partial \phi}{\partial x}\right)_e = \frac{\phi_E - \phi_P}{\delta x_e} \tag{5.4.2}$$

$$\left(\frac{\partial \phi}{\partial x}\right)_s = \frac{\phi_S - \phi_P}{\delta x_s} \tag{5.4.3}$$

$$\left(\frac{\partial \phi}{\partial x}\right)_n = \frac{\phi_N - \phi_P}{\delta x_n} \tag{5.4.4}$$

Otherwise, the convective and diffusive terms have to be calculated using numerical schemes with values of ϕ at the nodal points. There are two types of schemes: low order numerical schemes which use the nearest nodes (W, E, S and N); and high order numerical schemes which use more than two nodal values to calculate the variables at the cell faces.

The low order numerical schemes are: CDS, UDS, HDS, EDS, PLDS, and so on. The approximation of the value for a Peclet number can be seen in Figure 5.4.1. For more information about high order numerical schemes, see [5].

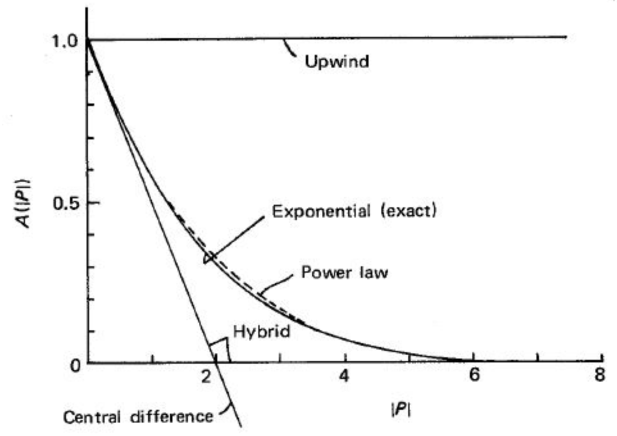


Figure 5.4.1.: The function $A\left(|Pe_{nf}|\right)$ for various low order schemes [31]

5.4.1. Central Difference Scheme (CDS)

As it has been explained in subsection 2.3.1, it is a second order scheme, and the variable at the faces should be interpolated between grid nodal values.

$$\phi_{nf} = \frac{1}{2}\left(\phi_P + \phi_{nb}\right) \tag{5.4.5}$$

This scheme has a disadvantage because F may be negative, what would cause some coefficients to become negative. Because of that, the Scarborough criterion will not be satisfied.

The function $A\left(|Pe_{nf}|\right)$ will be equal to $A\left(|Pe_{nf}|\right) = 1 - 0.5\left(|Pe_{nf}|\right)$ where the value of the Peclet number at the interface will be calculated by using a CDS scheme with the central node and the corresponding neighbouring node.

5.4.2. Upwind Difference Scheme (UDS)

As it has been explained in subsection 2.3.2, it is a first order scheme and can be computed as:

$$\phi_e = \phi_P \quad if \quad F_e > 0 \tag{5.4.6}$$

$$\phi_e = \phi_E \quad if \quad F_e < 0 \tag{5.4.7}$$

For that reason, the operator max is defined as $[\![A, B]\!]$. Thus,

$$F_e \phi_e = \phi_P [\![F_e, 0]\!] - \phi_E [\![-F_e, 0]\!] \tag{5.4.8}$$

This scheme works out the problem that the CDS scheme has, because all the coefficients are always positive or null. Therefore, the Scarborough criterion will always be satisfied.

The function $A\left(|Pe_{nf}|\right)$ will be equal to $A\left(|Pe_{nf}|\right) = 1$.

5.4.3. Hybrid Difference Scheme (HDS)

This scheme uses the CDS scheme for low velocities and the UDS scheme for high velocities. It consists of three linear zones to approximate the value of $\frac{a_E}{D_e} = \frac{Pe_e}{exp(Pe_e)-1}$. The approximation is shown in Figure 5.4.2.

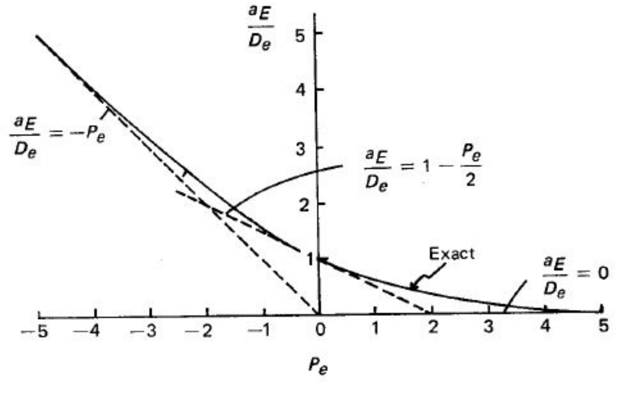


Figure 5.4.2.: Variation of the coefficient a_E with Peclet number [31]

Thus, the coefficient can be approximated by the next function:

$$a_E = D_e \left[\!\!\left[-Pe_e \, , \, 1 - \frac{Pe_e}{2} \, , \, 0 \right]\!\!\right] \tag{5.4.9}$$

Its behaviour can be compared to that of a CDS in the central range $Pe_e = [-2\, ,\, 2]$, while outside this range it is determined by UDS in which the diffusion is settled to 0. For that reason, the function $A\left(|Pe_{nf}|\right)$ will be equal to $A\left(|Pe_{nf}|\right) = max\left(0\, ,\, \left(1 - 0.5\left(|Pe_{nf}|\right)\right)\right)$.

5.4.4. Exponential Difference Scheme (EDS)

As it has been explained in subsection 2.3.3, it is a second order scheme and the evaluation of the variables at the cell faces comes from the exact solution of the Equation 5.2.5 for steady one-dimensional problem without source term. The solution for the one-dimensional problem is:

$$\frac{\phi - \phi_0}{\phi_L - \phi_0} = \frac{exp\left(Pe\, x/L\right) - 1}{exp\left(Pe\right) - 1} \tag{5.4.10}$$

Where ϕ_0 is the value of ϕ at the left boundary; ϕ_L is the value of ϕ at the right boundary; Pe is the Peclet number; and L is the distance of the domain.

For that reason, the function $A\left(|Pe_{nf}|\right)$ will be equal to $A\left(|Pe_{nf}|\right) = \left(|Pe|\right) / \left(e^{\left(|Pe|\right)} - 1\right)$.

5.4.5. Power-law Difference Scheme (PLDS)

It is a second order scheme and the evaluation of the variables at the cell faces comes from an approximation of EDS by a fifth degree polynomial. It provides a good representation of the exponential behaviour with not so much computer requirements.

The PLDS can be summarized as:

$$a_E = D_e \left[\!\!\left[0\, ,\, \left(1 - \frac{0.1|F_e|}{D_e}\right)^5 \right]\!\!\right] + [\![0\, ,\, -F_e]\!] \tag{5.4.11}$$

The function $A\left(|Pe_{nf}|\right)$ will be equal to $A\left(|Pe_{nf}|\right) = max\left(0\, ,\, \left(1 - 0.5\left(|Pe_{nf}|\right)\right)^5\right)$.

5.5. Linear discretization of Convective equation

According to [31, 5], a general linear equation can be obtained for Equation 5.3.6 by expressing the terms $\left(J_e e - F_e \phi_P\right)$ and $\left(J_w e - F_w \phi_P\right)$ as $a_E\left(\phi_P - \phi_E\right)$ and $a_W\left(\phi_W - \phi_P\right)$, respectively. Similar expressions apply for y-axis.

$$a_p\phi_P = a_E\phi_E + a_S\phi_S + a_W\phi_W + a_N\phi_N + b \tag{5.5.1}$$

The coefficients a_i can be evaluated with:

$$\begin{aligned}
a_E &= D_e \cdot A\left(|Pe_e|\right) + max\left(-F_e\,,\,0\right) && (5.5.2)\\
a_W &= D_w \cdot A\left(|Pe_w|\right) + max\left(F_w\,,\,0\right) && (5.5.3)\\
a_N &= D_n \cdot A\left(|Pe_n|\right) + max\left(-F_n\,,\,0\right) && (5.5.4)\\
a_S &= D_s \cdot A\left(|Pe_s|\right) + max\left(F_s\,,\,0\right) && (5.5.5)\\
a_P^0 &= \frac{\rho_P^0 V_p}{\Delta t} && (5.5.6)\\
b &= S_C^{n+1} V_P + a_P^0 \phi_P^0 && (5.5.7)\\
a_P &= a_E + a_S + a_W + a_N + a_P^0 - S_P^{n+1} V_P && (5.5.8)
\end{aligned}$$

Where

$$D_e = \frac{\Gamma_e S_e}{(\delta x)_e} \qquad D_w = \frac{\Gamma_w S_w}{(\delta x)_w} \qquad D_n = \frac{\Gamma_n S_n}{(\delta y)_n} \qquad D_s = \frac{\Gamma_s S_s}{(\delta y)_s} \tag{5.5.9}$$

$$F_e = (\rho v)_e S_e \qquad F_w = (\rho v)_w S_w \qquad F_n = (\rho v)_n S_n \qquad F_s = (\rho v)_s S_s \tag{5.5.10}$$

$$Pe_e = \frac{F_e}{D_e} \qquad Pe_w = \frac{F_w}{D_w} \qquad Pe_n = \frac{F_n}{D_n} \qquad Pe_s = \frac{F_s}{D_s} \tag{5.5.11}$$

Note that the value of $A\left(|Pe_{nf}|\right)$ depends on the numerical scheme used. It can be seen in Table 5.5.1:

Low numerical scheme	$A\left(\vert Pe_{nf}\vert\right)$
CDS	$1 - 0.5\left(\vert Pe\vert\right)$
UDS	1
HDS	$max\left(0\,,\,\left(1 - 0.5\left(\vert Pe\vert\right)\right)\right)$
EDS	$\left(\vert Pe\vert\right) / \left(e^{\left(\vert Pe\vert\right)} - 1\right)$
PLDS	$max\left(0\,,\,\left(1 - 0.5\left(\vert Pe\vert\right)\right)^5\right)$

Table 5.5.1.: Value of $A\left(|Pe_{nf}|\right)$ for the low numerical schemes

6. Convection-Diffusion problems

The study of 4 cases related to the Convection-Diffusion equation (Equation 5.2.5) is about to be undertaken. The variables ϕ , Γ and S are going to be general variables, so it will satisfy mass conservation equation, momentum conservation equation and energy conservation equation.

6.1. Unidimensional flow with a unidimensional variation of the variable solved in the same direction of the flow

6.1.1. Introduction

It is a steady convection-diffusion problem the analytical solution of which is known when a velocity field is given. The solution is an exponential function for an arbitrary value of the velocity field. See Figure 6.1.1.

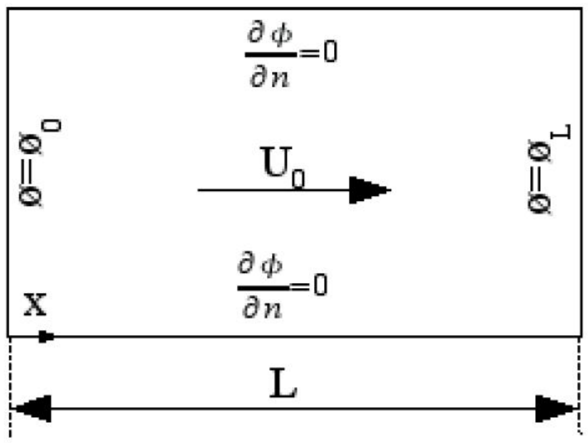

$\frac{\partial \phi}{\partial n}=0$
$\phi=\phi_0$
U_0
$\phi=\phi_L$
$\frac{\partial \phi}{\partial n}=0$
X
L

Figure 6.1.1.: Unidimensional flow with a unidimensional variation of the variable solved in the same direction of the flow

6.1.2. Boundary conditions

The boundary conditions are defined below

- Initial ϕ value: $\phi = 280.0$.
- Left boundary condition: $\phi_0 = 273.15$.
- Right boundary condition: $\phi_L = 350.15$.
- Bottom boundary condition: zero normal convective flow ($\partial\phi/\partial y = 0$).
- Top boundary condition: zero normal convective flow ($\partial\phi/\partial y = 0$).
- Velocity field: $\upsilon(x, y) = U_0 = 0.001\,\mathrm{m/s}$ $\nu(x, y) = 0$.

6.1.3. Spatial discretization

The spatial discretization is shown in Table 6.1.1.

Spatial property	L	H	N_x	N_y	Δx	Δy
Value	0.1	0.1	200	2	0.0005	0.05

Table 6.1.1.: Spatial discretization for unidimensional flow in x-axis

6.1.4. The governing equations

The equations used to solve the problem are shown in Equation 5.5.1. The scheme chosen to solve the convection-diffusion problem is an implicit scheme because it returns a physically satisfactory behaviour. On the other hand, the chosen solver is a Gauss-Seidel method due to its simplicity.

The Peclet number can be computed using the previous values and imaging that the energy equation is going to be solved for air. See properties of air: Table 6.1.2.

Physical property	μ (kg/ms)	C_p (J/KgK)	ρ (Kg/m^3)	λ (W/mK)	Φ (J/m^3)
Value	2.075e-5	1.009e3	0.9980	0.03003	0.0000

Table 6.1.2.: Air properties for the convection-diffusion problem at 50ºC

$$Pe = \frac{\rho \upsilon L}{\Gamma} = \frac{\rho \upsilon L}{\lambda / C_p} = 3.3533 \tag{6.1.1}$$

The analytic solution is shown below

$$\frac{\phi - \phi_0}{\phi_L - \phi_0} = \frac{\exp\left(Pe\, x/L\right) - 1}{\exp\left(Pe\right) - 1} \tag{6.1.2}$$

6.1.5. The program

The program creates a structure with all the information related to each grid node. Therefore, it contains the ϕ values, the coordinate values of the grid nodes and interface nodes, the physical properties, the velocity values of the interface nodes, Γ_{nf}, $\mathrm{D_{nf}}$, $\mathrm{F_{nf}}$ and $\mathrm{Pe_{nf}}$ and the coefficients of Equation 5.5.1.

The inputs required by the program are the point coordinates, the spatial step and the number of nodes in each direction, the convergence criterion, the coefficient to select the scheme, the width of the volume, the ϕ value for the left boundary condition, the ϕ value for the right boundary condition and, finally, the initial ϕ field. Moreover, the program needs the physical properties and the velocity field (norm of the velocity) to calculate the Peclet number. Another requirement needed by the program is the last time to solve the time loop.

The program consists of a variety of files which calculate different parts of the equations. The functions are explained below

- Main file: it contains the general algorithm that calls the different subfiles to perform its function inside the algorithm. See Figure 6.1.2. It also exports a file with the comparison between exact solution and calculated solution.
- Geometrical properties file: it calculates the geometrical properties of each grid node and its interfaces by using a uniform, orthogonal mesh. This file also calculates the velocity and density of each interface node and the specific heat of each grid node.
- Equation selection file: it selects the equation to be solved between the continuity equation, momentum in x direction equation, momentum in y direction equation and energy equation. It also computes the values for Γ_{nf} and the source term of the convection diffusion equation.
- Convection-Diffusion parameters file: it determines the $\mathrm{D_{nf}}$, $\mathrm{F_{nf}}$ and $\mathrm{Pe_{nf}}$ for each interface node. Note that there are 9 different situations.
- Coefficients file: it calculates the coefficients taking into account the different conditions of each node. In this case there are also 9 different situations. This code also solves the Equation 5.5.1.

6.1.6. The results

After the code was created, it was validated with the ϕ along x coordinate points to make sure that everything was correct. The results for the ϕ field when it converged

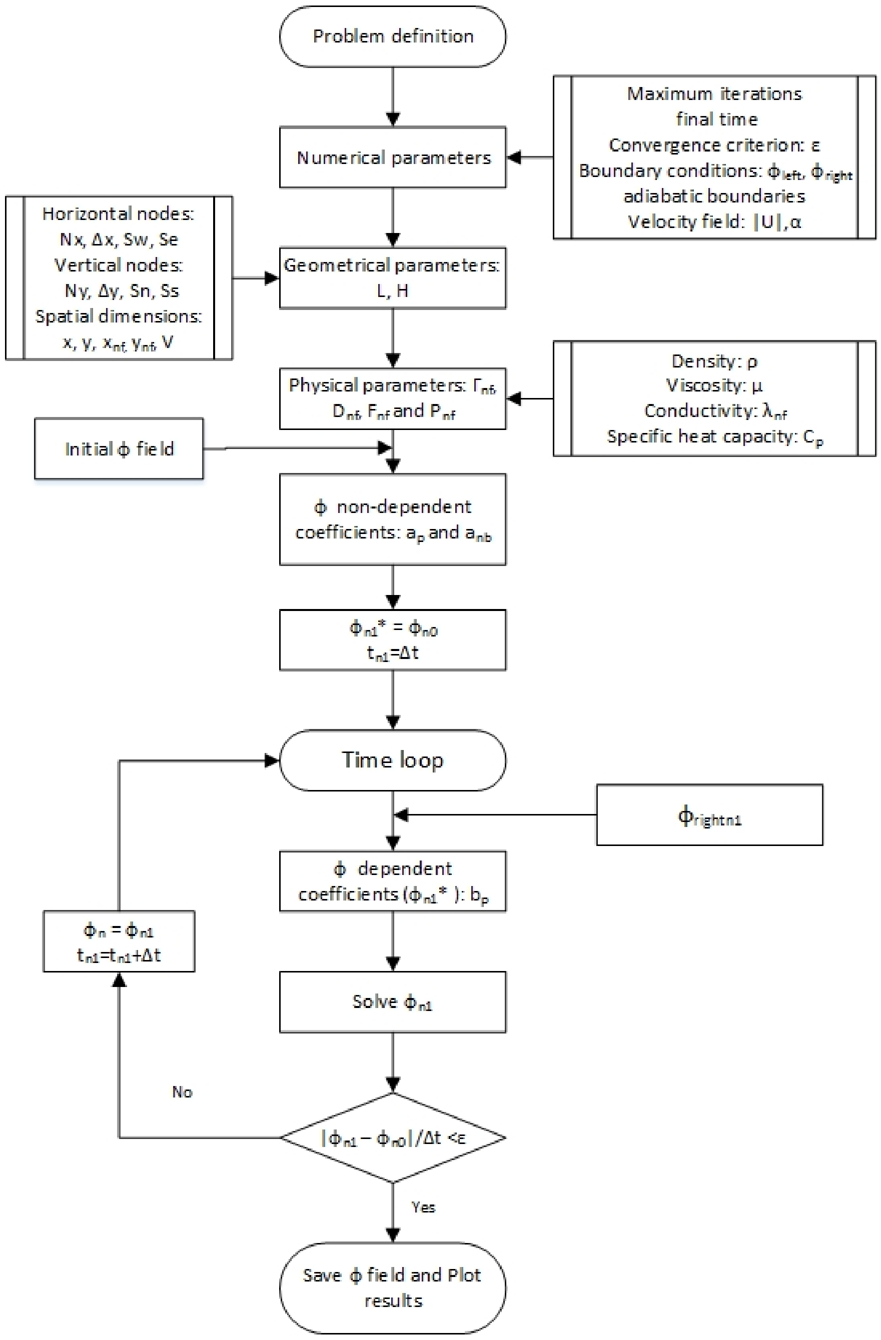


Figure 6.1.2.: Algorithm used in the specific case: x-axis

are shown below (Figure 6.1.3), and the calculated solution can be compared with the analytical solution (Figure 6.1.4). Note that both solution's curves are almost the same, so it can be concluded that the program is correctly written.

6.1.7. Conclusions

The solution obtained from the own code is unidimensional, as expected. The values of the variable ϕ on the boundaries are $\phi_0 = 273.15$ and $\phi_L = 350.15$, as expected.

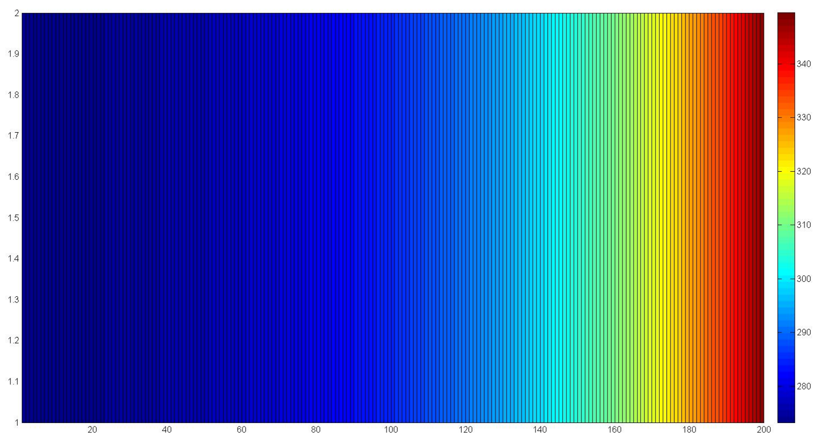


Figure 6.1.3.: ϕ field for unidimensional flow in X-direction

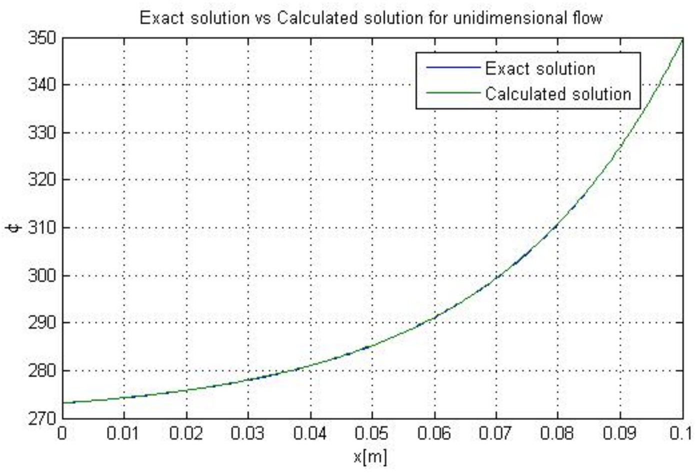


Figure 6.1.4.: Comparison between exact solution and calculated solution for unidimensional flow in X-direction

Along the x axis there is a smooth ϕ distribution.

Finally, it should be mentioned that the convective value F is bigger than the diffusive term D. Therefore, the upwind ϕ value – in that case the left ϕ boundary value, covers more space than the left boundary value.

6.2. Unidimensional flow with a unidimensional variation of the variable solved in the perpendicular direction of the flow

6.2.1. Introduction

It is a steady convection-diffusion problem the analytical solution of which is known when a velocity field is given. This situation is the same as conduction in a moving solid, independently of the velocity field. See Figure 6.2.1.

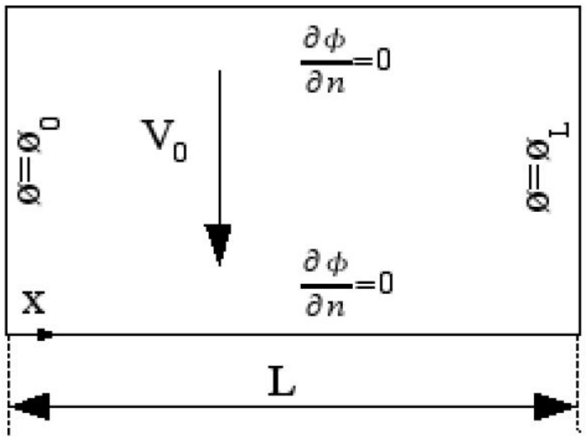

$\frac{\partial \phi}{\partial n}=0$
$\varnothing=\varnothing_0$
V_0
$\varnothing=\varnothing_L$
$\frac{\partial \phi}{\partial n}=0$
X
L

Figure 6.2.1.: Unidimensional flow with a unidimensional variation of the variable solved in the perpendicular direction of the flow

6.2.2. Boundary conditions

The boundary conditions are defined below

- Initial ϕ value: $\phi = 280.0$.
- Left boundary condition: $\phi_0 = 273.15$.
- Right boundary condition: $\phi_L = 350.15$.
- Bottom boundary condition: zero normal convective flow ($\partial\phi/\partial y = 0$).
- Top boundary condition: zero normal convective flow ($\partial\phi/\partial y = 0$).
- Velocity field: $\upsilon(x, y) = 0$ $\nu(x, y) = V_0 = 0.001$m/s.

6.2.3. Spatial discretization

The spatial discretization is shown in Table 6.2.1.

Spatial property	L	H	N_x	N_y	Δx	Δy
Value	0.1	0.1	200	2	0.0005	0.05

Table 6.2.1.: Spatial discretization for unidimensional flow in y-axis

6.2.4. The governing equations

The equations used to solve the problem are shown in Equation 5.5.1. The scheme chosen to solve the convection-diffusion problem is an implicit scheme because it returns a physically satisfactory behaviour. On the other hand, the solver chosen is a Gauss-Seidel method because of its simplicity.

The Peclet number can be computed using the previous values and imaging that the energy equation is going to be solved for air. See properties of air: Table 6.1.2.

$$Pe = \frac{\rho \nu L}{\Gamma} = \frac{\rho \nu L}{\lambda / C_p} = 3.3533 \tag{6.2.1}$$

The analytic solution is shown below

$$\phi = \phi_0 + \frac{\phi_L - \phi_0}{L} x \tag{6.2.2}$$

6.2.5. The program

The program creates a structure with all the information related to each grid node. Therefore, it contains the ϕ values, the coordinate values of the grid nodes and interface nodes, the physical properties, the velocity values of the interface nodes, $\mathit{\Gamma}_{nf}$, D_{nf}, F_{nf} and Pe_{nf} and the coefficients of Equation 5.5.1.

The inputs required by the program are the point coordinates, the spatial step and number of nodes in each direction, the convergence criterion, the coefficient to select the scheme, the width of the volume, the ϕ value for the left boundary condition, the ϕ value for the right boundary condition and, finally, the initial ϕ field. Moreover, the program needs the physical properties and the velocity field (norm of the velocity) to calculate the Peclet number. Another requirement needed by the program is the last time to solve the time loop.

The program consists in a variety of files which calculate different parts of the equations. The functions are explained below

- Main file: it contains the general algorithm that calls the different subfiles to perform its function inside the algorithm. See Figure 6.1.2. It also exports a file with the comparison between exact solution and calculated solution.

- Geometrical properties file: it calculates the geometrical properties of each grid node and its interfaces using a uniform, orthogonal mesh. This file also calculates the velocity and density of each interface node and the specific heat of each grid node.
- Equation selection file: it selects the equation to be solved between the continuity equation, momentum in x direction equation, momentum in y direction equation and energy equation. It also calculates the values for Γ_{nf} and the source term of the convection diffusion equation.
- Convection-Diffusion parameters file: it determines the $\mathrm{D_{nf}}$, $\mathrm{F_{nf}}$ and $\mathrm{Pe_{nf}}$ for each interface node. Note that there are 9 different situations.
- Coefficients file: it calculates the coefficients taking into account the different conditions of each node. In this case, there are also 9 different situations. This code also solves the Equation 5.5.1.

6.2.6. The results

After the code was created, it was validated with the ϕ along x coordinate points to make sure that everything was correct. The results for the ϕ field when it converged are shown below (Figure 6.2.2), and the calculated solution can be compared with the analytical solution (Figure 6.2.3). Note that both solution's curves are practically the same, so it can be concluded that the program is correctly written.

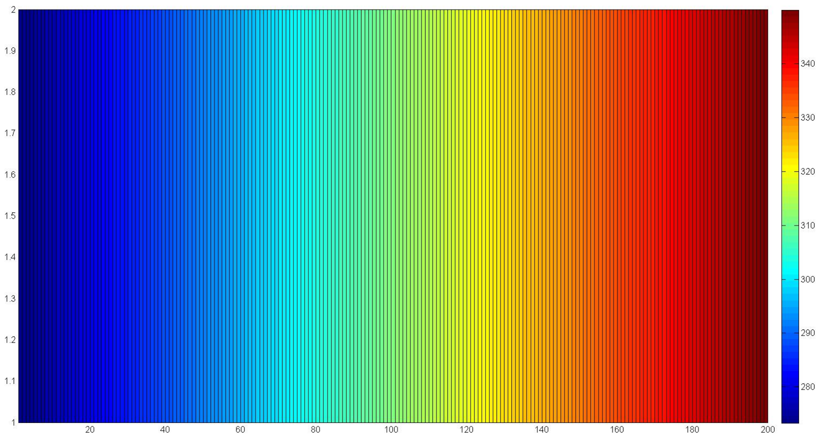


Figure 6.2.2.: ϕ field for unidimensional flow in Y-direction

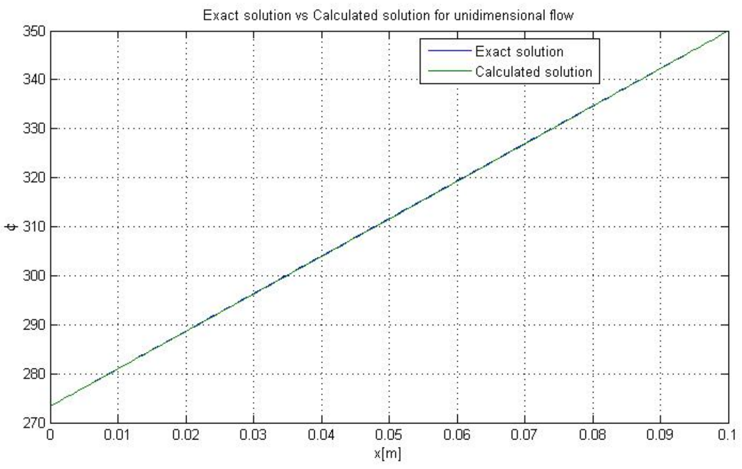


Figure 6.2.3.: Comparison between exact solution and calculated solution for unidimensional flow in y-direction

6.2.7. Conclusions

The solution obtained from the own code is unidimensional, as expected along the x axis. The values of the variable ϕ on the boundaries are $\phi_0 = 273.15$ and $\phi_L = 350.15$, as expected. Along the x direction, there is a linear ϕ distribution.

Finally, it should be mentioned that the convective value F in the x direction is equal to 0, so the only no-null term is the diffusive term D. Therefore, the ϕ field is equivalent to the solution that would be obtained in a conductive problem.

6.3. Diagonal flow

6.3.1. Introduction

It is a steady convection-diffusion problem the solution of which is known for an infinite total Peclet number when the flow is in the main diagonal and the boundary conditions of the dependent variable are the ones indicated in Figure 6.3.1.

6.3.2. Boundary conditions

The boundary conditions are defined below

- Initial ϕ value: $\phi = 280.0$.
- Left boundary condition: $\phi_1 = 273.15$.
- Right boundary condition: $\phi_2 = 350.15$.

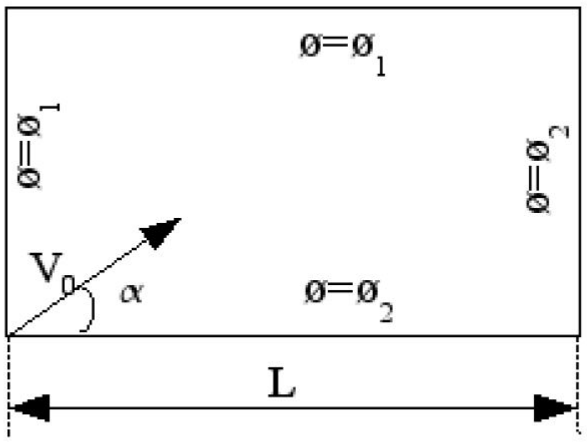


Figure 6.3.1.: Diagonal flow

- Bottom boundary condition: $\phi_2 = 350.15$.
- Top boundary condition: $\phi_1 = 273.15$.
- Velocity field: $\upsilon(x,y) = V_0 \cos(\alpha) = 10 \cdot \cos(45^{\text{o}}) = 7.0711 \mathrm{m/s}$ and $\nu(x,y) = V_0 \sin(\alpha) = 10 \cdot \sin(45^{\text{o}}) = 7.0711 \mathrm{m/s}$.

6.3.3. Spatial discretization

The spatial discretization is shown in Table 6.3.1.

Spatial property	L	H	N$_x$	N$_y$	Δx	Δy
Value	0.1	0.1	200	200	0.0005	0.0005

Table 6.3.1.: Spatial discretization for unidimensional flow in diagonal-axis

6.3.4. The governing equations

The equations used to solve the problem are shown in Equation 5.5.1. The scheme chosen to solve the convection-diffusion problem is an implicit scheme because it returns a physically satisfactory behaviour. On the other hand, the solver chosen is a Gauss-Seidel method due to its simplicity.

The Peclet number can be computed using the previous values and imaging that the energy equation is going to be solved for air. See properties of air: Table 6.1.2.

$$Pe_i = \frac{\rho u_i L}{\Gamma} = \frac{\rho \nu L}{\lambda / C_p} = 23711.0823 \tag{6.3.1}$$

This Peclet number is big enough to obtain the phenomenon of false diffusion in the exposed situation.

6.3.5. The program

The program creates a structure with all the information related to each grid node. Therefore, it contains the ϕ values, the coordinate values of the grid nodes and interface nodes, the physical properties, the velocity values of the interface nodes, Γ_{nf}, $\mathrm{D_{nf}}$, $\mathrm{F_{nf}}$ and $\mathrm{Pe_{nf}}$ and the coefficients of Equation 5.5.1.

The inputs required by the program are the point coordinates, the spatial step and number of nodes in each direction, the convergence criterion, the coefficient to select the scheme, the width of the volume, the ϕ value for the left and top boundary condition, the ϕ value for the right and bottom boundary condition and finally the initial ϕ field. Moreover, the program needs the physical properties and the velocity field (norm of the velocity and angle of incidence) to calculate the Peclet number. Another requirement needed by the program is the last time to solve the time loop.

The program consists in a variety of files which calculate different parts of the equations. The functions are explained below

- Main file: it contains the general algorithm that calls the different subfiles to perform its function inside the algorithm. See Figure 6.3.2. It also exports a file with the comparison between exact solution and calculated solution.
- Geometrical properties file: it computes the geometrical properties of each grid node and its interfaces using a uniform, orthogonal mesh. This file also calculates the velocity and density of each interface node and the specific heat of each grid node.
- Equation selection file: it selects the equation to be solved between the continuity equation, momentum in x direction equation, momentum in y direction equation and energy equation. It also calculates the values for Γ_{nf} and the source term of the convection diffusion equation.
- Convection-Diffusion parameters file: it determines the $\mathrm{D_{nf}}$, $\mathrm{F_{nf}}$ and $\mathrm{Pe_{nf}}$ for each interface node. Note that there are 9 different situations.
- Coefficients file: it calculates the coefficients taking into account the different conditions of each node. In this case, there are also 9 different situations. This code also solves the Equation 5.5.1.

6.3.6. The results

After the code was created, it was validated with the ϕ field to make sure that everything was correct. The results for the ϕ field when it converged are shown

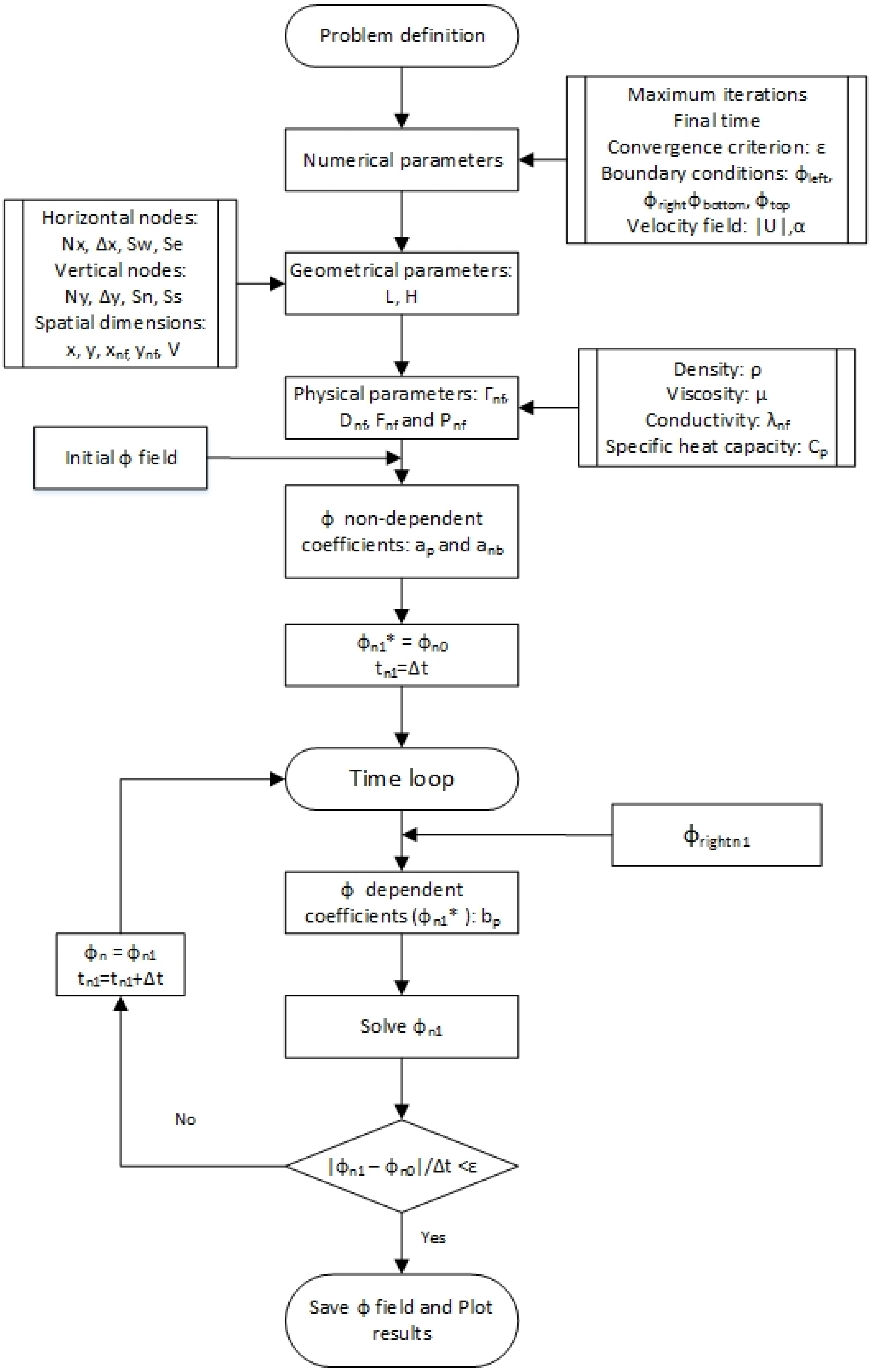

Problem definition
Numerical parameters
Maximum iterations
Final time
Convergence criterion: ε
Boundary conditions: ϕ_{left}, ϕ_{right} ϕ_{bottom}, ϕ_{top}
Velocity field: |U|,α
Horizontal nodes: Nx, Δx, Sw, Se
Vertical nodes: Ny, Δy, Sn, Ss
Spatial dimensions: x, y, x_{nf}, y_{nf}, V
Geometrical parameters: L, H
Physical parameters: Γ_{nf}, D_{nf}, F_{nf} and P_{nf}
Density: ρ
Viscosity: μ
Conductivity: λ_{nf}
Specific heat capacity: C_p
Initial ϕ field
ϕ non-dependent coefficients: a_p and a_{nb}
$\phi_{n1}^* = \phi_{n0}$
$t_{n1}=\Delta t$
Time loop
$\phi_{right\,n1}$
ϕ dependent coefficients (ϕ_{n1}^*): b_p
Solve ϕ_{n1}
$|\phi_{n1} - \phi_{n0}|/\Delta t < \varepsilon$
No
Yes
$\phi_n = \phi_{n1}$
$t_{n1}=t_{n1}+\Delta t$
Save ϕ field and Plot results

Figure 6.3.2.: Algorithm used in the specific case: false diffusion

below (Figure 6.3.3), and the calculated solution can be compared with the expected solution (Figure 6.2.3).

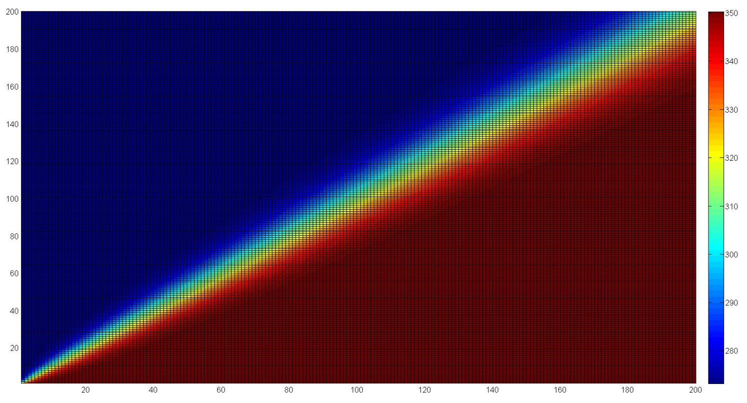


Figure 6.3.3.: ϕ field for unidimensional flow in diagonal-axis

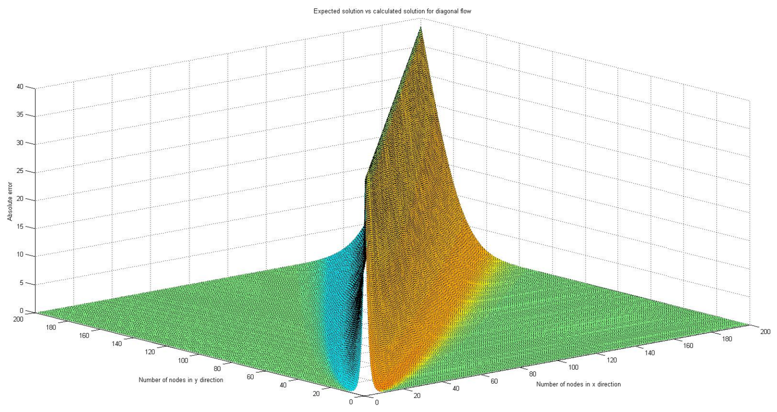


Figure 6.3.4.: Comparison between exact solution and calculated solution for unidimensional flow in diagonal-direction

6.3.7. Conclusions

The results show a case of false diffusion, which happens when the convective term F is large enough, so that its effect predominates on the diffusion effect D. Therefore, the ideally ϕ field corresponds only to two streams with different values, equals to the values of the boundaries.

The discrepancies with respect to the ideal case are on the boundary between the two streams, where there should not be a mixed layer. This divergence is due to the fact that the convective term is not mainly predominant with respect to the diffusive term. It may also be caused by the scheme used to solve the equations.

6.4. Solenoidal flow

6.4.1. Introduction

It is a steady convection-diffusion problem the solution of which is known when a velocity field is given. See Figure 6.4.1.

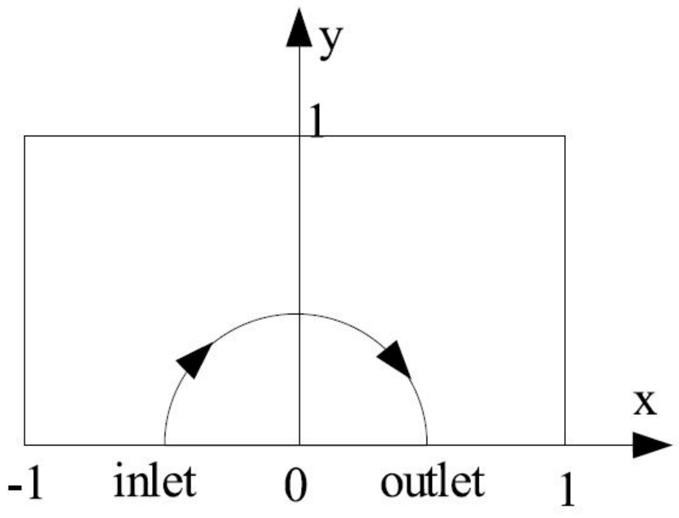


Figure 6.4.1.: Solenoidal flow

6.4.2. Boundary conditions

The boundary conditions are defined below

- Initial ϕ value: $\phi = 1.0$.
- Left boundary condition: $\phi = 1 - \tanh(\alpha)$.
- Right boundary condition: $\phi = 1 - \tanh(\alpha)$.
- Inlet boundary condition: $\phi = 1 + \tanh\left[(2x+1)\,\alpha\right]$.
- Outlet boundary condition: $\partial\phi/\partial y = 0$.
- Top boundary condition: $\phi = 1 - \tanh(\alpha)$.
- Velocity field: $\upsilon(x,y) = 2y\left(1-x^2\right)$ and $\nu(x,y) = -2x\left(1-y^2\right)$.

Where $\alpha = 10$.

6.4.3. Spatial discretization

The spatial discretization is shown in Table 6.4.1.

Spatial property	L	H	N_x	N_y	Δx	Δy
Value	$[-1, 1]$	1	200	200	0.01	0.005

Table 6.4.1.: Spatial discretization for solenoidal flow in Smith-Hutton problem

6.4.4. The governing equations

The equations used to solve the problem are shown in Equation 5.5.1. The scheme chosen to solve the convection-diffusion problem is an implicit scheme because it returns a physically satisfactory behaviour. On the other hand, the solver chosen is a Gauss-Seidel method due to its simplicity.

The Peclet number can be computed using the previous values and imaging that the energy equation is going to be solved for an ideal fluid.

$$Pe_i = \frac{\rho u_i L}{\Gamma} \tag{6.4.1}$$

6.4.5. The program

The program creates a structure with all the information related to each grid node. Therefore, it contains the ϕ values, the coordinate values of the grid nodes and interface nodes, the physical properties, the velocity values of the interface nodes, $\mathit{\Gamma}_{nf}$, $\mathrm{D_{nf}}$, $\mathrm{F_{nf}}$ and $\mathrm{Pe_{nf}}$ and the coefficients of Equation 5.5.1.

The inputs required by the program are the point coordinates, the spatial step and number of nodes in each direction, the convergence criterion, the coefficient to select the scheme, the width of the volume, t and, finally, the initial ϕ field. Moreover, the program needs the physical properties. Another requirement needed by the program is the last time to solve the time loop.

The program consists in a variety of files which calculate different parts of the equations. The functions are explained below

- Main file: it contains the general algorithm that calls the different subfiles to perform its function inside the algorithm. See Figure 6.4.2.
- Geometrical properties file: it calculates the geometrical properties of each grid node and its interfaces using a uniform, orthogonal mesh. This file also computes the velocity of each grid node.
- Equation selection file: it selects the equation to be solved between the continuity equation, momentum in x direction equation, momentum in y direction equation and energy equation. It also calculates the values for $\mathit{\Gamma}_{nf}$ and the source term of the convection diffusion equation.

- Convection-Diffusion parameters file: it determines the D_{nf}, F_{nf} and Pe_{nf} for each interface node. Note that there are 9 different situations.
- Coefficients file: it calculates the coefficients taking into account the different conditions of each node. In this case, there are also 9 different situations. This code also solves the Equation 5.5.1.
- ϕ field file: it stores the entire ϕ field in a txt file when it has converged. These values will be useful to validate the code.

6.4.6. The results

After the code was created, it was validated with the ϕ field to make sure that everything was correct. The results for the ϕ field when it converged are shown below (Figure 6.4.3, Figure 6.4.4 and Figure 6.4.5), and the calculated solution can be compared with the expected solution (Table 6.4.2).

	$\rho/\Gamma = 10$		$\rho/\Gamma = 1000$		$\rho/\Gamma = 1000000$	
Position x	Expected	Calculated	Expected	Calculated	Expected	Calculated
0.0	1.989	1.901	2.0000	2.0000	2.000	2.000
0.1	1.402	1.428	1.9990	2.0000	2.000	2.000
0.2	1.146	1.190	1.9997	2.0000	2.000	2.000
0.3	0.946	0.999	1.9850	1.9885	1.999	1.991
0.4	0.775	0.833	1.8410	1.7775	1.964	1.790
0.5	0.621	0.679	0.9510	0.9465	1.000	0.947
0.6	0.480	0.534	0.1540	0.1915	0.036	0.183
0.7	0.349	0.394	0.0010	0.0130	0.001	0.011
0.8	0.227	0.260	0.0000	0.0000	0.000	0.000
0.9	0.111	0.129	0.0000	0.0000	0.000	0.000
1.0	0.000	0.001	0.0000	0.0000	0.000	0.001

Table 6.4.2.: Numerical results at the outlet for different ρ/Γ numbers

6.4.7. Conclusions

It can be seen that the results are quite similar for small ρ/Γ rates. However, the accuracy decreases when this ratio increases because the mesh used in that case is not dense enough on the boundary layers. It can be caused by a wrong convergence criterion, which would return a ϕ field when it has not yet converged.

As the ρ/Γ ratio increases, the convective term takes a more predominant effect. Therefore, the diffusive effect decreases. This behaviour can be observed in the first image where the Peclet number is low, which corresponds to a greater diffusive effect (conductivity is easily observed in the temperature field). But as Peclet number

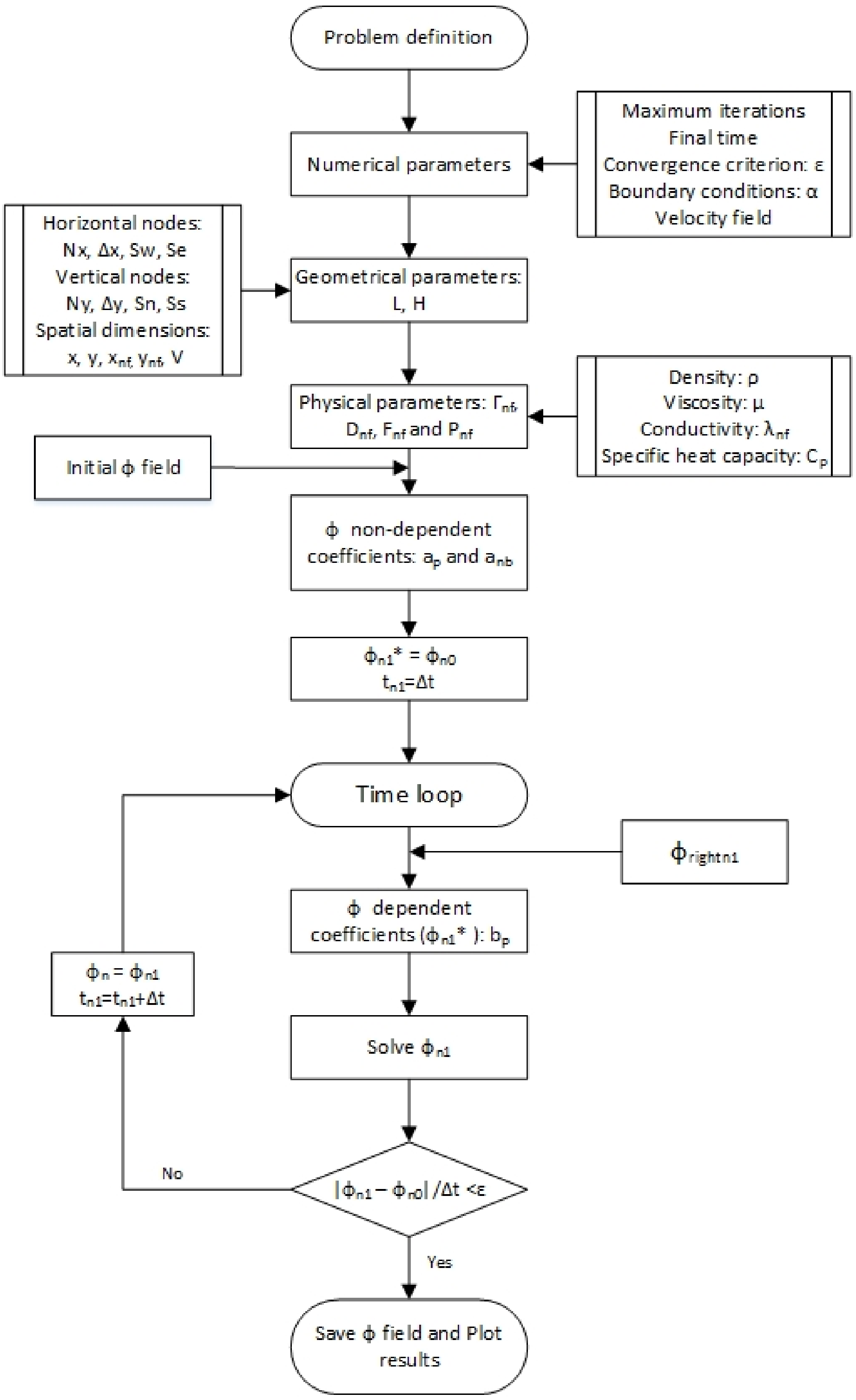

Problem definition
Numerical parameters
Maximum iterations
Final time
Convergence criterion: ε
Boundary conditions: α
Velocity field
Horizontal nodes: Nx, Δx, Sw, Se
Vertical nodes: Ny, Δy, Sn, Ss
Spatial dimensions: x, y, xnf, ynf, V
Geometrical parameters: L, H
Physical parameters: Γnf, Dnf, Fnf and Pnf
Density: ρ
Viscosity: μ
Conductivity: λnf
Specific heat capacity: Cp
Initial ϕ field
ϕ non-dependent coefficients: ap and anb
ϕn1* = ϕn0
tn1=Δt
Time loop
ϕrightn1
ϕ dependent coefficients (ϕn1*): bp
ϕn = ϕn1
tn1=tn1+Δt
Solve ϕn1
|ϕn1 − ϕn0| /Δt <ε
No
Yes
Save ϕ field and Plot results

Figure 6.4.2.: Algorithm used in the specific case: Smith-Hutton

increases, the diffusive term decreases, and only a solenoidal field is seen because the convective term predominates versus the diffusive effect.

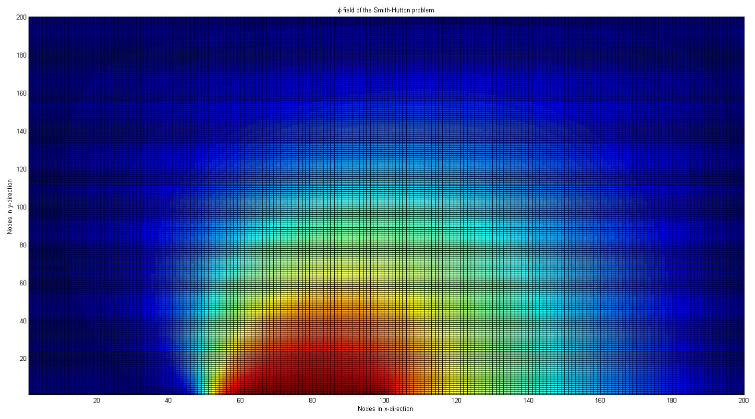


Figure 6.4.3.: ϕ field for Smith-Hutton flow for $\rho/\Gamma = 10$

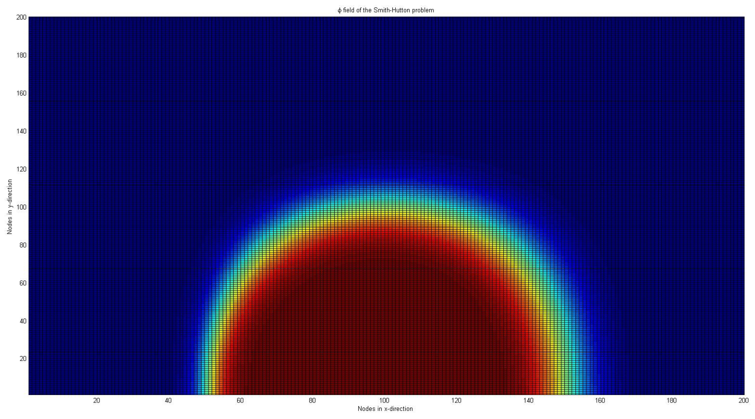


Figure 6.4.4.: ϕ field for Smith-Hutton flow for $\rho/\Gamma = 10^3$

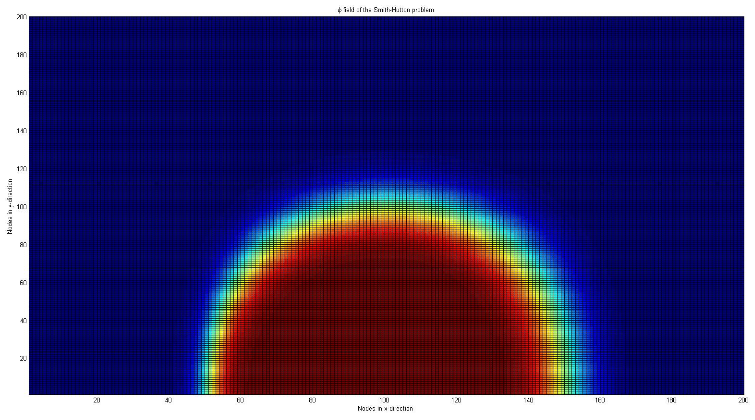


Figure 6.4.5.: ϕ field for Smith-Hutton flow for $\rho/\Gamma = 10^6$

7. Fractional Step Method

The fractional step method [6] solves the incompressible Navier-Stokes equations with a better performance than other methods such as SIMPLE-like algorithms.

This technique projects the system of equations into a divergence-free velocity space. The predictor velocity is an approximate solution of the momentum equation, which is obtained with no pressure gradient contribution. Therefore, this velocity cannot satisfy the incompressibility constraint at the next time level. For that reason, the Poisson equation determines the minimum perturbation that will make the predictor velocity incompressible.

7.1. The governing equations

Quantities t, x, y, u, v, p are non-dimensional forms of their corresponding quantities τ, X, Y, U, V, P and θ, normalized as follows

$$x = \frac{X}{L};\ y = \frac{Y}{L};\ u = \frac{U}{U_\infty};\ v = \frac{V}{U_\infty};\ p = \frac{P}{\rho U_\infty};\ t = \frac{\tau U_\infty}{L}$$

The dimensionless governing equations for incompressible flows of Newtonian fluids are:

$$\frac{\partial \overrightarrow{u}}{\partial t} + (\overrightarrow{u} \cdot \nabla)\overrightarrow{u} = \frac{1}{Re}\triangle \overrightarrow{u} - \nabla p \qquad (7.1.1)$$

$$\nabla \cdot \overrightarrow{u} = 0 \qquad (7.1.2)$$

where $\overrightarrow{u}$ is the velocity vector, p is the pressure and Re is the non-dimensional Reynolds number defined as

$$Re = \frac{\rho V_0 L}{\mu} \qquad (7.1.3)$$

where ρ is the density, V_0 is the characteristic velocity, L is the characteristic length and μ is the dynamic viscosity of the fluid.

The Helmholtz-Hodge decomposition theorem is applied to the incompressible Navier-Stokes equations (Equation 7.1.1 and Equation 7.1.2). The $\Pi(\,\cdot\,)$ is the projector operator that projects any vector field onto a divergence-free space

$$\nabla \cdot \Pi\left(\overrightarrow{a}\right) = 0 \tag{7.1.4}$$

where $\overrightarrow{a}$ is a vector field.

Note that $\Pi\left(\overrightarrow{a}\right)$ will be equal to $\overrightarrow{a}$ if the fluid is incompressible, so $\nabla \cdot \overrightarrow{a} = 0$. On the other hand, $\Pi\left(\nabla\varphi\right)$ will equal 0, where φ is a scalar field.

If this theorem is applied to the Navier-Stokes equations, the projected dimensionless N-S equations yield

$$\Pi\left(\frac{\partial \overrightarrow{u}}{\partial t} + \nabla p\right) = \Pi\left(\frac{1}{Re}\triangle\overrightarrow{u} - \left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u}\right) \tag{7.1.5}$$

The transient term remains unchanged due to the fact that the velocity field is incompressible (Equation 7.1.2).

$$\Pi\left(\frac{\partial \overrightarrow{u}}{\partial t}\right) = \frac{\partial \overrightarrow{u}}{\partial t} \tag{7.1.6}$$

The projection of the pressure gradient vanishes.

$$\Pi\left(\nabla p\right) = 0 \tag{7.1.7}$$

The Navier-Stokes equations can be distributed into two parts, the first one represents a divergence-free vector and the second one is a gradient of an scalar field

$$\begin{aligned} \frac{\partial \overrightarrow{u}}{\partial t} &= \Pi\left(-\left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u} + \frac{1}{Re}\triangle\overrightarrow{u}\right) && (7.1.8)\\ \nabla p &= -\left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u} + \frac{1}{Re}\triangle\overrightarrow{u} - \frac{\partial \overrightarrow{u}}{\partial t} && (7.1.9) \end{aligned}$$

Ultimately, applying the divergence operator to the previous equation and using the project definition shown in Equation 7.1.4 leads to a Poisson equation for pressure

$$\begin{aligned} \nabla\cdot\nabla p &= \nabla\cdot\left(-\left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u} + \frac{\triangle\overrightarrow{u}}{Re}\right) - \nabla\cdot\Pi\left(-\left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u} + \frac{\triangle\overrightarrow{u}}{Re}\right) && (7.1.10)\\ \triangle p &= \nabla\cdot\left(-\left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u} + \frac{\triangle\overrightarrow{u}}{Re}\right) && (7.1.11) \end{aligned}$$

Therefore, the role of the pressure gradient is to project the vector field $\overrightarrow{R}\left(\overrightarrow{u}\right)$ into a divergence-free space. See Figure 7.1.1.

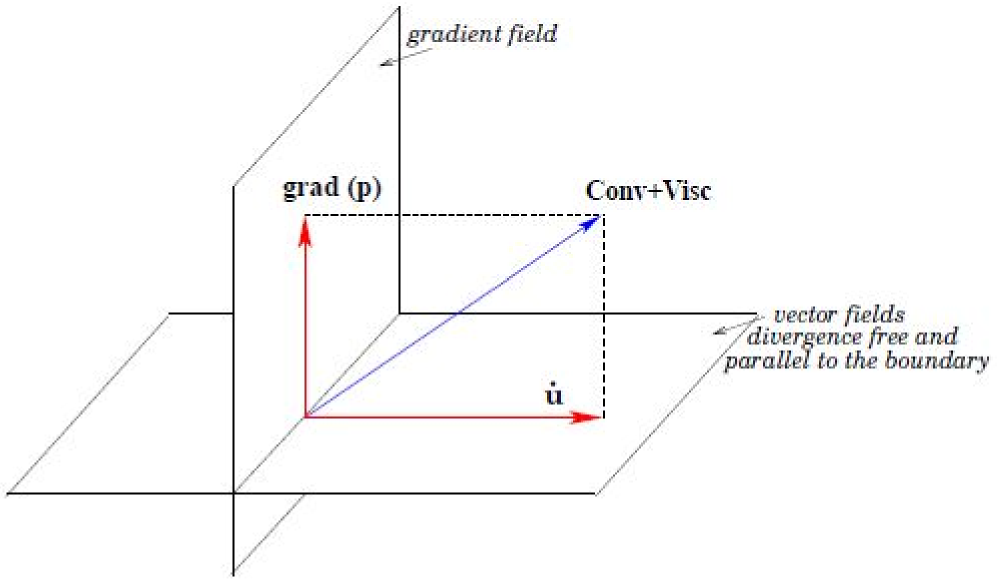


Figure 7.1.1.: Convective and Viscous term vector field of Helmholtz-Hodge decomposition theorem

The momentum equation can be rewritten as

$$\frac{\partial \overrightarrow{u}}{\partial t} = \overrightarrow{R}\left(\overrightarrow{u}\right) - \nabla p \tag{7.1.12}$$

where $\overrightarrow{R}\left(\overrightarrow{u}\right)$ can be defined as

$$\overrightarrow{R}\left(\overrightarrow{u}\right) = -\left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u} + \frac{1}{Re}\triangle\overrightarrow{u} \tag{7.1.13}$$

7.2. Time discretization

Using an explicit time integration scheme, the momentum equation can be discretized as a central difference scheme for the time derivative term, a second-order Adams-Bashforth scheme for $\overrightarrow{R}\left(\overrightarrow{u}\right)$ and a first-order backward Euler scheme for the pressure-gradient term.

$$\left.\frac{\partial \overrightarrow{u}}{\partial t}\right|^{n+1/2} \approx \frac{\overrightarrow{u}^{\,n+1/2} - \overrightarrow{u}^{\,n}}{\Delta t} + \mathcal{O}\left(\Delta t^2\right) \tag{7.2.1}$$

$$\overrightarrow{R}^{\,n+1/2}\left(\overrightarrow{u}\right) \approx \frac{3}{2}\overrightarrow{R}\left(\overrightarrow{u}^{\,n}\right) - \frac{1}{2}\overrightarrow{R}\left(\overrightarrow{u}^{\,n-1}\right) + \mathcal{O}\left(\Delta t^2, \Delta x^m\right) \tag{7.2.2}$$

$$\nabla p^{n+1/2} = \nabla p^{n+1} \tag{7.2.3}$$

Thus, the semi-discretized Navier-Stokes equations can be defined as

$$\frac{\vec{u}^{\,n+1/2}-\vec{u}^{\,n}}{\Delta t}=\frac{3}{2}\vec{R}\left(\vec{u}^{\,n}\right)-\frac{1}{2}\vec{R}\left(\vec{u}^{\,n-1}\right)-\nabla p^{n+1} \tag{7.2.4}$$

The decomposition used by the fractional step method is shown below

$$\vec{u}^{\,p}=\vec{u}^{\,n+1}+\nabla\tilde{p} \tag{7.2.5}$$

where the predictor velocity $\vec{u}^{\,p}$ is given by

$$\vec{u}^{\,p}=\vec{u}^{\,n}+\Delta t\left(\frac{3}{2}\vec{R}\left(\vec{u}^{\,n}\right)-\frac{1}{2}\vec{R}\left(\vec{u}^{\,n-1}\right)\right) \tag{7.2.6}$$

where $\tilde{p}$ is the pseudo-pressure, $\tilde{p}=\Delta t\, p^{n+1}$. Taking the divergence of Equation 7.2.5 yields a Poisson equation for $\tilde{p}$

$$\nabla\cdot\vec{u}^{\,p}=\nabla\cdot\vec{u}^{\,n+1}+\nabla\cdot\left(\nabla\tilde{p}\right)\longrightarrow\Delta\tilde{p}=\nabla\cdot\vec{u}^{\,p} \tag{7.2.7}$$

Then, the velocity at the next time step is obtained from the correction

$$\vec{u}^{\,n+1}=\vec{u}^{\,p}-\nabla\tilde{p} \tag{7.2.8}$$

7.3. Spatial discretization

The spatial discretization of the Equation 7.2.5, Equation 7.2.6, Equation 7.2.7, Equation 7.2.8 and Equation 7.1.13 will be shown below by using the divergence theorem ($\int_S \vec{F}\cdot\vec{n}\,dS=\int_V\nabla\cdot\vec{F}\,dV$):

$$\int\limits_V\frac{\vec{u}^{\,n+1/2}-\vec{u}^{\,n}}{\Delta t}dV=\int\limits_V\frac{3}{2}\vec{R}\left(\vec{u}^{\,n}\right)dV-\int\limits_v\frac{1}{2}\vec{R}\left(\vec{u}^{\,n-1}\right)dV-\int\limits_V\nabla p^{n+1}dV$$

$$\frac{\vec{u}^{\,n+1/2}-\vec{u}^{\,n}}{\Delta t}V_p=\frac{3}{2}\int\limits_V\vec{R}\left(\vec{u}^{\,n}\right)dV-\frac{1}{2}\int\limits_v\vec{R}\left(\vec{u}^{\,n-1}\right)dV-\int\limits_V\nabla p^{n+1}dV \tag{7.3.1}$$

where

$$\int\limits_V\vec{R}\left(\vec{u}^{\,n}\right)dV=\int\limits_V-\left(\vec{u}\cdot\nabla\right)\vec{u}+\frac{1}{Re}\triangle\vec{u}dV=-\int\limits_V\left(\vec{u}\cdot\nabla\right)\vec{u}\,dV+\frac{1}{Re}\int\limits_S\nabla\vec{u}\cdot\vec{n}dS$$

$$\int\limits_V\vec{R}\left(\vec{u}^{\,n}\right)dV=-\int\limits_S\left(\vec{u}\right)\vec{u}\cdot\vec{n}\,dS+\frac{1}{Re}\int\limits_S\nabla\vec{u}\cdot\vec{n}dS \tag{7.3.2}$$

$$\int\limits_V \nabla p^{n+1} dV = p_e S_e - p_w S_w + p_n S_n - p_s S_s \tag{7.3.3}$$

Thus, for each direction

$$\int\limits_V R\left(u^n\right) dV = -\left[(u)_e\, u_e S_e - (u)_w\, u_w S_w + (v)_n\, u_n S_n - (v)_s\, u_s S_s\right] + \\ \frac{1}{Re}\left[\frac{u_E - u_P}{\delta x_e} S_e - \frac{u_P - u_W}{\delta x_w} S_w + \frac{u_N - u_P}{\delta y_n} S_n - \frac{u_P - u_S}{\delta y_s} S_s\right] \tag{7.3.4}$$

$$\int\limits_V R\left(v^n\right) dV = -\left[(u)_e\, v_e S_e - (u)_w\, v_w S_w + (v)_n\, v_n S_n - (v)_s\, v_s S_s\right] + \\ \frac{1}{Re}\left[\frac{v_E - v_P}{\delta x_e} S_e - \frac{v_P - v_W}{\delta x_w} S_w + \frac{v_N - v_P}{\delta y_n} S_n - \frac{v_P - v_S}{\delta y_s} S_s\right] \tag{7.3.5}$$

For x-axis, the velocity at the interface nodes are determined by the next equations

$$\begin{gathered} u_e = Upwind\left(u_P, u_E\right) \\ u_e = CDS\left(u_P, u_E\right) \\ u_e = QUICK\left(u_p, u_E, u_W, u_{EE}, u_{WW}\right) \end{gathered} \tag{7.3.6}$$

$$\begin{gathered} u_n = Upwind\left(u_P, u_N\right) \\ u_n = CDS\left(u_P, u_N\right) \\ u_n = QUICK\left(u_p, u_N, u_S, u_{NN}, u_{SS}\right) \end{gathered} \tag{7.3.7}$$

Equation 7.3.6 can be extrapolated to u_w. Similarly, Equation 7.3.7 can be extrapolated to u_s. For the mass flow values, the velocities are calculated with a CDS scheme.

$$(u)_e = \frac{u_E + u_P}{2} \tag{7.3.8}$$

$$(v)_n = \frac{v_{nA} S_{An} + v_{nB} S_{Bn}}{S_n} \tag{7.3.9}$$

Equation 7.3.8 can be extrapolated to $(u)_w$. Similarly, Equation 7.3.9 can be extrapolated to $(v)_s$. See Figure 7.3.1.

For y-direction, the velocity at the interface nodes are determined by the next equations

$$\begin{gathered} v_e = Upwind\left(v_P, v_E\right) \\ v_e = CDS\left(v_P, v_E\right) \\ v_e = QUICK\left(v_p, v_E, v_W, v_{EE}, v_{WW}\right) \end{gathered} \tag{7.3.10}$$

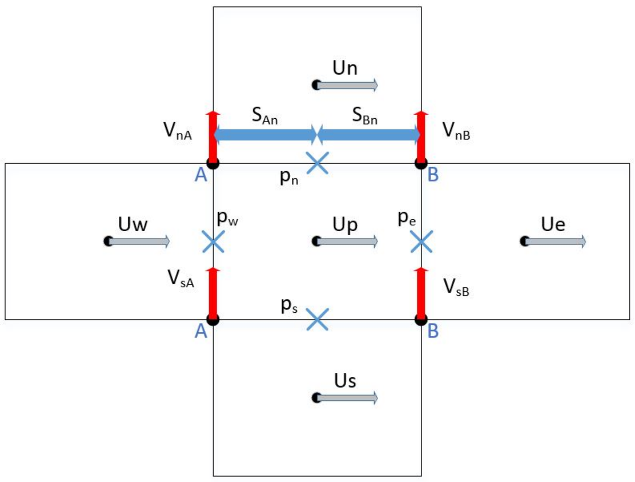


Figure 7.3.1.: Velocity at the x-staggered mesh

$$\begin{gathered} v_n = Upwind\,(v_P, v_N) \\ v_n = CDS\,(v_P, v_N) \\ v_n = QUICK\,(v_p, v_N, v_S, v_{NN}, v_{SS}) \end{gathered} \tag{7.3.11}$$

Equation 7.3.10 can be extrapolated to v_w. Similarly, Equation 7.3.11 can be extrapolated to v_s. For the mass flow values, the velocities are calculated with a CDS scheme.

$$(u)_e = \frac{u_{eA} S_{Ae} + u_{eB} S_{Be}}{S_e} \tag{7.3.12}$$

$$(v)_n = \frac{v_N + v_P}{2} \tag{7.3.13}$$

Equation 7.3.12 can be extrapolated to $(u)_w$. Similarly, Equation 7.3.13 can be extrapolated to $(v)_s$. See Figure 7.3.2.

These equations can be extrapolated to the previous time step.

Equation 7.2.6 can be integrated as

$$\begin{aligned} \int_V \vec{u}^{\,p}\, dV = \int_V \vec{u}^{\,n}\, dV + \Delta t \int_V \left(\frac{3}{2}\vec{R}\left(\vec{u}^{\,n}\right) - \frac{1}{2}\vec{R}\left(\vec{u}^{\,n-1}\right)\right) dV \\ \vec{u}^{\,p} V_p = \vec{u}^{\,n} V_p + \Delta t \int_V \left(\frac{3}{2}\vec{R}\left(\vec{u}^{\,n}\right) - \frac{1}{2}\vec{R}\left(\vec{u}^{\,n-1}\right)\right) dV \end{aligned} \tag{7.3.14}$$

The integration of $\vec{R}\left(\vec{u}^{\,n}\right)$ and $\vec{R}\left(\vec{u}^{\,n-1}\right)$ is shown in Equation 7.3.4 and Equation 7.3.5. The discretized equations applied at staggered grid for each direction are

$$u^p = u^n + \frac{\Delta t}{V_p} \cdot \left[\frac{3}{2}\int_V R\left(u^n\right) dV - \frac{1}{2}\int_V R\left(u^{n-1}\right) dV\right] \tag{7.3.15}$$

$$v^P = v^n + \frac{\Delta t}{V_p} \cdot \left[\frac{3}{2}\int_V R\left(v^n\right) dV - \frac{1}{2}\int_V R\left(v^{n-1}\right) dV\right] \tag{7.3.16}$$

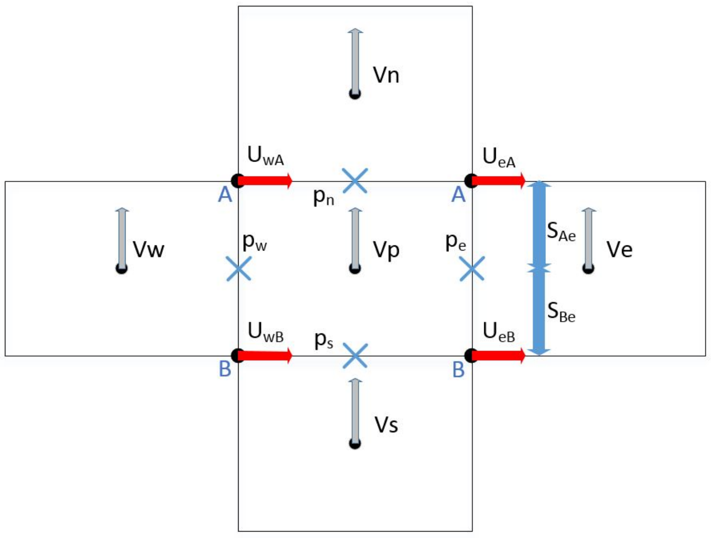


Figure 7.3.2.: Velocity at the y-staggered mesh

The Poisson equation for the central grid can be discretized as

$$\int_V \nabla\cdot\nabla\tilde{p}\,dV = \int_V \nabla\cdot\overrightarrow{u}^{\,p}\,dV \implies \int_S \nabla\tilde{p}\,dS = \int_S \overrightarrow{u}^{\,p}\,dS \implies$$
$$\frac{\tilde{p}_E-\tilde{p}_P}{\delta x_e}S_e-\frac{\tilde{p}_P-\tilde{p}_W}{\delta x_w}S_w+\frac{\tilde{p}_N-\tilde{p}_P}{\delta y_n}S_n-\frac{\tilde{p}_P-\tilde{p}_S}{\delta y_s}S_s=(u_e^pS_e-u_w^pS_w)+(v_n^pS_n-v_s^pS_s) \tag{7.3.17}$$

The linear discretization equation for the previous equation is

$$a_P\tilde{p}_P = a_E\tilde{p}_E + a_W\tilde{p}_W + a_N\tilde{p}_N + a_S\tilde{p}_S + b_P \tag{7.3.18}$$

where

$$a_E = \frac{S_e}{\delta x_e} \tag{7.3.19}$$
$$a_W = \frac{S_w}{\delta x_w} \tag{7.3.20}$$
$$a_N = \frac{S_n}{\delta y_n} \tag{7.3.21}$$
$$a_S = \frac{S_s}{\delta y_s} \tag{7.3.22}$$
$$a_P = a_E + a_W + a_N + a_S \tag{7.3.23}$$
$$b_P = -\left[(u_e^pS_e - u_w^pS_w) + (v_n^pS_n - v_s^pS_s)\right] \tag{7.3.24}$$

Equation 7.2.8 can be integrated for the staggered mesh as

$$\int_v \vec{u}^{\,n+1}\, dV = \int_V \vec{u}^{\,p}\, dV - \int_V \nabla\tilde{p}\, dV \tag{7.3.25}$$

For x direction, the previous equation yields

$$u^{n+1} = u^p - \frac{\tilde{p_e}S_e - \tilde{p_w}S_w}{V_p} \tag{7.3.26}$$

For y direction, the equation yields

$$v^{n+1} = v^p - \frac{\tilde{p_n}S_n - \tilde{p_s}S_s}{V_p} \tag{7.3.27}$$

7.4. The Fractional Step Method Algorithm

The algorithm for the integration of each time step (Figure 7.4.1) is

1. Evaluate $\vec{R}\,(\vec{u}^{\,n})$.
2. Evaluate $\vec{u}^{\,p}$ from Equation 7.2.6.
3. Evaluate $\nabla \cdot \vec{u}^{\,p}$ and solve the discrete Poisson equation Equation 7.2.7.
4. Obtain the new velocity field with Equation 7.2.8.

The explicit temporal scheme introduces severe restrictions on the time step due to stability reasons. The value of the time step is established by the CFL condition

$$\Delta t_{conv}\left(\frac{|u_i|}{\Delta x_i}\right)_{max} \leq C_{conv} \tag{7.4.1}$$

$$\Delta t_{visc}\left(\frac{\nu}{\Delta x_i^2}\right)_{max} \leq C_{visc} \tag{7.4.2}$$

where $C_{convs} = 0.35$ and $C_{visc} = 0.2$.

7.5. The boundary conditions

The boundary conditions for the treatment of temperature are in section 3.6. This is, it may be possible to establish a specific value at the boundary or a heat flow value which will be 0 for adiabatic conditions. Furthermore, the boundary conditions for the treatment of pseudo-pressure are treated in the same manner as heat conduction treatment. It may be possible to establish a specific pressure value or a zero gradient pressure in the boundaries.

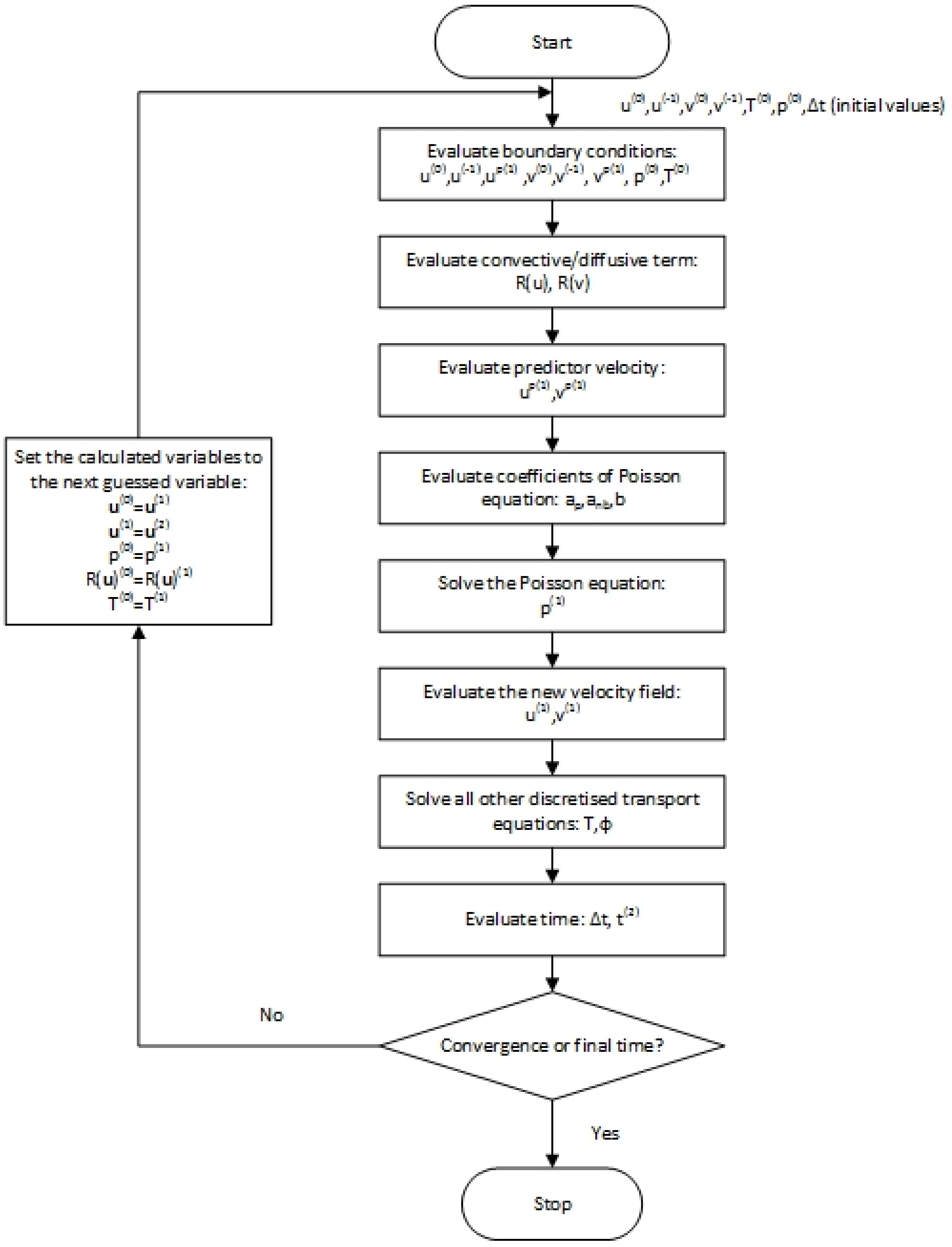

Start
$u^{(0)}, u^{(-1)}, v^{(0)}, v^{(-1)}, T^{(0)}, p^{(0)}, \Delta t$ (initial values)
Evaluate boundary conditions: $u^{(0)}, u^{(-1)}, u^{p(1)}, v^{(0)}, v^{(-1)}, v^{p(1)}, p^{(0)}, T^{(0)}$
Evaluate convective/diffusive term: R(u), R(v)
Evaluate predictor velocity: $u^{p(1)}, v^{p(1)}$
Evaluate coefficients of Poisson equation: a_p, a_{nb}, b
Solve the Poisson equation: $p^{(1)}$
Evaluate the new velocity field: $u^{(1)}, v^{(1)}$
Solve all other discretised transport equations: T, ϕ
Evaluate time: Δt, $t^{(1)}$
Convergence or final time?
No
Yes
Stop
Set the calculated variables to the next guessed variable:
$u^{(0)}=u^{(1)}$
$u^{(1)}=u^{(2)}$
$p^{(0)}=p^{(1)}$
$R(u)^{(0)}=R(u)^{(1)}$
$T^{(0)}=T^{(1)}$

Figure 7.4.1.: The FSM algorithm

However, the boundary conditions for the velocity and predictor velocity have to be established in all the boundary with a specific value, which will be equal for both velocities. It may be possible to establish a specific velocity value for inlets or outputs or a zero value to fulfill the non-slip conditions at the walls. Nevertheless, it may also be imposed a particular value of its gradient.

8. Driven cavity case

8.1. Introduction

The study of a well-known case related to the dimensionless equations for incompressible flows of Newtonian fluids for different Reynolds values using the fractional step method is going to be undertaken. Therefore, it will satisfy mass conservation equation and momentum conservation equation. The unknown variables are the velocity vector at the interface nodes and the pressure at each node. The case chosen to be solved and to prove the previous equations is a steady case, which is called lid-driven cavity flow [14]. See Figure 8.1.1.

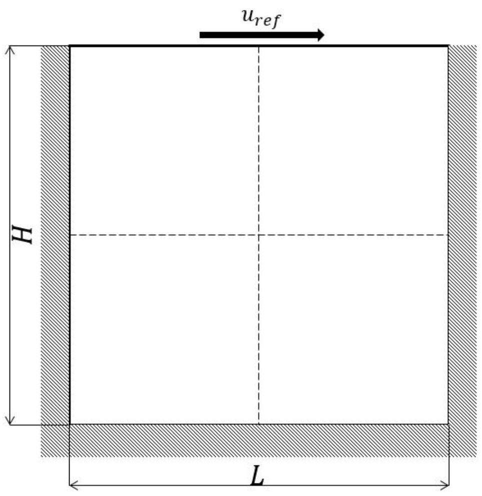


Figure 8.1.1.: Lid-driven cavity flow problem

8.2. Boundary conditions

The boundary conditions for velocity are defined below

- Initial u *and* v value: $u = 0.0$ *and* $v = 0.0$.
- Left boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Right boundary condition: $u = 0.0$ *and* $v = 0.0$.

- Bottom boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Top boundary condition: $u = U_0 = 1.0$ *and* $v = 0.0$.

The boundary conditions for predictor velocity are defined below

- Initial u^p *and* v^p value: $u^p = 0.0$ *and* $v^p = 0.0$.
- Left boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Right boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Bottom boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Top boundary condition: $u^p = U_0 = 1.0$ *and* $v^p = 0.0$.

The boundary conditions for pseudo-pressure are defined below

- Initial $\tilde{p}$ value: $\tilde{p} = 1.2$.
- Left boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial x = 0$).
- Right boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial x = 0$).
- Bottom boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial y = 0$).
- Top boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial y = 0$).

8.3. Spatial discretization

The spatial discretization is shown in Table 8.3.1.

Spatial property	L	H	N_x	N_y	Δx	Δy
Value	1	1	129	129	7.75e-3	7.75e-3

Table 8.3.1.: Spatial discretization for driven cavity flow

8.4. The governing equations

The program uses the same equations shown in chapter 7. For that reason, no more information is needed.

Note that an explicit method is used to determine the velocity field and the pressure field, so the time increment must be exhaustively computed in order to achieved a converged solution which will be a consistent result. On the other hand, a CDS scheme is used to determine the velocity on the interface nodes.

A TDMA-GS solver has been implemented due to the fact that this type of problem implies a high computational cost to solve the poisson equation. Then, it is needed a solver faster than a GS solver. Nevertheless, a conjugate gradient solver could be used.

The problem will be solved for the next Reynolds numbers: 100, 400, 1000, 3200, 5000, 7500 and 10000. The Reynolds number can be computed using the previous formula shown in Equation 7.1.3, so the characteristic velocity is equal to 1, the characteristic length is equal to 1, the dynamic viscosity is equal to 1 and the density will be equal to the Reynolds number.

8.5. The program

The program creates a class with the matrix of the coefficients to solve the Poisson equation. It also creates a class for the u component of the interface nodes for the staggered mesh in x axis, a class for the v component of the interface nodes for the staggered mesh in y axis, a class for the mass flow of the interface nodes and a class to calculate the velocity gradient of the interface nodes.

Finally, the program creates a class with all the information related to each grid node. Therefore, it contains the u and v values for the three time steps, the u^p and v^ppredictor velocity, the coordinate values of the grid node and interface nodes, the physical properties, the $\overrightarrow{R}(\overrightarrow{u})$values for the two previous time steps, the velocity values of the interface nodes, the mass flow values of the interface nodes, the velocity gradients values of the interface nodes and the coefficients of Equation 7.3.18.

The inputs required by the program are: the point coordinates, the number of nodes in each direction, the maximum number of iterations allowed, the convergence criterion for velocity and pressure equations, the Reynolds number, the viscosity value, the density value, and, finally, the condition for the four boundaries, which can be either wall or airflow. If the boundary condition is airflow, it will need the values for u and v components. Another requirement needed by the program is the last time to solve the time loop.

The program consists in a file with different functions which computes parts of the equations. The functions are explained below

- Main function: it contains the general algorithm that calls the different sub-functions to perform its function inside the algorithm. See Figure 8.5.1.
- Set Mesh function: it calculates the geometrical properties of each grid node and its interfaces using a uniform, orthogonal mesh.
- Initial Values function: it establishes the initial values of velocity and pressure.

- Evaluate boundary velocity function: it converts the inputs of the boundary conditions to velocity and predictor velocity.
- CDS scheme function: it returns the velocity value of the interface nodes with a specific indexes.
- Interface velocity function: it determines the velocity value of the interface nodes.
- Mass flow function: it calculates the value for the mass flow at each interface.
- Gradient velocity function: it determines the value for the velocity gradient of the interface nodes.
- Evaluate $\overrightarrow{R}(\overrightarrow{u})$ function: it calculates the value for Equation 7.3.4 and Equation 7.3.5.
- Evaluate u^p velocity function: it returns the value for the predictor velocity of each node of the staggered mesh.
- Coefficients function: it calculates the coefficients taking into account the different conditions of each node. In this case there are also 9 different situations. It computes the constant coefficients and the variable coefficients independently.
- TDMA Solver function: this code solves the equation Equation 7.3.18 using a TDMA-GS line-by-line algorithm.
- Evaluate current velocity function: it calculates the current velocity field using Equation 7.3.26 and Equation 7.3.27.
- Evaluate time step function: this function determines the time step for the next iteration using the parameters shown in Equation 7.4.1 and Equation 7.4.2.
- Evaluate convergence function: it determines the maximum error made by the derivative of the velocity field respect the previous instant.
- New step function: this function saves the current variables to the previous time step variables for the next iteration.
- Save values function: it exports a file with the velocity field and the pressure field.

8.6. The results

After the code was created, it was validated with u along y coordinate points for $x = 0.5$ to make sure that everything was correct and v along x coordinate points for $y = 0.5$. The results for the u field and v field when it converged are shown below for

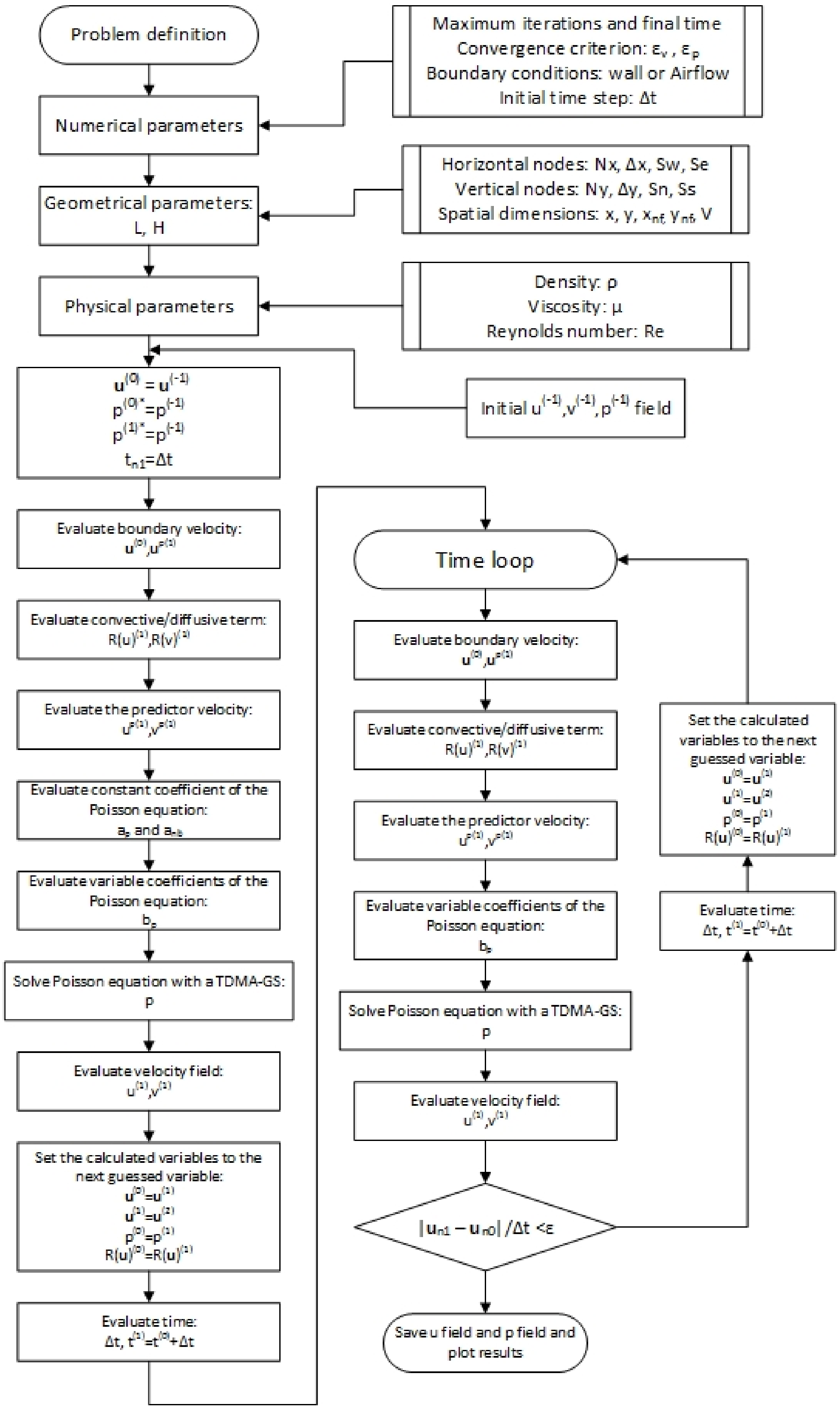

Problem definition
Numerical parameters
Maximum iterations and final time
Convergence criterion: ε_v, ε_p
Boundary conditions: wall or Airflow
Initial time step: Δt
Geometrical parameters: L, H
Horizontal nodes: Nx, Δx, Sw, Se
Vertical nodes: Ny, Δy, Sn, Ss
Spatial dimensions: x, y, x_{nf}, y_{nf}, V
Physical parameters
Density: ρ
Viscosity: μ
Reynolds number: Re
$u^{(0)} = u^{(-1)}$
$p^{(0)*}=p^{(-1)}$
$p^{(1)*}=p^{(-1)}$
$t_{n1}=\Delta t$
Initial $u^{(-1)}$,$v^{(-1)}$,$p^{(-1)}$ field
Evaluate boundary velocity: $u^{(0)}$,$u^{p(1)}$
Evaluate convective/diffusive term: $R(u)^{(1)}$,$R(v)^{(1)}$
Evaluate the predictor velocity: $u^{p(1)}$,$v^{p(1)}$
Evaluate constant coefficient of the Poisson equation: a_e and a_{nb}
Evaluate variable coefficients of the Poisson equation: b_p
Solve Poisson equation with a TDMA-GS: p
Evaluate velocity field: $u^{(1)}$,$v^{(1)}$
Set the calculated variables to the next guessed variable: $u^{(0)}=u^{(1)}$ $u^{(1)}=u^{(2)}$ $p^{(0)}=p^{(1)}$ $R(u)^{(0)}=R(u)^{(1)}$
Evaluate time: Δt, $t^{(1)}=t^{(0)}+\Delta t$
Time loop
Evaluate boundary velocity: $u^{(0)}$,$u^{p(1)}$
Evaluate convective/diffusive term: $R(u)^{(1)}$,$R(v)^{(1)}$
Evaluate the predictor velocity: $u^{p(1)}$,$v^{p(1)}$
Evaluate variable coefficients of the Poisson equation: b_p
Solve Poisson equation with a TDMA-GS: p
Evaluate velocity field: $u^{(1)}$,$v^{(1)}$
$|u_{n1} - u_{n0}| / \Delta t < \varepsilon$
Save u field and p field and plot results
Set the calculated variables to the next guessed variable: $u^{(0)}=u^{(1)}$ $u^{(1)}=u^{(2)}$ $p^{(0)}=p^{(1)}$ $R(u)^{(0)}=R(u)^{(1)}$
Evaluate time: Δt, $t^{(1)}=t^{(0)}+\Delta t$

Figure 8.5.1.: The driven cavity algorithm

different Reynolds numbers (See Annex A). Finally, the calculated solution can be compared with the benchmarking solution (See Table 8.6.1, Table 8.6.2, Table 8.6.3 and Table 8.6.4).

It is going to compare the values graphically to observe the error between the values obtained by the own code and the values given by Ghia [14]. See Figure 8.6.1.

8.7. Conclusions

It is easy to see that the relative error becomes bigger and bigger for high Reynolds numbers even though the mesh grid has been resized to obtain more precision. The reason for such error is the turbulence of the flow for a high Reynolds number, that makes the velocity values fluctuate over time. The values returned by the code are specific values at the time step of convergence instead of statistical values as those calculated by Ghia.

If the values given by Ghia [14] are studied in detail, it can be immediately observed that the velocity gradient of the bottom surface increases with the Reynolds number. For that reason, it is necessary to densify the grid when the Reynolds number increases. Increasing the density near the surfaces would lead to an optimal solution due to the fact that there is no need to impose a high density in the centre of the element. However, the computational cost of such densification would not counteract the increase of accuracy.

Note that the flow for solved cases with $Re = 7500$ and $Re = 10000$ is turbulent, or at least it should be. Nevertheless, it is not clear if the flow for a $Re = 7500$ is still stationary or is already transitory. However, this case with $Re = 7500$ has been computed, thus obtaining accurate results, despite they could depend on the convergence criterion that has been used due to the fact that it is a slow transitory case. On the other hand, the case with $Re = 10000$ has been unable to be solved due to its fluctuations over time. The values shown in result tables for this Reynolds number are only approximations at a specific moment due to the fact that it has not converged (the convergence error was about 10^{-2}). This phenomenon can be observed in the graphics of errors due to the fact that the accuracy of the results decreases when Reynolds number increases.

Another problem or difference between the code results and the Ghia results may be the different scheme used due to the fact that Ghia used an upwind scheme which may lead to spurious results due to the anisotropy of the artificial viscosity [3].

The flow shows the logical movement, since the top border causes a horizontal displacement of the fluid (red colour range in the graph of the horizontal velocity) until

	Re=100		Re=400		Re=1000		Re=3200		Re=5000		Re=7500		Re=10000	
y	bench	code	bench	code	bench	code	bench	code	bench	code	bench	code	bench	code
1,0000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000	1,00000
0,9766	0,84123	0,84371	0,75837	0,75992	0,65928	0,66164	0,53236	0,51555	0,48223	0,47317	0,47244	0,44797	0,47221	0,43444
0,9688	0,78871	0,79187	0,68439	0,68634	0,57492	0,57762	0,48296	0,46479	0,46120	0,44798	0,47048	0,44207	0,47783	0,43575
0,9609	0,73722	0,74048	0,61756	0,61896	0,51117	0,51322	0,46547	0,44624	0,45992	0,44511	0,47323	0,44472	0,48070	0,43749
0,9531	0,68717	0,69098	0,55892	0,56054	0,46604	0,46807	0,46101	0,44143	0,46036	0,44544	0,47167	0,44402	0,47804	0,43402
0,8516	0,23151	0,23684	0,29093	0,29026	0,33304	0,33250	0,34682	0,33449	0,33556	0,32525	0,34228	0,31614	0,34635	0,30040
0,7344	0,00332	0,00373	0,16256	0,16159	0,18719	0,18618	0,19791	0,19340	0,20087	0,18915	0,20591	0,18495	0,20673	0,17384
0,6172	-0,13641	-0,13841	0,02135	0,02071	0,05702	0,05600	0,07156	0,07357	0,08183	0,07419	0,08342	0,07397	0,08344	0,06940
0,5000	-0,20581	-0,21077	-0,11477	-0,11503	-0,06080	-0,06166	-0,04272	-0,03602	-0,03039	-0,03032	-0,03800	-0,02699	-0,03111	-0,02380
0,4531	-0,21090	-0,21537	-0,17119	-0,17132	-0,10648	-0,10725	-0,86636	-0,07859	-0,07404	-0,07095	-0,07503	-0,06622	-0,07540	-0,06003
0,2813	-0,15662	-0,15799	-0,32726	-0,32655	-0,27805	-0,27794	-0,24427	-0,23353	-0,22855	-0,21991	-0,23176	-0,20982	-0,23186	-0,19562
0,1719	-0,10150	-0,10167	-0,24299	-0,24194	-0,38289	-0,38239	-0,34323	-0,33286	-0,33050	-0,31622	-0,32393	-0,30257	-0,32709	-0,28546
0,1016	-0,06434	-0,06412	-0,14612	-0,14535	-0,29730	-0,29484	-0,41933	-0,41374	-0,40435	-0,39276	-0,38324	-0,36596	-0,38000	-0,34289
0,0703	-0,04775	-0,04640	-0,10338	-0,10279	-0,22220	-0,21918	-0,37827	-0,38536	-0,43643	-0,41197	-0,43025	-0,40623	-0,41657	-0,38091
0,0625	-0,04192	-0,04186	-0,09266	-0,09216	-0,20196	-0,19907	-0,35344	-0,36363	-0,42901	-0,40111	-0,43590	-0,40779	-0,42537	-0,38949
0,0547	-0,03717	-0,03705	-0,08186	-0,08145	-0,18109	-0,17846	-0,32407	-0,33682	-0,41165	-0,38169	-0,43154	-0,40000	-0,42735	-0,39241
0,0000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000

Table 8.6.1.: Comparative between benchmarking values [14] and code values for u velocity component

	Re=100		Re=400		Re=1000		Re=3200		Re=5000		Re=7500		Re=10000	
x	bench	code	bench	code	bench	code	bench	code	bench	code	bench	code	bench	code
1,0000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000
0,9688	-0,05906	-0,06220	-0,12146	-0,12441	-0,21388	-0,22364	-0,39017	-0,41277	-0,49774	-0,47262	-0,53858	-0,49804	-0,54302	-0,49317
0,9609	-0,07391	-0,07790	-0,15663	-0,16070	-0,27669	-0,28865	-0,47425	-0,49045	-0,55069	-0,52382	-0,55216	-0,51648	-0,52987	-0,49182
0,9531	-0,08864	-0,09337	-0,19254	-0,19718	-0,33714	-0,34960	-0,52357	-0,53071	-0,55408	-0,53032	-0,52347	-0,49813	-0,49099	-0,46443
0,9453	-0,10313	-0,10840	-0,22847	-0,23356	-0,39188	-0,40396	-0,54053	-0,53962	-0,52876	-0,51131	-0,48590	-0,46842	-0,45863	-0,43544
0,9063	-0,16914	-0,17691	-0,23827	-0,38774	-0,51550	-0,51813	-0,44307	-0,43137	-0,41442	-0,40547	-0,41050	-0,39127	-0,41496	-0,37549
0,8594	-0,22445	-0,23507	-0,44993	-0,45219	-0,42665	-0,42230	-0,37401	-0,36435	-0,36214	-0,35320	-0,36213	-0,34041	-0,36737	-0,32433
0,8047	-0,24533	-0,25201	-0,38598	-0,38367	-0,31966	-0,31709	-0,31184	-0,30209	-0,30018	-0,29058	-0,30448	-0,28051	-0,30719	-0,26581
0,5000	0,05454	0,05382	0,05186	0,05234	0,02526	0,02582	0,00999	0,01378	0,00945	0,01063	0,00824	0,00819	0,00831	0,00639
0,2344	0,17527	0,17805	0,30174	0,30148	0,32235	0,32190	0,28188	0,27750	0,27280	0,26265	0,27348	0,25026	0,27224	0,23453
0,2266	0,17507	0,17996	0,30203	0,30181	0,33075	0,33031	0,29030	0,28557	0,28066	0,27050	0,28117	0,25780	0,28003	0,24190
0,1563	0,16077	0,16422	0,28124	0,28149	0,37095	0,37105	0,37119	0,36392	0,35368	0,34329	0,35060	0,32746	0,35070	0,31076
0,0938	0,12317	0,12649	0,22965	0,22992	0,32627	0,32722	0,42768	0,41306	0,42951	0,41326	0,41824	0,39780	0,41487	0,37992
0,0781	0,10890	0,11194	0,20920	0,20926	0,30353	0,30441	0,41906	0,40182	0,43648	0,41664	0,43564	0,41251	0,43124	0,39868
0,0703	0,10091	0,10413	0,19713	0,19716	0,29012	0,29094	0,40917	0,39079	0,43329	0,41193	0,44030	0,41506	0,43733	0,40524
0,0625	0,09233	0,09505	0,18360	0,18358	0,27485	0,27556	0,39560	0,37622	0,42447	0,40212	0,43979	0,41242	0,43983	0,40696
0,0000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000	0,00000

Table 8.6.2.: Comparative between benchmarking [14] values and code values for v velocity component

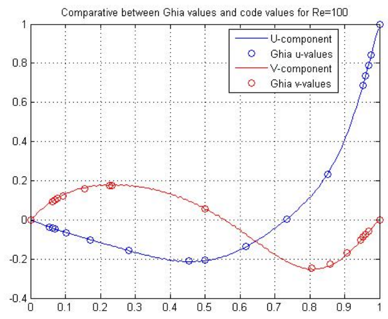
y	Re=100	Re=400	Re=1000	Re=3200	Re=5000	Re=7500	Re=10000
1,0000	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%
0,9766	0,294%	0,205%	0,358%	3,158%	1,879%	5,179%	8,000%
0,9688	0,401%	0,285%	0,470%	3,762%	2,866%	6,038%	8,807%
0,9609	0,442%	0,227%	0,401%	4,132%	3,221%	6,025%	8,989%
0,9531	0,555%	0,290%	0,436%	4,248%	3,240%	5,863%	9,209%
0,8516	2,303%	0,230%	0,163%	3,556%	3,072%	7,638%	13,268%
0,7344	12,481%	0,595%	0,541%	2,280%	5,833%	10,181%	15,908%
0,6172	-1,465%	3,010%	1,785%	2,803%	9,333%	11,329%	16,824%
0,5000	-2,411%	-0,228%	-1,418%	-15,686%	-0,215%	-28,977%	-23,509%
0,4531	-2,121%	-0,076%	-0,719%	-90,929%	-4,169%	-11,738%	-20,382%
0,2813	-0,877%	-0,217%	-0,040%	-4,398%	-3,781%	-9,466%	-15,632%
0,1719	-0,172%	-0,433%	-0,132%	-3,022%	-4,320%	-6,593%	-12,727%
0,1016	-0,346%	-0,529%	-0,829%	-1,332%	-2,866%	-4,508%	-9,765%
0,0703	-2,820%	-0,570%	-1,358%	-1,874%	-5,605%	-5,583%	-8,560%
0,0625	-0,134%	-0,541%	-1,429%	-2,884%	-6,504%	-6,448%	-8,435%
0,0547	-0,311%	-0,505%	-1,452%	-3,934%	-7,278%	-7,308%	-8,175%
0,0000	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%

Table 8.6.3.: Relative error between benchmarking values [14] and code values for u velocity component

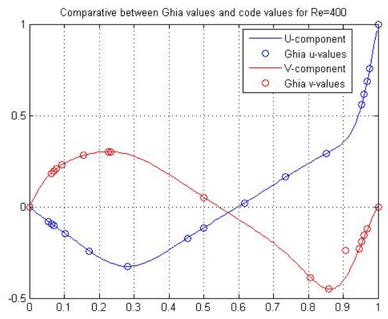
x	Re=100	Re=400	Re=1000	Re=3200	Re=5000	Re=7500	Re=10000
1,0000	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%
0,9688	-5,311%	-2,433%	-4,563%	-5,793%	-5,048%	-7,527%	-9,180%
0,9609	-5,403%	-2,598%	-4,323%	-3,416%	-4,879%	-6,462%	-7,180%
0,9531	-5,337%	-2,411%	-3,696%	-1,364%	-4,289%	-4,842%	-5,409%
0,9453	-5,109%	-2,229%	-3,083%	-0,169%	-3,300%	-3,596%	-5,055%
0,9063	-4,596%	-62,733%	-0,509%	-2,641%	-2,159%	-4,685%	-9,512%
0,8594	-4,729%	-0,502%	-1,019%	-2,584%	-2,470%	-5,997%	-11,715%
0,8047	-2,724%	-0,598%	-0,805%	-3,128%	-3,197%	-7,873%	-13,471%
0,5000	1,317%	0,924%	2,232%	37,961%	12,460%	0,570%	23,065%
0,2344	1,584%	0,087%	0,138%	1,555%	3,720%	8,490%	13,851%
0,2266	2,795%	0,071%	0,134%	1,628%	3,620%	8,311%	13,617%
0,1563	2,144%	0,087%	0,026%	1,959%	2,937%	6,599%	11,387%
0,0938	2,696%	0,116%	0,292%	3,419%	3,784%	4,887%	8,425%
0,0781	2,789%	0,031%	0,290%	4,113%	4,544%	5,309%	7,551%
0,0703	3,191%	0,014%	0,283%	4,492%	4,930%	5,732%	7,339%
0,0625	2,946%	0,008%	0,258%	4,899%	5,265%	6,224%	7,472%
0,0000	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%	0,000%

Table 8.6.4.: Relative error between benchmarking values [14] and code values for u velocity component

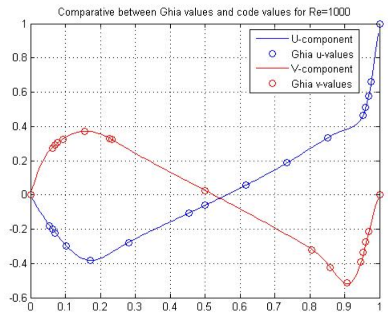
it collides with the right wall, thus making the flow go downwards quickly (blue colour area in the graph of the vertical velocity). Due to the high viscosity of the fluid, the energy is damped quickly making it turn aside towards the left side. Afterwards,



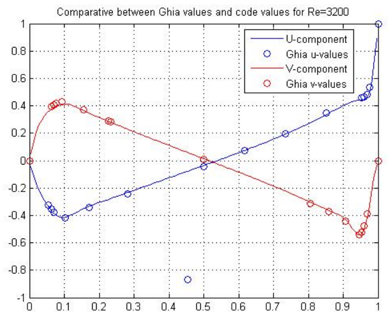
(a) Re=100



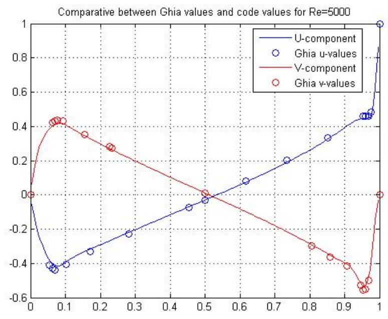
(b) Re=400



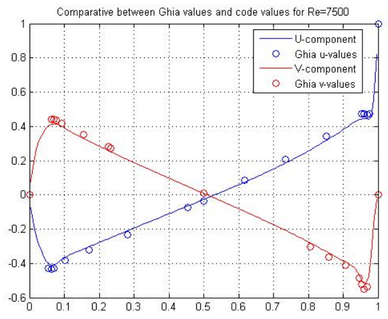
(c) Re=1000



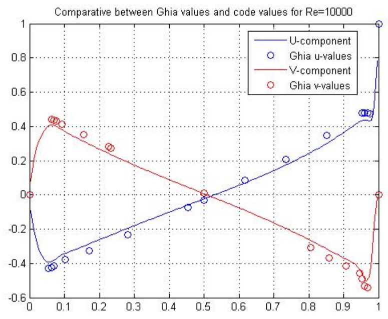
(d) Re=3200



(e) Re=5000

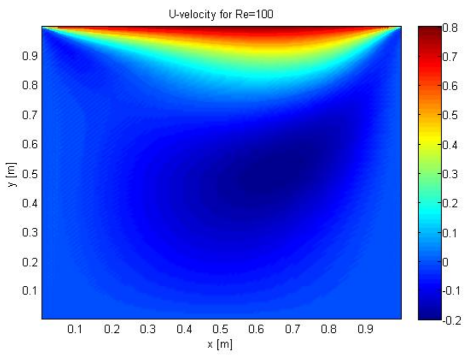


(f) Re=7500

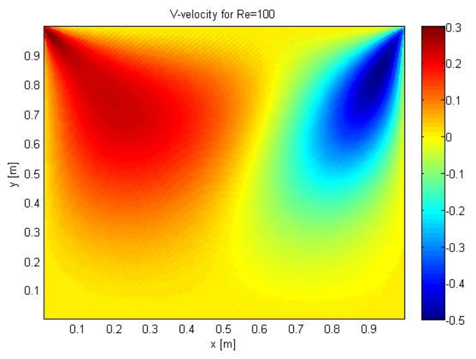


(g) Re=10000

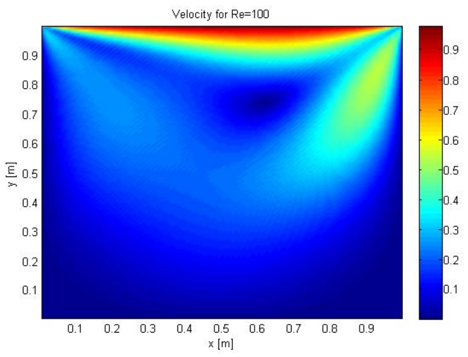
Figure 8.6.1.: Graphical error between benchmarking values and code values



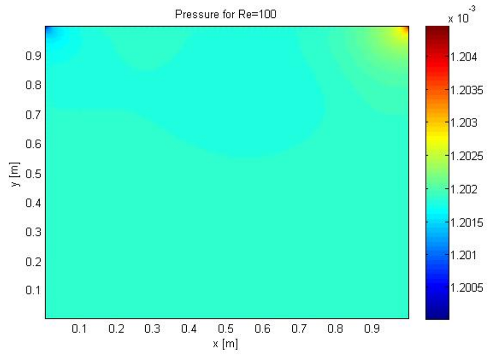
(a) U field



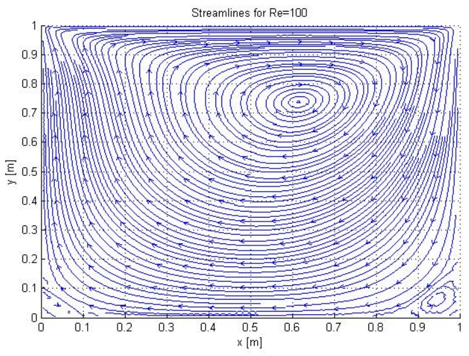
(b) V field



(c) Velocity field



(d) Pressure field



(e) Streamlines

Figure 8.6.2.: Results for Re=100

it ascends.

From the plotted graphs of the variables u, v, p and ψ, it is possible to observe that the circulation of the flow increases (the values and areas of the horizontal and vertical velocities are bigger) with the Reynolds number, even becoming turbulent for high Reynolds (Reynolds number higher than 7500). This phenomenon is also reflected in the movement of the centroid of the circulation towards the centre of the domain. In turn, the creation of recirculations is observed in the top left corner, the bottom left corner and the bottom right corner.

9. Differentially heated cavity case

9.1. Introduction

The study of a well-known case related to the dimensionless equations for incompressible flows of Newtonian fluids for different Rayleigh values and a Prandtl number of 0.71 using the fractional step method is about to be undertaken. Therefore, it must satisfy mass conservation equation with constant properties, momentum conservation equation with Boussinesq approximation for density and energy conservation equation. The unknown variables are the velocity vector at the interface nodes, the pressure and the temperature at each node.

The case chosen to be solved and to prove the previous equations is a steady case which, is called differentially heated cavity flow [11, 12]. See Figure 9.1.1. It is a natural convection problem that uses the Boussinesq approximation[1] for density, which changes linearly with the temperature.

$$\rho = \rho_0 \left(1 - \beta_0 \left(T - T_0\right)\right) \tag{9.1.1}$$

9.2. Boundary conditions

The boundary conditions for velocity are defined below

- Initial u *and* v value: $u = 0.0$ *and* $v = 0.0$.
- Left boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Right boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Bottom boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Top boundary condition: $u = 0.0$ *and* $v = 0.0$.

The boundary conditions for predictor velocity are defined below

[1]The density is treated as constant in the unsteady and convection terms, and treated as variable only in the gravitational term.

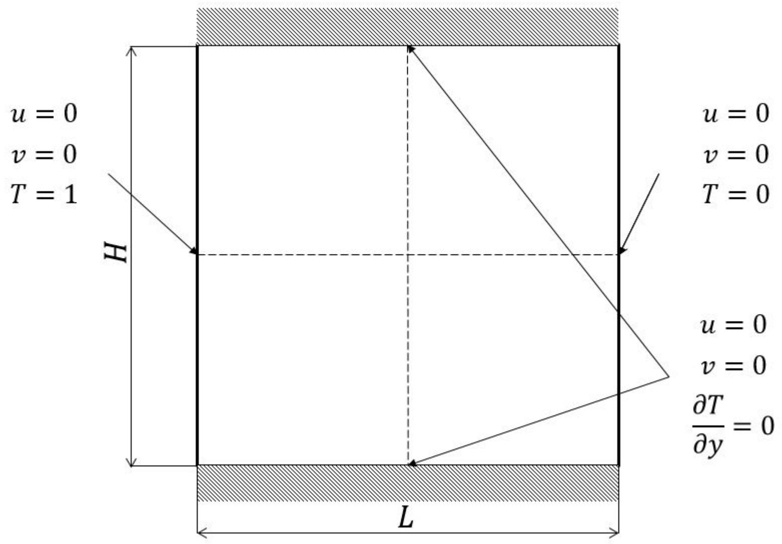


Figure 9.1.1.: Differentially heated cavity flow problem

- Initial u^p *and* v^p value: $u^p = 0.0$ *and* $v^p = 0.0$.
- Left boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Right boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Bottom boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Top boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.

The boundary conditions for pseudo-pressure are defined below

- Initial $\tilde{p}$ value: $\tilde{p} = 1.2$.
- Left boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial x = 0$).
- Right boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial x = 0$).
- Bottom boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial y = 0$).
- Top boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial y = 0$).

The boundary conditions for temperature are defined below

- Initial T value: $T = 0.5$.
- Left boundary condition: $T = 1.0$.
- Right boundary condition: $T = 0.0$.
- Bottom boundary condition: zero normal gradient temperature ($\partial T/\partial y = 0$).
- Top boundary condition: zero normal gradient temperature ($\partial T/\partial y = 0$).

9.3. Spatial discretization

The spatial discretization is shown in Table 9.3.1.

Spatial property	L	H	N_x	N_y	Δx	Δy
Value	1	1	201	201	4.975E-3	4.975E-3

Table 9.3.1.: Spatial discretization for differentially heated cavity flow

9.4. The governing equations

The problem will be solved for the next Rayleigh numbers: 10^3 , 10^4 , 10^5 and 10^6 and a 0.71 Prandtl number [11, 12]. The Rayleigh number can be computed using the next formula:

$$Ra = \frac{\beta g\,\Delta T\,L^3}{\nu\alpha} \tag{9.4.1}$$

where β is the thermal expansion coefficient, g is the gravitational acceleration, ΔT is the temperature difference $(T_H - T_C)$, L is the cavity length, ν is the kinematic viscosity and α is the thermal diffusivity which can be defined as $\alpha = \frac{\lambda}{\rho C_p}$, where λ is the thermal conductivity, ρ is the density and C_p is the specific thermal heat.

The Prandtl number can be computed using the next formula:

$$Pr = \frac{\nu}{\alpha} \tag{9.4.2}$$

where ν is the kinematic viscosity and α is the thermal diffusivity. This value will be 0.71 due to the fact that the fluid inside the cavity is air.

The program uses the same equations shown in chapter 7, however the non-dimensional variables are different. In this case, the dimensionless variables are the Rayleigh number and the Prandtl number. For that reason, the differences between the equations shown in chapter 7 are going to be explained.

The dimensionless governing equations for incompressible flows of Newtonian fluids are:

$$\begin{aligned}
\frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\,\vec{u} &= \frac{Pr}{Ra^{1/2}}\triangle\vec{u} - \nabla p + \vec{f} && (9.4.3)\\
\nabla\cdot\vec{u} &= 0 && (9.4.4)\\
\frac{\partial T}{\partial t} + (\vec{u}\cdot\nabla)\,T &= \frac{1}{Ra^{1/2}}\nabla T + S && (9.4.5)
\end{aligned}$$

where $\vec{u}$ is the velocity vector, p is the pressure, Pr is the non-dimensional Prandtl number, Ra is the non-dimensional Rayleigh number, $\vec{f}$ is the body force vector which is equal to $(0, Pr\,T)$, T is the non-dimensional temperature and S is the non-dimensional internal heat generation.

Quantities t, x, y, u, v, p and T are non-dimensional forms of their corresponding quantities τ, X, Y, U, V, P and θ, normalized as follows [29, 34]

$$x = \frac{X}{L};\; y = \frac{Y}{L};\; u = \frac{UL}{\kappa Ra^{1/2}};\; v = \frac{VL}{\kappa Ra^{1/2}};\; T = \frac{\theta - \theta_C}{\theta_H - \theta_C};\; p = \frac{PL^2}{\rho\kappa^2 Ra};\; t = \frac{\tau\kappa Ra^{1/2}}{L^2}$$

Note that the dimensionless forms used by G. de Vahl Davis are different for u and v. He used a $u = \frac{UL}{\kappa}$ instead of $u = \frac{UL}{\kappa Ra^{1/2}}$. For that reason, the final results must be modified to obtain the same results.

The Helmholtz-Hodge decomposition theorem is applied to the incompressible Navier-Stokes equations (Equation 9.4.3 and Equation 9.4.4) as explained before. In that case, the Navier-Stokes equations can be distributed into two parts: the first one represents a divergence-free vector and the second one is a gradient of a scalar field

$$\frac{\partial \vec{u}}{\partial t} = \Pi\left(-\left(\vec{u}\cdot\nabla\right)\vec{u} + \frac{Pr}{Ra^{1/2}}\triangle\vec{u} + \vec{f}\right) \tag{9.4.6}$$

$$\nabla p = -\left(\vec{u}\cdot\nabla\right)\vec{u} + \frac{Pr}{Ra^{1/2}}\triangle\vec{u} + \vec{f} - \frac{\partial \vec{u}}{\partial t} \tag{9.4.7}$$

Finally, applying the divergence operator to the previous equation and using the project definition a Poisson equation for pressure is obtained

$$\triangle p = \nabla\cdot\left(-\left(\vec{u}\cdot\nabla\right)\vec{u} + \frac{Pr}{Ra^{1/2}}\triangle\vec{u} + \vec{f}\right) \tag{9.4.8}$$

Therefore, the role of the pressure gradient is to project the vector field $\vec{R}\left(\vec{u}\right)$ into a divergence-free space.

The momentum equation can be rewritten as

$$\frac{\partial \vec{u}}{\partial t} = \vec{R}\left(\vec{u}\right) - \nabla p \tag{9.4.9}$$

where $\vec{R}\left(\vec{u}\right)$ can be defined as

$$\vec{R}\left(\vec{u}\right) = -\left(\vec{u}\cdot\nabla\right)\vec{u} + \frac{Pr}{Ra^{1/2}}\triangle\vec{u} + \vec{f} \tag{9.4.10}$$

The time discretzation is done in the same manner as section 7.2. However, the spatial discretization needs some indications. The first one, Equation 7.3.2 needs an extra explanation

$$\int\limits_V \overrightarrow{R}\left(\overrightarrow{u}^{\,n}\right) dV = -\int\limits_V \left(\overrightarrow{u}\cdot\nabla\right)\overrightarrow{u}\, dV + \frac{Pr}{Ra^{1/2}}\int\limits_V \triangle\overrightarrow{u} dV + \int\limits_V \overrightarrow{f}\, dV$$
$$\int\limits_V \overrightarrow{R}\left(\overrightarrow{u}^{\,n}\right) dV = -\int\limits_S \left(\overrightarrow{u}\right)\overrightarrow{u}\cdot\overrightarrow{n}\, dS + \frac{Pr}{Ra^{1/2}}\int\limits_S \nabla\overrightarrow{u}\cdot\overrightarrow{n} dS + \int\limits_V \overrightarrow{f}\, dV \qquad (9.4.11)$$

Thus, for each direction

$$\int\limits_V R\left(u^{\,n}\right) dV = -\left[(u)_e u_e S_e - (u)_w u_w S_w + (v)_n u_n S_n - (v)_s u_s S_s\right] + \frac{Pr}{Ra^{1/2}}\left[\frac{u_E - u_P}{\delta x_e}S_e - \frac{u_P - u_W}{\delta x_w}S_w + \frac{u_N - u_P}{\delta y_n}S_n - \frac{u_P - u_S}{\delta y_s}S_s\right] \qquad (9.4.12)$$

$$\int\limits_V R\left(v^{\,n}\right) dV = -\left[(u)_e v_e S_e - (u)_w v_w S_w + (v)_n v_n S_n - (v)_s v_s S_s\right] + \frac{Pr}{Ra^{1/2}}\left[\frac{v_E - v_P}{\delta x_e}S_e - \frac{v_P - v_W}{\delta x_w}S_w + \frac{v_N - v_P}{\delta y_n}S_n - \frac{v_P - v_S}{\delta y_s}S_s\right] + Pr\, T\, V_P \qquad (9.4.13)$$

where the velocity of the interface nodes is calculated as it has been explained before. However the temperature term needs some considerations. The temperature field is obtained at the central nodes (see Figure 9.4.1), so an approximation is needed to obtain the temperature value for the staggered grid in y-axis.

$$T_{staggered} = CDS\left(T_N, T_S\right) \qquad (9.4.14)$$

With the exception of the changes explained before, the Fractional step method can be applied without any problem. Nonetheless, the non-dimensional energy equation must be added. For that reason, at each loop, the program has to solve the convection-diffusion equation (Equation 5.2.6) but in that case the equation to be computed is Equation 9.4.5. Thus, the equations to be solved will be

$$\frac{\partial T}{\partial t} + \left(\overrightarrow{u}\cdot\nabla\right)T = \frac{1}{Ra^{1/2}}\nabla T + \rho S \qquad (9.4.15)$$

Therefore, S will be 0, Γ will be $\frac{Pr}{Ra^{1/2}}$ and diffusive and mass flow terms will be exactly the same as before for the centred grid excluding the density term.

Note that an explicit method is used to determine the velocity field and the pressure field[2], so the time increment must be exhaustively computed in order to achieved a converged solution which will be a consistent result. On the other hand, a CDS

[2]It is used the temperature of the previous time step to solve Equation 9.4.13.

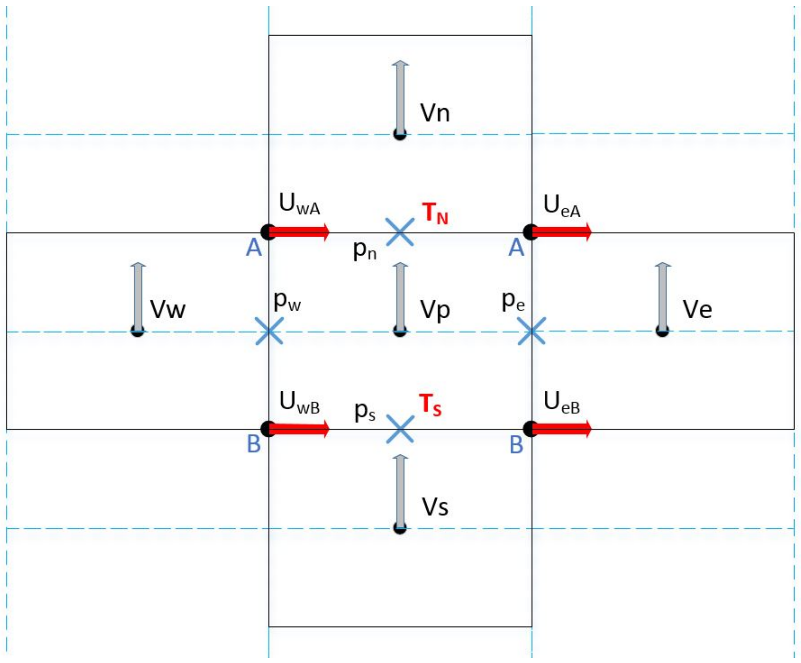


Figure 9.4.1.: Staggered mesh in y-axis with temperature term

scheme is used to determine the velocity on the interface nodes. A TDMA-GS solver has been implemented due to the fact that this type of problem implies a high computational cost to solve the Poisson equation.

On the other hand, an implicit method is used to determine the temperature field by solving the non-dimension convection-diffusion equation (Equation 9.4.15). In that case, a GS solver is used to find the temperature field.

9.5. The program

The program creates a class with the matrix of the coefficients to solve the Poisson equation. It also creates a class for the u component of the interface nodes for the staggered mesh in x direction, a class for the v component of the interface nodes for the staggered mesh in y direction, a class for the mass flow of the interface nodes, a class to calculate the velocity gradient of the interface nodes, a class for the diffusion term and a class for the Peclet number.

Finally, the program creates a class with all the information related to each grid node. Therefore, it contains u and v values for the three time steps, u^p and v^ppredictor velocity, the coordinate values of the grid node and interface nodes, the physical properties, the $\overrightarrow{R}\,(\overrightarrow{u})$values for the two previous time steps, the velocity values of the interface nodes, the mass flow values of the interface nodes, the velocity gradients values of the interface nodes, the coefficients of Equation 7.3.18, the temperature for two time steps, the diffusion terms of the interface nodes at the centre node grid, the mass flow term of the interface nodes for the central node grid, the Peclet number of the interface nodes and the coefficients for the convection-diffusion equation.

The inputs required by the program are the point coordinates, the number of nodes in each direction, the maximum number of iterations allowed, the convergence criterion, the Prandtl number, the Rayleigh number, the viscosity value, the density value, and finally the condition of the four boundaries for velocity - which can be wall or airflow -, and the boundaries for temperature - which can be Dirichlet or Neumann. If the boundary condition is airflow, it will also need the component values of u and v components. It also happens for the temperature if the boundary condition is Dirichlet, the program will need the value for the temperature. Another requirement needed by the program is the last time to solve the time loop.

The program consists in a file with different functions which calculate parts of the equations. The functions are explained below

- Main function: it contains the general algorithm that calls the different sub-functions to perform its function inside the algorithm. See Figure 9.5.1.
- Set Mesh function: it calculates the geometrical properties of each grid node and its interfaces using a uniform, orthogonal mesh.
- Initial Values function: it establishes the initial values of velocity and pressure.
- Evaluate boundary velocity function: it converts the inputs of the boundary conditions to velocity and predictor velocity.
- CDS scheme function: it returns the velocity value of the interface nodes with a specific indexes.
- Interface velocity function: it determines the velocity value of the interface nodes.
- Mass flow function: it calculates the value for the mass flow at each interface.
- Gradient velocity function: it determines the value for the velocity gradient of the interface nodes.
- Evaluate $\overrightarrow{R}(\overrightarrow{u})$ function: it calculates the value for Equation 7.3.4 and Equation 7.3.5.
- Evaluate u^p velocity function: it returns the value for the predictor velocity of each node of the staggered mesh.
- Pressure coefficients function: it calculates the coefficients taking into account the different conditions of each node. In this case there are also 9 different situations.

- TDMA Solver function: this code solves the equation Equation 7.3.18 using a TDMA-GS algorithm.
- Evaluate current velocity function: it calculates the current velocity field using Equation 7.3.26 and Equation 7.3.27.
- Evaluate diffusion term function: it determines the diffusion term for the convection-diffusion equation applied to the energy dimensionless conservation equation.
- Evaluate mass flow rate function: it calculates the mass flow terms for the centred node grid at the interfaces nodes.
- Evaluate Peclet function: it returns the Peclet number of the interface nodes.
- Temperature coefficients function: it calculates the coefficients taking into account the different conditions of each node. In this case there are also 9 different situations.
- TDMA Solver function: this code solves the equation Equation 5.5.1 using a TDMA-GS algorithm.
- Evaluate time step function: this function determines the time step for the next iteration using the parameters shown in Equation 7.4.1 and Equation 7.4.2.
- Evaluate convergence function: it determines the maximum absolute error made by the derivative of the velocity field respect with respect to the previous instant.
- New step function: this function saves the the current variables to the previous time step variables for the next iteration.
- Final results: it calculates the final results for the heat transferred along boundaries, maximum Nusselt number, maximum velocity components, and so on.
- Save values function: it exports a file with the velocity field and the pressure field.

9.6. The results

The parameters to compare are:

- u_{max}: the maximum horizontal velocity on the vertical mid-plane of the cavity with its location.

- v_{max}: the maximum vertical velocity on the horizontal mid-plane of the cavity with its location.
- $\overline{Nu}$: the average Nusselt number throughout the cavity.
- $Nu_{1/2}$: the average Nusselt number on the vertical mid-plane of the cavity.
- Nu_0: the average Nusselt number on the vertical boundary of the cavity at x=0.
- Nu_{max}: the maximum value of the local Nusselt number on the boundary at x=0 with its location.
- Nu_{min}: the minimum value of the local Nusselt number on the boundary at x=0 with its location.
- $\overrightarrow{u}\ field$: the contour plot of the velocity field.
- $T\ field$: the contour plot of the temperature field.
- $P\ field$: the contour plot of the pressure field.
- $\psi\ field$: the streamline plot of the stream function.

The heat flow across any line joining the horizontal boundaries must be equal to 0, due to the fact that the two horizontal boundaries are insulated. The heat flow through any line parallel to the y-axis is given by

$$Nu_x = \int_0^1 Q(x,y)\,dy \tag{9.6.1}$$

where Q is defined as $Q = uT - \frac{\partial T}{\partial x}$.

The average Nusselt number can be computed as

$$\overline{Nu} = \int_0^1 Nu_x\,dx \tag{9.6.2}$$

The integrals are computed using the trapezoidal method instead of the Simpson's rule used by G. de Vahl Davis.

After the code was created, it was validated with the parameters listed previously. The results for the u field and v field when it converged are shown below for different Rayleigh numbers (See Figure 9.6.1 and Annex B). Finally, the calculated solution can be compared with the benchmarking solution (See Table 9.6.1 and Table 9.6.2).

	Ra=10^3		Ra=10^4		Ra=10^5		Ra=10^6	
Parameter	bench	code	bench	code	bench	code	bench	code
u_{max}	3,6490	3,6497	16,1780	16,1847	34,7300	34,7591	64,6300	64,9699
$y_{u_{max}}$	0,8130	0,8134	0,8230	0,8234	0,8550	0,8532	0,8500	0,8483
v_{max}	3,6970	3,6975	19,6170	19,6266	68,5900	68,6356	219,3600	220,8337
$x_{v_{max}}$	0,1780	0,1766	0,1190	0,1169	0,0660	0,0672	0,0379	0,0373
$\overline{Nu}$	1,1180	1,1178	2,2430	2,2454	4,5190	4,5252	8,8000	8,8433
$Nu_{1/2}$	1,1180	1,1179	2,2430	2,2455	4,5190	4,5256	8,7990	8,8400
Nu_0	1,1170	1,1178	2,2380	2,2454	4,5090	4,5251	8,8170	8,8445
Nu_{max}	1,5050	1,5054	3,5280	3,5229	7,7170	7,6501	17,9250	16,8770
$y_{Nu_{max}}$	0,0920	0,0871	0,1430	0,1468	0,0810	0,0871	0,0378	0,0473
Nu_{min}	0,6920	0,6912	0,5860	0,5847	0,7290	0,7268	0,9890	0,9849
$y_{Nu_{min}}$	1,0000	0,9975	1,0000	0,9975	1,0000	0,9975	1,0000	0,9975

Table 9.6.1.: Comparative between benchmarking values [11, 12] and code values for differentially heated cavity

Parameter	Ra=10^3	Ra=10^4	Ra=10^5	Ra=10^6
u_{max}	-0,0193%	-0,0415%	-0,0839%	-0,5259%
$y_{u_{max}}$	-0,0532%	-0,0465%	0,2066%	0,2049%
v_{max}	-0,0129%	-0,0490%	-0,0664%	-0,6718%
$x_{v_{max}}$	0,7770%	1,7517%	-1,7639%	1,5477%
$\overline{Nu}$	0,0144%	-0,1072%	-0,1381%	-0,4924%
$Nu_{1/2}$	0,0125%	-0,1125%	-0,1470%	-0,4663%
Nu_0	-0,0745%	-0,3287%	-0,3569%	-0,3120%
Nu_{max}	-0,0291%	0,1455%	0,8672%	5,8468%
$y_{Nu_{max}}$	5,3645%	-2,6337%	-7,4873%	-25,0362%
Nu_{min}	0,1206%	0,2247%	0,2999%	0,4108%
$y_{Nu_{min}}$	0,2488%	0,2488%	0,2488%	0,2488%

Table 9.6.2.: Relative error between benchmarking values and code values for differentially heated cavity

9.7. Conclusions

The results are very accurate, they are highly similar to the results provided by G. de Vahl Davis [11, 12] obtaining relative errors lower than 10%. Nevertheless, the obtained error for the position of maximum Nusselt for $Ra = 10^6$ is too big. This may be due to the effects of turbulence or the convergence criterion of this specific case.

The left wall is at a hot temperature. Therefore, the flow is heated along the left wall and, by the effect of the flotation, the flow rises and rotates clockwise when it collides with the upper wall. It goes towards the right wall where the fluid is cooled and descends, reaching the low wall, where it turns towards the left, thus creating a circulatory flow.

Two different areas are appreciated in the pressure field, the upper zone where there are relative positive pressures due to the fact that the fluid is accumulated there, and the lower area where there are relative negative pressures.

When the Rayleigh number increases, the flow becomes more unstable until it reaches a critical Rayleigh for which the flow becomes turbulent. The flow is completely and smoothly laminar for a Rayleigh number of 10^3. On the other hand, the flow becomes almost turbulent, presenting certain fluctuations for high Rayleigh numbers such as $Ra = 10^6$. Even though the fluctuations, it converges. For high Rayleigh numbers, recirculation loops are appreciated. The cause of this turbulent behaviour is caused by the increase of the convective term with respect to the diffusive term.

This is, the conduction predominates over convection in heat transfer for low Rayleigh numbers, but the convection is the phenomenon that prevails in heat transfer when the Rayleigh number increases. This can be easily observed in the temperature field where the area of warm temperature originated from the warm wall becomes bigger at the top part, as well as in the area of cold temperature that comes from the cold wall at the low part.

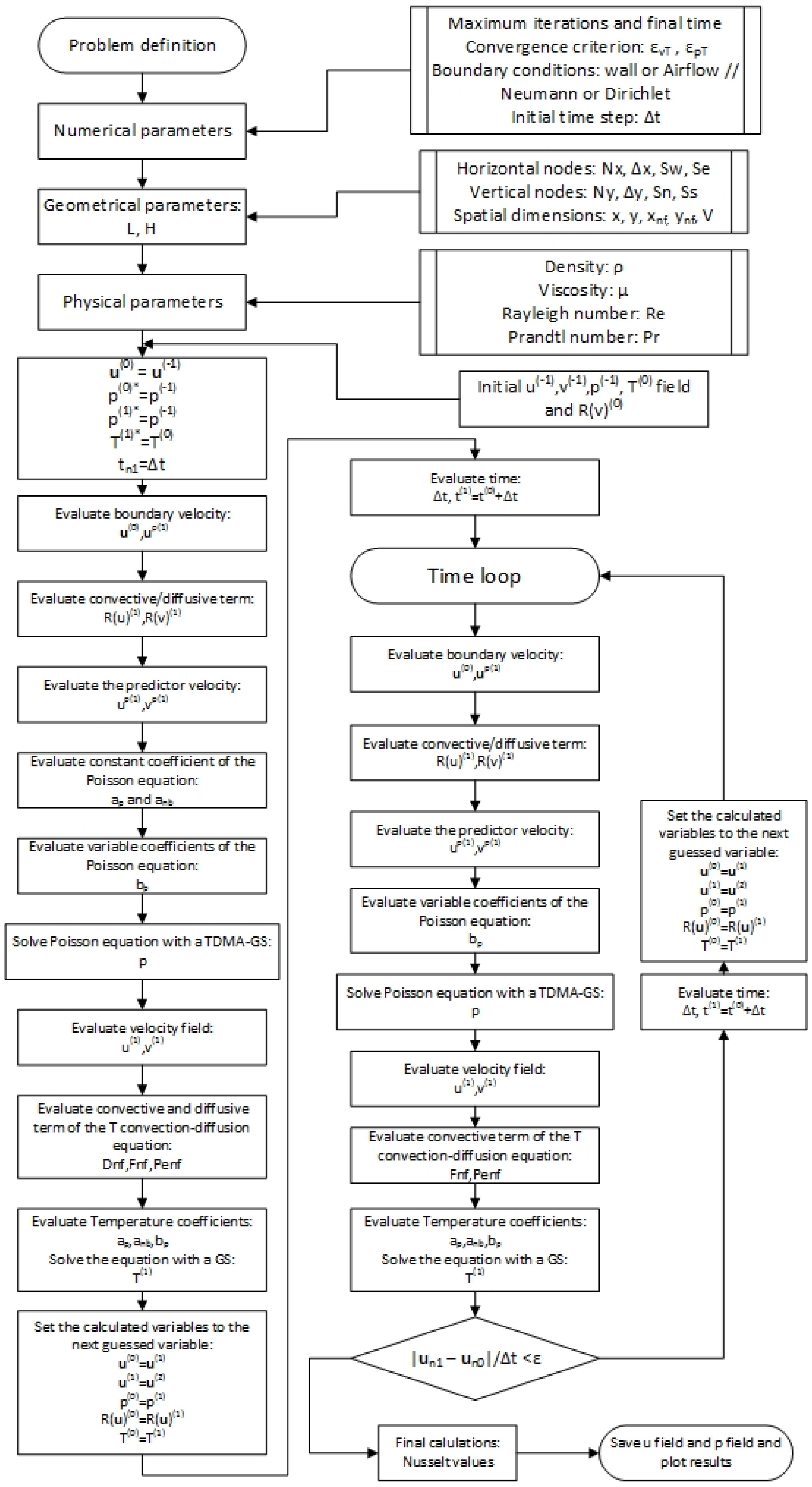
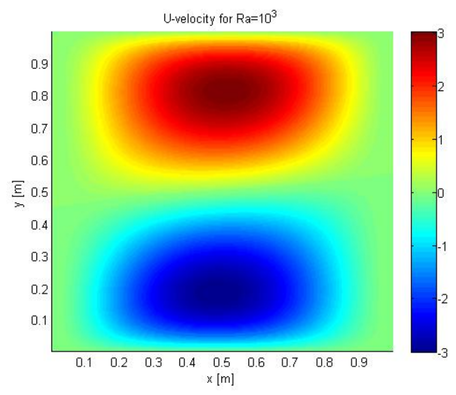
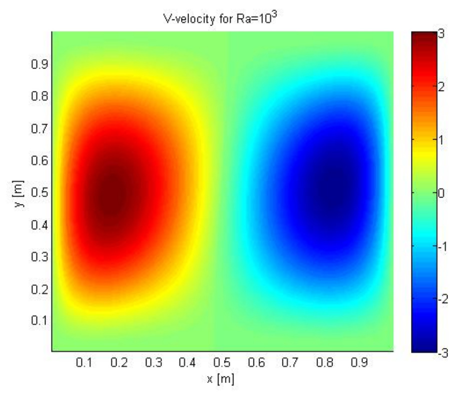

Problem definition
Numerical parameters
Maximum iterations and final time
Convergence criterion: ε_{vT} , ε_{pT}
Boundary conditions: wall or Airflow // Neumann or Dirichlet
Initial time step: Δt
Geometrical parameters: L, H
Horizontal nodes: Nx, Δx, Sw, Se
Vertical nodes: Ny, Δy, Sn, Ss
Spatial dimensions: x, y, x_{nf}, y_{nf}, V
Physical parameters
Density: ρ
Viscosity: μ
Rayleigh number: Re
Prandtl number: Pr
Initial $u^{(-1)}$,$v^{(-1)}$,$p^{(-1)}$, $T^{(0)}$ field and $R(v)^{(0)}$
$u^{(0)} = u^{(-1)}$
$p^{(0)*}=p^{(-1)}$
$p^{(1)*}=p^{(-1)}$
$T^{(1)*}=T^{(0)}$
$t_{n1}=\Delta t$
Evaluate boundary velocity: $u^{(0)}$,$u^{p(1)}$
Evaluate convective/diffusive term: $R(u)^{(1)}$,$R(v)^{(1)}$
Evaluate the predictor velocity: $u^{p(1)}$,$v^{p(1)}$
Evaluate constant coefficient of the Poisson equation: a_p and a_{nb}
Evaluate variable coefficients of the Poisson equation: b_p
Solve Poisson equation with a TDMA-GS: p
Evaluate velocity field: $u^{(1)}$,$v^{(1)}$
Evaluate convective and diffusive term of the T convection-diffusion equation: Dnf,Fnf,Penf
Evaluate Temperature coefficients: a_p,a_{nb},b_p
Solve the equation with a GS: $T^{(1)}$
Set the calculated variables to the next guessed variable:
$u^{(0)}=u^{(1)}$
$u^{(1)}=u^{(2)}$
$p^{(0)}=p^{(1)}$
$R(u)^{(0)}=R(u)^{(1)}$
$T^{(0)}=T^{(1)}$
Evaluate time: Δt, $t^{(1)}=t^{(0)}+\Delta t$
Time loop
Evaluate boundary velocity: $u^{(0)}$,$u^{p(1)}$
Evaluate convective/diffusive term: $R(u)^{(1)}$,$R(v)^{(1)}$
Evaluate the predictor velocity: $u^{p(1)}$,$v^{p(1)}$
Evaluate variable coefficients of the Poisson equation: b_p
Solve Poisson equation with a TDMA-GS: p
Evaluate velocity field: $u^{(1)}$,$v^{(1)}$
Evaluate convective term of the T convection-diffusion equation: Fnf,Penf
Evaluate Temperature coefficients: a_p,a_{nb},b_p
Solve the equation with a GS: $T^{(1)}$
$|u_{n1} - u_{n0}|/\Delta t < \varepsilon$
Set the calculated variables to the next guessed variable:
$u^{(0)}=u^{(1)}$
$u^{(1)}=u^{(2)}$
$p^{(0)}=p^{(1)}$
$R(u)^{(0)}=R(u)^{(1)}$
$T^{(0)}=T^{(1)}$
Evaluate time: Δt, $t^{(1)}=t^{(0)}+\Delta t$
Final calulations: Nusselt values
Save u field and p field and plot results

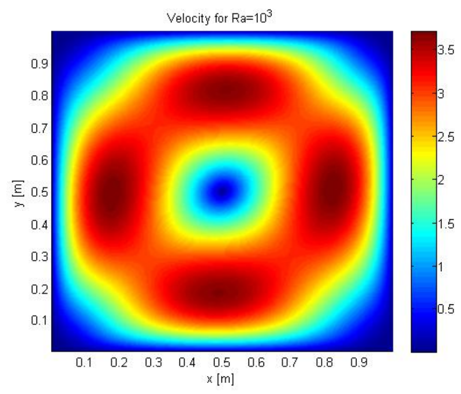
Figure 9.5.1.: Differentially heated cavity algorithm



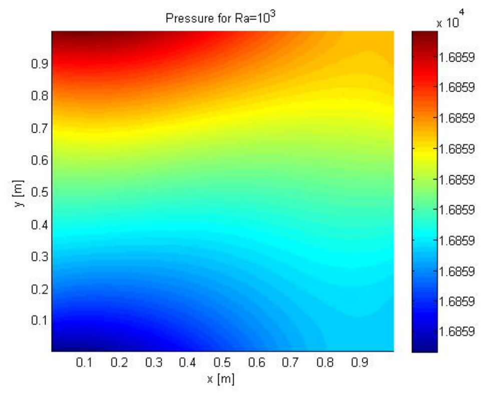
(a) U field



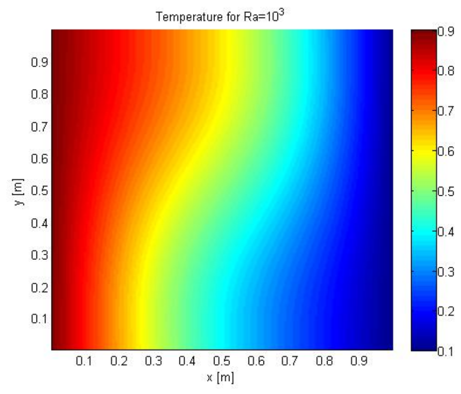
(b) V field



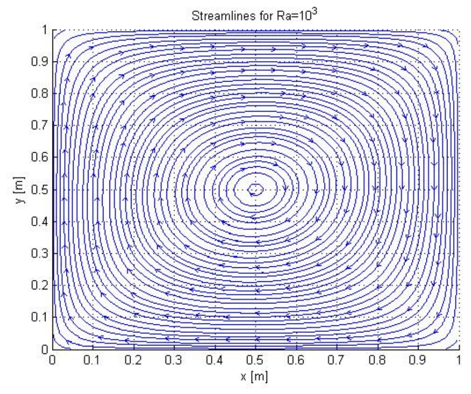
(c) Velocity field



(d) Pressure field



(e) Temperature field



(f) Stream function

Figure 9.6.1.: Results for Ra=10^3

Part V.

Turbulent heat convection transfer

10. Turbulent heat convection

10.1. Introduction

Most of the flows encountered in engineering become unstable above a certain Reynolds number (Re_{crit}). Below that critical Reynolds number, flows are laminar. These instabilities cause a chaotic and random state of motion in which the velocity, pressure and temperature change continuously with time. This regime is called turbulent flow.

Turbulent flows vary in a random and chaotic way, although they are commonly periodic and achieve the convergence when the variables are statistically stationary. For that reason, the velocity and other properties are studied as statistical variables composed by a steady mean value Φ and a fluctuating component over time $\phi'(t)$: $\phi(t) = \Phi + \phi'(t)$. This decomposition is called the Reynolds decomposition.

Turbulent fluctuations always have a three-dimensional spatial character creating rotational flow structures which are called turbulent eddies. These structures are composed by a wide range of length scales which scale with the Reynolds number according to Kolmogorov K41: $\delta x \sim Re^{-3/4}$ and $\delta t \sim Re^{-1/2}$. Thereupon, if all the turbulent eddies are computed (DNS simulation), the perfect code will require a memory capacity that will grow with $Re^{9/4}$ for a 3D problem. Moreover, the computational cost will grow with $Re^{11/4}$ [9].

This enormous computational cost is unfeasible for most computers. It can only be computed on supercomputers and and the calculations last thousands of hours. For that reason, scientists and engineers are doing researches in order to develop numerical methods to capture the turbulent effects with a lower computational cost. The numerical methods are listed below [32]:

- Turbulence models for Reynolds-averaged Navier-Stokes (RANS) equations: this method focuses on the mean flow and the effects of turbulence on mean flow properties. RANS equations are composed by the time averaged of the Navier-Stokes and other extra terms to take into account the interaction between turbulent fluctuations, which are determined by turbulence models like $k - \varepsilon$

model. The computational cost for accurate solutions are modest. Therefore, it has been a good approach to solve the turbulence.

- Large eddy simulation (LES) [10]: this method focuses on the behaviour of larger eddies as well as the mean flow. Thus, the Navier-Stokes equations are filtered in order to compute the larger eddies and rejects the smaller eddies. The effect of these small unresolved eddies is included by means of a sub-grid scale model. The computational cost is high, although it is lower than the DNS. For that reason, it is starting to tackle complex CFD problems.

- Direct numerical simulation (DNS): this method focuses on the behaviour of all turbulent eddies as well as the mean flow. This method needs a fine grid in order to solve the Kolmogorov length scales and sufficiently small time steps to be able to resolve the fastest fluctuations. The computational cost is enormous as it has been shown before, so it is only used for high accuracy flows in high-tech industries.

10.2. Burgers model equation in Fourier Space for 1D

The Navier-Stokes equations provide with an appropriate model for the nonlinear dynamics of turbulence, although their direct numerical simulation (DNS) is difficult and very expensive in terms of computational cost because the convective term produces too many dynamically relevant scales of motion. For that reason, a simplified equation based on Burgers equation is used to solve a LES simulation. The Burgers equation is going to be studied.

10.2.1. Burger equation in Fourier space

The Burgers equation is a simplified model that shares many of the aspects of the N-S equations. It is shown below for one-dimensional problem [9]

$$\partial_t u + u\partial_x u = \frac{1}{Re}\partial_{xx} u + f \tag{10.2.1}$$

where ∂_t is the time partial derivative, ∂_x is the first order spatial partial derivative, ∂_{xx} is the second order spatial partial derivative, u is the horizontal velocity, Re is the Reynolds number, and f is the forcing term.

In Fourier space, the previous equation can be rewritten for an internal Ω with periodic boundary conditions as

$$\partial_t \hat{u}_k + \sum_{k=p+q} \hat{u}_p iq \hat{u}_q = -\frac{k^2}{Re}\hat{u}_k + F_k \qquad k = 0, ..., N \tag{10.2.2}$$

where N is the total number of Fourier modes, k represents the k-th Fourier mode, as well as the q and p subscripts, $F_k \in \mathbb{C}$ is the forcing term which is equal to 0 for $k \neq 1$ and a value such that $\partial_t \hat{u}_1$ equals 0 for $k = 1$, and $\hat{u}_k \in \mathbb{C}$ is the k-th Fourier coefficient of $u(x,t) \in \mathbb{R}$. The velocity field can be defined as

$$u\left(x,t\right) = \sum_{k=-N}^{k=N} \hat{u}_k\left(t\right) e^{ikx} \tag{10.2.3}$$

The complex conjugate must be done to $\hat{u}_k$ for the negative Fourier modes in order to assure that $u(x,t)$ will be a real number.

Equation 10.2.2 can be integrated over time using a fully explicit time-integration scheme, which leads to

$$\frac{\hat{u}_k^1 - \hat{u}_k^0}{\Delta t} = -\sum_{k=p+q} \hat{u}_p^0 iq \hat{u}_q^0 - \frac{k^2}{Re}\hat{u}_k^0 + F_k^0 \tag{10.2.4}$$

where $F_1 = \sum_{1=p+q} \hat{u}_p^0 iq \hat{u}_q^0 + \frac{1^2}{Re}\hat{u}_1^0$.

10.2.2. Kinetic energy transport equation

The energy $E_k \in \mathbb{R}$ of the k-th mode is obtained by taking the product of $\hat{u}_k$ with its complex conjugate. Then, the energy is obtained by the kinetic energy transport equation [9], defined as

$$\partial_t E_k = -\frac{2k^2}{Re}E_k - \left(\overline{\hat{u}_k}\, C_k\left(\hat{u}_p, \hat{u}_q\right) + \hat{u}_k\, \overline{C_k\left(\hat{u}_p, \hat{u}_q\right)}\right) + \left(\overline{\hat{u}_k} F_k + \hat{u}_k \overline{F_k}\right) \tag{10.2.5}$$

where

- The first term represents the variation of the kinetic energy along time.
- The second term is the diffusive term, which is damping energy. This is, this phenomenon is eliminating the smaller scales and it becomes more effective for the high-frequency modes (small scales) due to the fact that its coefficient scales with k^2.
- The third term is the convective term, which is transporting the energy from one to other scales, specifically from large scales to small scales. The $C_k\left(\hat{u}_p, \hat{u}_q\right) \in \mathbb{C}$ is the convective contribution and can be defined as $C_k\left(\hat{u}_p, \hat{u}_q\right) = \sum_{k=p+q} \hat{u}_p iq \hat{u}_q$.

- The fourth term is the forcing energy.

Note that $k = p + q$ represents a straight line with a unit slope. For that reason, if p increases one unit, q must decrease one unit in order to maintain constant the difference between them. It is not necessary to solve the negative modes due to the fact that they can be obtained by using the complex conjugate. Only the positive modes between 0 and N must be solved.

Equation 10.2.5 can be integrated over time using a fully explicit time-integration scheme which leads to

$$\frac{E_k^1 - E_k^0}{\Delta t} = -\frac{2k^2}{Re} E_k^0 - \left(\overline{\hat{u}_k^0} C_k^0 \left(\hat{u}_p, \hat{u}_q \right) + \hat{u}_k^0 \overline{C_k^0 \left(\hat{u}_p, \hat{u}_q \right)} \right) + \left(\overline{\hat{u}_k^0} F_k^0 + \hat{u}_k^0 \overline{F_k^0} \right) \quad (10.2.6)$$

The previous equation depends on the Reynolds number, being the Reynolds number a relation between the convective term and the diffusive term. Therefore, for higher Reynolds number, the convective term will increase, and therefore more turbulence scales will be generated to higher frequencies. This will make necessary the use of a higher number of spectral modes to cover all the turbulence scales and obtain an accurate result.

10.2.3. Large-Eddy Simulation (LES) for 1D

The Smagorinsky model is the simplest LES model that can be written for a 1D problem [9] as

$$\partial_t \overline{u} + \overline{u} \partial_x \overline{u} = \nu \partial_{xx} \overline{u} + f - \partial_x \tau \left(u \right) \quad (10.2.7)$$

where $\overline{u}$ is the filtered velocity, ν is the nondimensional kinematic viscosity which is equal to $\frac{1}{Re}$, and $\tau(u)$ is the subfilter tensor which for Smagorinsky model can be modelled as

$$\begin{aligned} \tau \left(u \right) &\approx \nu_t \partial_x \overline{u} & (10.2.8) \\ \nu_t &= l_s^2 \sqrt{2 \left(\partial_x \overline{u} \right)^2} & (10.2.9) \end{aligned}$$

where ν_t is the eddy-viscosity, and l_s is the Smagorinsky lengthscale and equals $l_s = C_s \Delta$; where C_s is the Smagorinsky constant that usually takes values between 0.1 and 0.2, and Δ is the equivalent filter width.

An spectral eddy-viscosity is used instead of the eddy-viscosity in order to be able to apply the Smagorinsky model in Fourier space. The formula for the spectral eddy-viscosity is shown below

$$\nu_t\left(k/k_N\right) = \nu_t^{+\infty}\left(\frac{E_{k_N}}{k_N}\right)^{1/2}\nu_t^* \tag{10.2.10}$$

$$\nu_t^{+\infty} = 0.31\frac{5-m}{m+1}\sqrt{3-m}C_K^{-3/2} \tag{10.2.11}$$

$$\nu_t^*\left(\frac{k}{k_N}\right) = 1+34.5e^{-3.03(k_N/k)} \tag{10.2.12}$$

where E_{k_N} is the energy at the cut-off frequency, k_N is the cut-off frequency, ν_t^* is a non-dimensional eddy-viscosity, m is the slope of the energy spectrum, and C_K is the Kolmogorov constant for 1D Burgers equations which is equal to $C_K \approx 0.4523$.

A combination of Equation 10.2.7 and Equation 10.2.9 leads to

$$\partial_t\overline{u} + \overline{u}\partial_x\overline{u} = (\nu+\nu_t)\,\partial_{xx}\overline{u} + f \tag{10.2.13}$$

Equation 10.2.13 can be integrated over time using a fully explicit time-integration scheme which leads to

$$\frac{\hat{u}_k^1-\hat{u}_k^0}{\Delta t} = -\sum_{k=p+q}\hat{u}_p^0 iq\hat{u}_q^0 - k^2\left(\nu+\nu_t\right)\hat{u}_k^0 + F_k^0 \tag{10.2.14}$$

This turbulent viscosity $\nu_t(k)$ represents an addend in the convective term with the purpose to fulfill the difference of energy removed between a DNS simulation and a LES simulation, that is to say, the difference between the energy that would vanish in a simulation in which all the spectral modes were considered regarding a simulation in which a specific number of spectral modes was considered. Although, this turbulent viscosity has to depend on the spectral mode with the purpose to avoid modifications of the spectrum for small frequencies (large scales), which offers the macroscopic aspect.

11. A specific problem: 1D LES problem

The case chosen to be solved and to prove the previous equations is an unsteady case that consists of a one-dimensional Burgers equation problem. The Burgers equation will be solved for a Reynolds number of 40, an initial null mean flow ($\hat{u}_0 = 0$) and the other k-th Fourier coefficient will be equal to $\hat{u}_k = k^{-1} + 0.0i$. The initial kinetic energy will be equal to $E_k = \hat{u}_k \cdot \overline{\hat{u}}_k$.

The problem is solved for Equation 10.2.2, Equation 10.2.13 with $C_K = 0.4523$ and Equation 10.2.13 with $C_K = 0.05$.

11.1. The program

The program creates a class with all the information related to each Fourier mode. Therefore, it contains the $\hat{u}_k$ values for the previous and current time steps, the F_k for the current time step and the E_K values for the previous and the current time steps.

The inputs required by the program are the total number of Fourier modes, the maximum number of allowed iterations, the convergence criterion for velocity and energy equations, the Reynolds number, and the time constant for the CFL-like condition: $\Delta t < C_1 \frac{Re}{N^2}$. Another needed requirement by the program is the last time to solve the time loop. The program also requires the C_K, k_N and m in case the LES equation is solved.

The program consists in a file with different functions which calculate parts of the equations. The functions are explained below

- Main function: it contains the general algorithm that calls the different sub-functions to perform its function inside the algorithm. See Figure 11.1.1.
- Initial values function: it establishes the initial values of velocity and energy.
- Evaluate $\hat{u}_k$ function: it calculates the value for Equation 10.2.2 or Equation 10.2.7 of each Fourier mode.
- Evaluate E_K function: it returns the value for the kinetic energy of each Fourier mode solving Equation 10.2.5.

- Evaluate $\nu_{eff} = \nu + \nu_t$ function: it determines the value of this variable for each time step.
- Evaluate time step function: this function determines the time step for the next iteration using the parameters of CFL-like condition.
- Evaluate convergence function: it determines the maximum absolute error made by the derivative of the velocity or the derivative of the energy versus time.
- New step function: this function saves the current variables to the previous time step variables for the next iteration.
- Save values function: it exports a file with the velocity field and the pressure field.

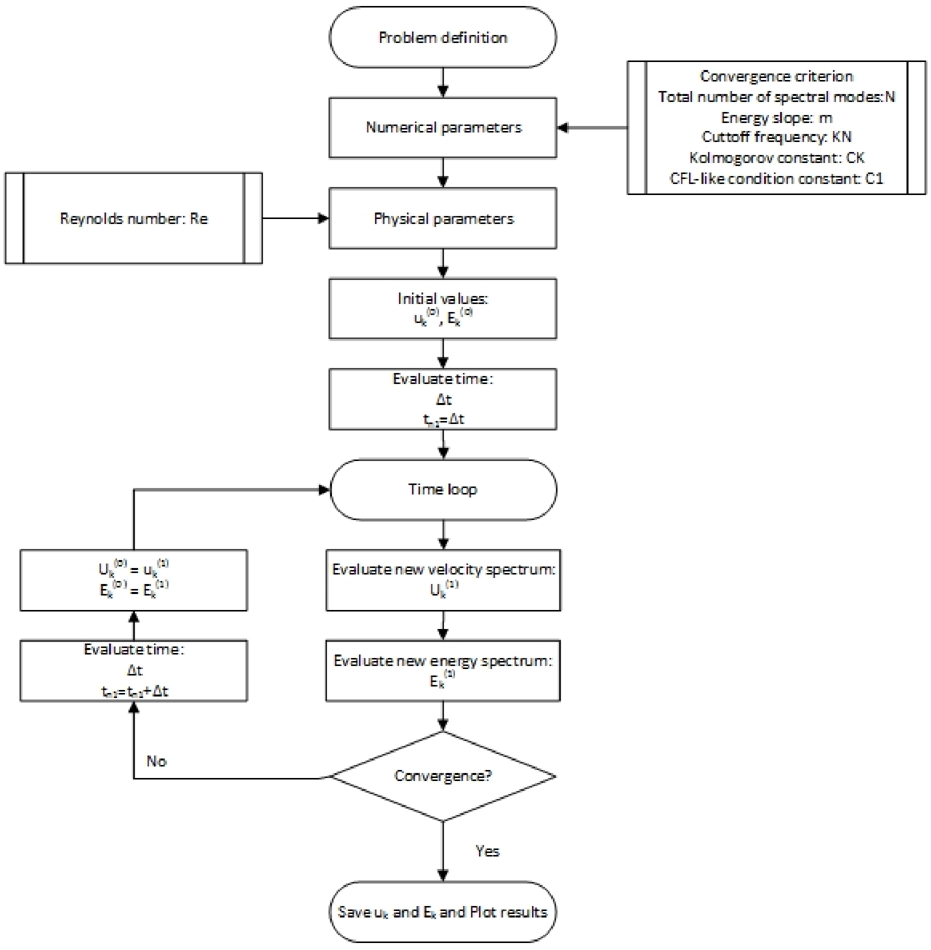

Problem definition
Convergence criterion
Total number of spectral modes:N
Energy slope: m
Cuttoff frequency: KN
Kolmogorov constant: CK
CFL-like condition constant: C1
Numerical parameters
Reynolds number: Re
Physical parameters
Initial values: $u_k^{(0)}$, $E_k^{(0)}$
Evaluate time: Δt $t_{n+1}=\Delta t$
Time loop
$U_k^{(0)} = u_k^{(1)}$
$E_k^{(0)} = E_k^{(1)}$
Evaluate new velocity spectrum: $U_k^{(1)}$
Evaluate time: Δt $t_{n+1}=t_{n+1}+\Delta t$
Evaluate new energy spectrum: $E_k^{(1)}$
No
Convergence?
Yes
Save u_k and E_k and Plot results

Figure 11.1.1.: 1D LES problem algorithm

11.2. The results

The results are shown in Table 11.2.1.

k-mode	Burgers Equation		LES with $C_K = 0.4523$		LES with $C_K = 0.05$	
	$\hat{u}_k$	E_k	$\hat{u}_k$	E_k	$\hat{u}_k$	E_k
0	-6.759e-016 + 1.541e-015 i	2.828E-30	1.061e-018 - 4.948e-015 i	2.449E-29	-4.325e-019 - 9.227e-016 i	8.511E-31
1	1.000e+000 + 0.000e+000 i	1.000E+0	1.000e+000 + 0.000e+000 i	1.000E+0	1.000e+000 + 0.000e+000 i	1.000E+0
2	3.905e-002 - 3.428e-001 i	1.191E-1	-2.144e-019 - 3.978e-001 i	1.583E-1	8.732e-020 - 3.949e-001 i	1.559E-1
3	-2.722e-001 + 8.545e-003 i	7.413E-2	-2.551e-001 + 2.430e-019 i	6.508E-2	-2.503e-001 - 9.860e-020 i	6.267E-2
4	-3.412e-002 + 1.597e-001 i	2.668E-2	2.532e-019 + 1.882e-001 i	3.543E-2	-1.009e-019 + 1.821e-001 i	3.317E-2
5	1.591e-001 - 7.087e-003 i	2.536E-2	1.493e-001 - 2.562e-019 i	2.228E-2	1.417e-001 + 1.006e-019 i	2.008E-2
6	3.193e-002 - 1.076e-001 i	1.260E-2	-2.585e-019 - 1.236e-001 i	1.528E-2	9.894e-020 - 1.147e-001 i	1.317E-2
7	-1.100e-001 + 5.719e-003 i	1.213E-2	-1.055e-001 + 2.587e-019 i	1.113E-2	-9.540e-002 - 9.668e-020 i	9.101E-3
8	-3.052e-002 + 8.398e-002 i	7.984E-3	2.600e-019 + 9.211e-002 i	8.484E-3	-9.404e-020 + 8.080e-002 i	6.529E-3
9	8.202e-002 - 4.525e-003 i	6.748E-3	8.179e-002 - 2.598e-019 i	6.690E-3	6.934e-002 + 9.110e-020 i	4.808E-3
10	2.954e-002 - 7.151e-002 i	5.987E-3	-2.617e-019 - 7.373e-002 i	5.436E-3	8.778e-020 - 5.998e-002 i	3.597E-3
11	-6.360e-002 + 3.420e-003 i	4.057E-3	-6.714e-002 + 2.615e-019 i	4.508E-3	-5.202e-002 - 8.393e-020 i	2.706E-3
12	-2.895e-002 + 6.515e-002 i	5.083E-3	2.648e-019 + 6.190e-002 i	3.831E-3	-7.935e-020 + 4.501e-002 i	2.026E-3
13	5.001e-002 - 2.335e-003 i	2.507E-3	5.726e-002 - 2.636e-019 i	3.279E-3	3.864e-002 + 7.388e-020 i	1.453E-3
14	2.887e-002 - 6.361e-002 i	4.879E-3	-2.699e-019 - 5.379e-002 i	2.893E-3	6.752e-020 + - 3.277e-002 i	1.074E-3
15	-3.840e-002 + 1.293e-003 i	1.476E-3	-5.005e-002 + 2.646e-019 i	2.505E-3	-2.736e-002 + - 6.042e-020 i	7.488E-4
16	-2.996e-002 + 6.832e-002 i	5.566E-3	2.806e-019 + 4.840e-002 i	2.342E-3	-5.296e-020 + 2.249e-002 i	5.060E-4
17	2.525e-002 - 7.487e-004 i	6.379E-4	4.410e-002 - 2.591e-019 i	1.945E-3	1.824e-002 + 4.559e-020 i	3.326E-4
18	3.488e-002 - 8.690e-002 i	8.768E-3	-3.204e-019 - 4.778e-002 i	2.283E-3	3.885e-020 - 1.468e-002 i	2.156E-4
19	-2.406e-003 + 3.202e-003 i	1.604E-5	-3.516e-002 + 2.136e-019 i	1.236E-3	-1.180e-002 - 3.282e-020 i	1.392E-4
20	-5.786e-002 + 1.540e-001 i	2.707E-2	5.715e-019 + 7.617e-002 i	5.802E-3	-3.320e-020 + 1.114e-002 i	1.242E-4

Table 11.2.1.: Results for the Burgers equation

The graphical results are shown in Figure 11.2.1, and they can be compared with the expected results shown in Figure 11.2.2.

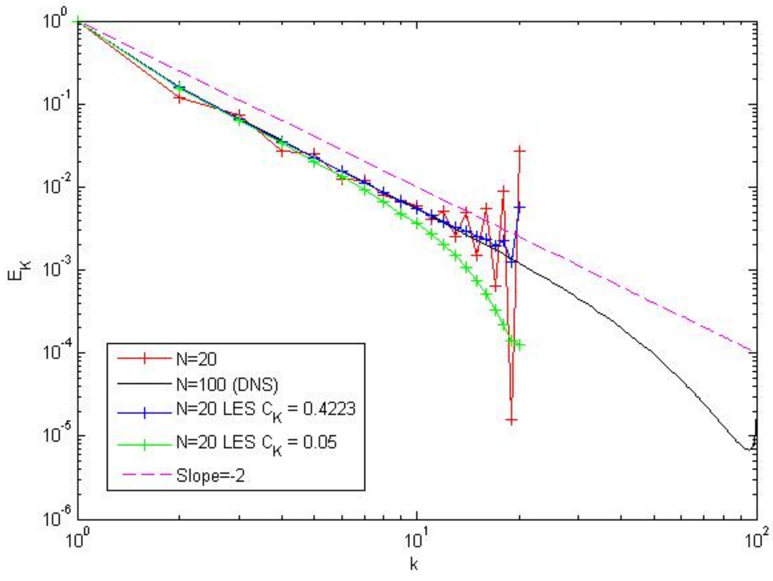


Figure 11.2.1.: Graphical results for the Burgers equation

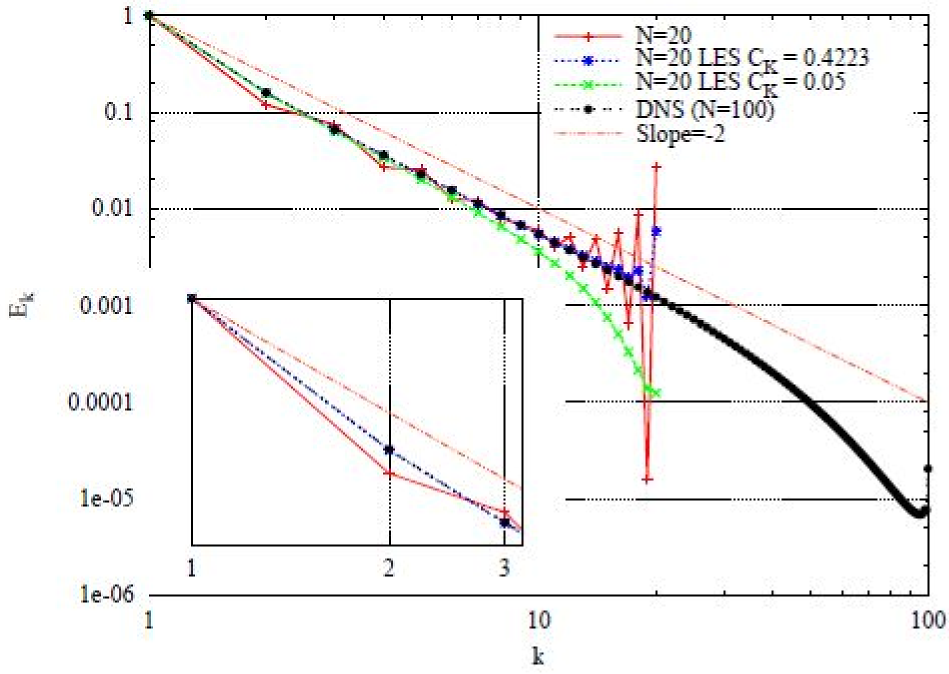


Figure 11.2.2.: Expected graphical results for the Burgers equation [9]

11.3. Conclusions

Figure 11.2.2 shows the values of the kinetic energy in a wide range of Fourier modes including a DNS simulation with 100 modes. The DNS simulation represents the energy of almost all scales of turbulence, but the LES simulation only represents the energy of big scales instead. For that reason, the DNS simulation is considered an accurate solution while the LES simulation is only an approximation. Therefore, the LES solution should be as close as possible to the DNS solution. It is clear that the LES solution with $C_K = 0.4523$ is a better solution than the LES solution with $C_K = 0.05$.

On the other hand, the large scale motions are generally much more energetic than the small scale ones. The size and the strength of the large scale motions make them by far the most effective transporters of the conserved properties, instead of the smallest scales.

Part VI.

Radiative heat transfer

12. Radiative heat transfer

12.1. Introduction

Radiative heat transfer can be explained as the variation of the energy of any system due to absorption or emission of electromagnetic waves with a specific frequency (ν) which propagate at the speed of light c [1][4, 24, 25]. These waves consist of a group of massless particles called photons that carry a specific amount of energy ($e_\lambda = h\nu$ where h is the Planck's constant $h = 6.626 \cdot 10^{-34} Js$). Therefore, this phenomenon can be understood as an emission of a number of photons, or an absorption of photons which travel in straight lines.

The radiative heat transfer term is located in the heat flow term of the energy conservation equation. This term $\dot{q}$ is defined as the conduction heat flow term, as well as the radiative heat flow. The energy conservation equation is shown below

$$\frac{\partial}{\partial}\int\limits_V (u+e_c)\,\rho\, dV + \int\limits_S (u+e_c)\,\rho \overrightarrow{u}\cdot\overrightarrow{n}\, dV = -\int\limits_S \dot{q}\cdot\overrightarrow{n}\, dS + \int\limits_S \overrightarrow{u}\cdot\overrightarrow{f_{(\overrightarrow{n})}}\, dS + \int\limits_V \overrightarrow{u}\cdot\overrightarrow{b}\, dV \tag{12.1.1}$$

The amount of radiative heat transfer depends on the difference of the fourth power of the temperature, so the radiative heat transfer must be taken into account if there are large differences with high temperature values. It should be also considered in vacuum applications. Nevertheless, this term can be neglected if the walls have highly reflective properties.

The electromagnetic radiation spectrum can be divided into different regions: low wavelength region ($\lambda < 0.1\mu m$), thermal radiation zone ($0.1\mu m \le \lambda \le 100\mu m$) and high wavelength region ($\lambda > 100\mu m$). In turn, the thermal radiation area can be separated into several bands: the ultraviolet band ($\lambda < 0.4\mu m$), the visible or solar band ($0.4\mu m < \lambda < 0.7\mu m$) and the infrared band ($\lambda > 0.7\mu m$). For more information, see Figure 12.1.1.

The energy density I is the number of photons considering a wavelength between λ

[1] In any other media than the vacuum, there is a reflective index that determines the ratio of the speed of light in vacuum and the speed in the current media, $n = c_0/c$. This fraction is obtained from the Snell's law.

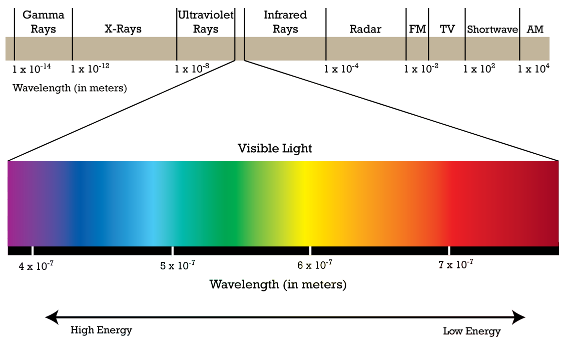


Figure 12.1.1.: The electromagnetic radiation spectrum

and $\lambda + d\lambda$, crossing an area dA perpendicular to the photon direction $\overrightarrow{s}$,within a solid angle $d\Omega$ per unit time dt and unit wavelength $d\lambda$, multiplied by the energy of each photon. Therefore, the energy density is obtained with the next formula

$$I_{\lambda\omega} = \frac{energy\,of\,wavelength\,\lambda\,trhough\,an\,area\,normal\,to\,\overrightarrow{s}}{dA\,d\Omega\,d\lambda\,dt} = n_{\lambda\omega}e_{\lambda}c\ \left[\frac{W}{m^2\cdot sr\cdot m}\right], \quad (12.1.2)$$

The total energy through a surface perpendicular to a given vector $\overrightarrow{n}$ (See Figure 12.1.2), per unit time and area is computed as

$$\dot{q} = \int_0^\infty d\lambda \int_{4\pi} d\Omega\,(\overrightarrow{n}\cdot\overrightarrow{s})\,I_{\lambda\omega} \quad (12.1.3)$$

The total energy will be positive in case it is incoming energy. However, the total energy will be negative in case it is outgoing energy.

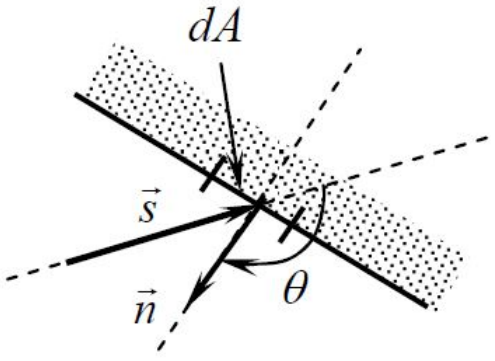


Figure 12.1.2.: Radiative heat transfer system

The energy density I can be applied for a black body, which yields

$$I_b\,(T,\lambda) = \frac{2hc^2}{\lambda^5}\frac{1}{\exp\,(hc/\lambda kT)-1} \quad (12.1.4)$$

where h is the Planck's constant, c is the speed of light, λ is the wavelength, k is the Boltzmann constant ($k = 1.380658 \cdot 10^{-23} J/K$), and T is the temperature.

On the other hand, the total hemispherical energy emitted by a black body per unit area, time and wavelength can be determined as

$$\dot{q}_b\left(T,\lambda\right) = \pi I_b\left(T,\lambda\right) \tag{12.1.5}$$

$$I_b\left(T\right) = \frac{\sigma_B T^4}{\pi} \tag{12.1.6}$$

where σ_B is the Stefan-Boltzmann constant, $\sigma_B = 5.6705 \cdot 10^{-8} W/m^2K^4$. Wien's law determines that the product between temperature and the wavelength for maximum intensity is constant. The formula is $\lambda_{max} T = 2897.82 \mu mK$.

Each surface will radiate a certain amount of energy, which will depend on its temperature, the properties of the surface, the direction of the wavelength, etc. Therefore, it can be defined an emission energy relation between the energy emitted by the surface and the energy that would radiate a black body. That ratio of energy is called emissivity:

$$\varepsilon = \frac{I_{emmited}\left(T,\lambda,\overrightarrow{s}\right)}{I_b\left(T,\lambda\right)} \tag{12.1.7}$$

On the other hand, each surface will absorb a certain amount of the incident energy, it will reflect other quantity of this incident energy, and finally it will transmit the remaining quantity through the material (all these parameters depend on the direction and wavelength). Therefore, the fractions can be defined as

$$\alpha = \frac{I_{absorbed}}{I_{incoming}} \tag{12.1.8}$$

$$\tau = \frac{I_{transmitted}}{I_{incoming}} \tag{12.1.9}$$

$$\rho = \frac{I_{reflected}}{I_{incoming}} \tag{12.1.10}$$

where α is the absorptivity, τ is the transmissivity, and ρ is the reflectivity. The sum of these three fractions must be equal to 1 if the energy is to be conserved. Note that these fractions may vary with the incident angle as well as the wavelength of the original wave. Moreover, the reflectivity can be split into two terms: the diffused component ρ_d, an intensity equally distributed for all possible outgoing directions; and the specular component ρ_s, intensity reflected in the same angle of the incident wave. The Kirchoff law[2] assumes that $\alpha\left(T,\lambda,\overrightarrow{s}\right) = \varepsilon\left(T,\lambda,\overrightarrow{s}\right)$ if it is at thermal equilibrium.

[2]Kirchoff's law states that a body in thermodynamic equilibrium emits as much energy as it absorbs in each direction and at each wavelength. For a diffuse body, the emmitance and absorptance do not depend on the angles, so $\varepsilon_\lambda\left(T\right) = \alpha_\lambda\left(T\right)$. If, in addition, the body is gray, the law becomes $\varepsilon\left(T\right) = \alpha\left(T\right)$.

12.2. The radiosity-Irradiosity method

This method is used when the media can be considered transparent. In other words, the media does not participate in the radiative heat transfer. It also has the diffuse surface assumption implicit.

This method does not need to solve the radiative transfer equation, but it calculates a set of purely geometrical entities and computes the energy balance of each surface instead [24]. That geometrical entities, which are called view factors, depend only on the relative location and the orientation of the surfaces of the domain, and are defined as

$$F_{dA_1 \to dA_2} = \frac{energy\,emitted\,by\,dA_1\,and\,intercepted\,by\,dA_2}{total\,energy\,emitted\,by\,dA_1} \tag{12.2.1}$$

The total energy leaving dA_1that is intercepted by dA_2, $F_{dA_1 \to dA_2} dA_1$, is equal to the total energy leaving dA_2that is intercepted by dA_1, $F_{dA_2 \to dA_1} dA_2$. This criteria is called the reciprocity law. The view factor between two finite areas (See Figure 12.2.1) can be computed as

$$F_{A_i \to A_j} = \frac{1}{A_i} \int_{A_j} \int_{A_i} \frac{\cos(\theta_i)\cos(\theta_j)}{\pi s_{ij}^2} dA_i dA_j \tag{12.2.2}$$

where A_i is the emitting area, A_j is the receiving area, θ_i is the angle between the vector $\overrightarrow{s_{ij}}$ and the normal $\overrightarrow{n_i}$, θ_j is the angle between the vector $\overrightarrow{s_{ij}}$ and the normal $\overrightarrow{n_j}$, $\overrightarrow{s_{ij}}$ is the vector that joins the centre of the two elements, $\overrightarrow{n_i}$ is the normal vector for A_i area and $\overrightarrow{n_j}$ is the normal vector for A_j. Applying the conservation of energy in each area, it is clear that

$$\sum_{j=1}^{N} F_{i \to j} = 1 \quad \forall i \tag{12.2.3}$$

where N is the total number of surfaces.

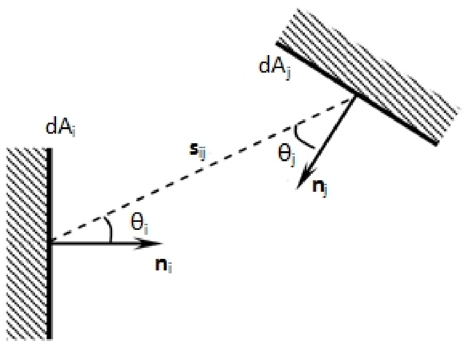


Figure 12.2.1.: The view factor

After obtaining the view factors, an energy balance is made for each surface, which will relate the energy emitted by one surface to the energy received from the other surfaces. Thus, the radiosity of a surface, j_K, is defined as the energy that leaves the surface per unit area. This component can be expressed as

$$j_K = \varepsilon_K \dot{q}_{b,K} + \rho_K g_K \tag{12.2.4}$$

$$\dot{q}_{b,K} = \sigma_B T_K^4 \tag{12.2.5}$$

where ε_K is the emissivity of the surface K, $E_{b,K}$ is the black body emission at $\mathrm{T_K}$, ρ_K is the reflectivity of the surface K, and g_K is the irradiosity of the surface K. The irradiosity is defined as the total incident energy on the surface K per unit area. Therefore,

$$g_K = \sum_{l=1}^{N} F_{K\to l}\, j_l \tag{12.2.6}$$

A linear system with N unknowns, the radiosities j_K, can be defined with the radiosities and irradiosities of each surface. Once it is solved, the total heat flux per unit area for each surface can be computed as

$$\dot{q}_{rad,K} = j_K - g_K \tag{12.2.7}$$

If the net heat flow is positive, it represents an incoming heat term, and if the net heat flow is negative, it represents an outgoing heat term.

Considering the total energy conservation, the total heat flux must be 0.

$$\sum_{K=1}^{N} q_K A_K = 0 \tag{12.2.8}$$

Two spectral ranges are consider in engineering: the thermal band, and the solar band. Moreover, the black body emission $\dot{q}_{b,K}$ in radiosity equation is considered to be $\sigma_B T_K^4$ for the thermal band, and zero for the solar band.

Part VII.

Case of study

13. Free convective heat transfer in inclined cavities

13.1. Introduction

Once the software has been verified from the benchmarking cases that can solve properly the Navier-Stokes equations, it is appropriate to use the program for an engineering purpose. For the current bachelor's thesis, the study of inclined solar collectors has been chosen, in which a thermal fluid has to be heated by solar radiation.

Figure 13.1.1.: Inclined solar collector

On the left side of the solar collector, there is the thermal fluid circuit, which is warmer than the environment, and on the other side, there would be the environment where the collector is placed. Between both sides, there is a hermetic cavity, where it is preferable to have the vacuum in order to avoid the convective heat transfer, although it is difficult and expensive to achieve. For that reason, a certain fluid exists between these faces, and the only way to reduce the convective phenomenon is to impose different intermediate sheets or small cavities. Besides of convective transfer, the radiative effect must be taken into account.

For the case of study, the radiative effect has not been considered and only the convective heat transfer between boundaries is studied, in the case when the left boundary is imposed to be at a hot temperature, and the right boundary to a cold temperature. The top and bottom boundaries are considered to be adiabatic. For the study, it is considered to have an infinite, thus being only necessary to solve the 2D case.

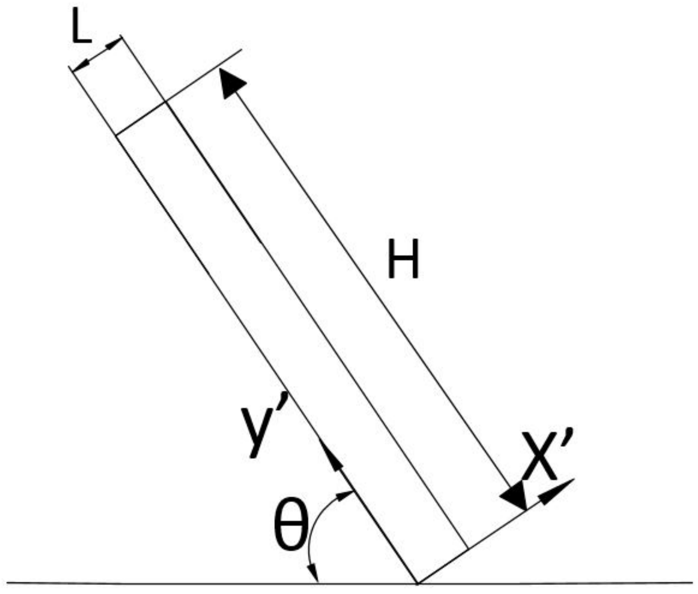


Figure 13.1.2.: Scheme of the inclined solar collector

The study of the previous case is about to be undertaken with the dimensionless equations for incompressible flows of Newtonian fluids for different Rayleigh values, a Prandtl number of 0.71 and different θ values using the fractional step method. Therefore, it must satisfy mass conservation equation with constant properties, momentum conservation equation with Boussinesq approximation for density, and energy conservation equation. The unknown variables are the velocity vector at the interface nodes, the pressure and the temperature at each node.

13.2. Boundary conditions

The boundary conditions for velocity are defined below

- Initial u *and* v value: $u = 0.0$ *and* $v = 0.0$.
- Left boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Right boundary condition: $u = 0.0$ *and* $v = 0.0$.
- Bottom boundary condition: $u = 0.0$ *and* $v = 0.0$.

- Top boundary condition: $u = 0.0$ *and* $v = 0.0$.

The boundary conditions for predictor velocity are defined below

- Initial u^p *and* v^p value: $u^p = 0.0$ *and* $v^p = 0.0$.
- Left boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Right boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Bottom boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.
- Top boundary condition: $u^p = 0.0$ *and* $v^p = 0.0$.

The boundary conditions for pseudo-pressure are defined below

- Initial $\tilde{p}$ value: $\tilde{p} = 1.2$.
- Left boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial x = 0$).
- Right boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial x = 0$).
- Bottom boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial y = 0$).
- Top boundary condition: zero normal gradient pressure ($\partial\tilde{p}/\partial y = 0$).

The boundary conditions for temperature are defined below

- Initial T value: $T = 0.5$.
- Left boundary condition: $T = 1.0$.
- Right boundary condition: $T = 0.0$.
- Bottom boundary condition: zero normal gradient temperature ($\partial T/\partial y = 0$).
- Top boundary condition: zero normal gradient temperature ($\partial T/\partial y = 0$).

13.3. Spatial discretization

As it has been shown in Figure 13.1.2, the geometry of the case consists of a high aspect ratio rectangle. In this case, the aspect ratio is equal to 50. That is to say the dimension y' is 50 times greater than the dimension x'. For that reason, a higher number of control volumes is needed in y' direction.

Otherwise, the convective phenomenon should be taken into account at boundaries. In order to achieve that objective, a non-uniform mesh has been applied to concentrate more control volumes near the boundaries. The expressions used to determine the coordinates of the nodes are shown below:

$$x_j = \frac{L}{2}\left(1 + \frac{\tanh\left\{\gamma_x\left(\frac{2(j-1)}{N_x} - 1\right)\right\}}{\tanh \gamma_x}\right) \tag{13.3.1}$$

$$y_i = \frac{H}{2}\left(1 + \frac{\tanh\left\{\gamma_y\left(\frac{2(i-1)}{N_y} - 1\right)\right\}}{\tanh \gamma_y}\right) \tag{13.3.2}$$

where L is the horizontal dimension, H is the vertical dimension, N_x is the number of horizontal nodes, N_y is the number of vertical nodes, γ_x and γ_y are the parameters to determine how the mesh changes along each axis. For uniform meshes, this value will be close to 0, whereas for non-uniform meshes will be greater than 0, thus densifying the mesh near the boundaries the higher the value is.

In this case, low values of γ have been used due to the fact that the flow easily becomes turbulent, so a uniform mesh is preferred to observe accurately the turbulent phenomenon in the centre of the rectangle, but a higher concentration of nodes near the boundaries is necessary to obtain reliable values of velocity and temperature gradients.

The spatial discretization is shown in Table 13.3.1.

Spatial property	L	H	N_x	N_y	γ_x	γ_y
Value	1	50	50	600	1.2	1.2

Table 13.3.1.: Spatial discretization for inclined solar collector

The mesh grid should be much finer for a DNS simulation, although it cannot be possible for computational reasons. Therefore, the mesh has been simplified expecting similar results. However, a LES simulations would be preferable to perform. But, its study could only be performed in a DNS simulation, because a LES simulation has not been studied to 2D cases.

A mesh study has been carried out to determine the optimum mesh concentration parameters. It has been computed for the same angle in order to vary the γ_x and γ_y parameters, as well as the number of control volumes in each direction. The optimal selected parameters are shown in Table 13.3.1. However, these parameters may depend on the angle and on the Rayleigh number. Nevertheless, as it has been mentioned before, the number of nodes is too low to obtain an accurate solution for the treated case.

13.4. The governing equations

The problem will be solved for the next Rayleigh numbers: $2.47 \cdot 10^4$ and $2.47 \cdot 10^5$, and a 0.71 Prandtl number. The case will be solved for a range of angles ϑ: 0º, 10º, 20º, 30º, 40º, 50º, 60º, 70º, 80º and 90º. The program uses the same equations shown in section 9.4. However, the dimensionless governing equations for incompressible flows of Newtonian fluids have to take into account the tilt angle:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = \frac{Pr}{Ra^{1/2}} \triangle \vec{u} - \nabla p + \vec{f} \tag{13.4.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{13.4.2}$$

$$\frac{\partial T}{\partial t} + (\vec{u} \cdot \nabla) T = \frac{1}{Ra^{1/2}} \nabla T + S \tag{13.4.3}$$

where $\vec{u}$ is the velocity vector, p is the pressure, Pr is the non-dimensional Prandtl number, Ra is the non-dimensional Rayleigh number, $\vec{f}$ is the body force vector which is equal to $(Pr\,T\ \cos(\vartheta), Pr\,T\ \sin(\vartheta))$, T is the non-dimensional temperature, ϑ is the tilt angle and S is the non-dimensional internal heat generation.

Quantities t, x, y, u, v, p and T are non-dimensional forms of their corresponding quantities τ, X, Y, U, V, P and θ, normalized with L dimension :

$$x = \frac{X}{L};\ y = \frac{Y}{L};\ u = \frac{UL}{\kappa Ra^{1/2}};\ v = \frac{VL}{\kappa Ra^{1/2}};\ T = \frac{\theta - \theta_C}{\theta_H - \theta_C};\ p = \frac{PL^2}{\rho \kappa^2 Ra};\ t = \frac{\tau \kappa Ra^{1/2}}{L^2}$$

$\vec{R}(\vec{u})$ from the Helmholtz-Hodge decomposition theorem can be defined as

$$\vec{R}(\vec{u}) = -(\vec{u} \cdot \nabla)\vec{u} + \frac{Pr}{Ra^{1/2}} \triangle \vec{u} + \vec{f} \tag{13.4.4}$$

The time discretzation is done in the same manner as section 7.2. Nonetheless, the spatial discretization needs some indications. The first one, Equation 7.3.2 needs an extra explanation

$$\begin{aligned}\int_V \vec{R}(\vec{u}^{\,n})\, dV &= -\int_V (\vec{u} \cdot \nabla)\vec{u}\, dV + \frac{Pr}{Ra^{1/2}} \int_V \triangle \vec{u}\, dV + \int_V \vec{f}\, dV \\ &= -\int_S (\vec{u})\,\vec{u} \cdot \vec{n}\, dS + \frac{Pr}{Ra^{1/2}} \int_S \nabla \vec{u} \cdot \vec{n}\, dS + \int_V \vec{f}\, dV\end{aligned} \tag{13.4.5}$$

Thus, for each direction

$$\int_V R\left(u^n\right) dV = -\left[(u)_e u_e S_e - (u)_w u_w S_w + (v)_n u_n S_n - (v)_s u_s S_s\right] +$$
$$\frac{Pr}{Ra^{1/2}}\left[\frac{u_E - u_P}{\delta x_e} S_e - \frac{u_P - u_W}{\delta x_w} S_w + \frac{u_N - u_P}{\delta y_n} S_n - \frac{u_P - u_S}{\delta y_s} S_s\right] + Pr\, T \cos(\vartheta)\, V_P \tag{13.4.6}$$

$$\int_V R\left(v^n\right) dV = -\left[(u)_e v_e S_e - (u)_w v_w S_w + (v)_n v_n S_n - (v)_s v_s S_s\right] +$$
$$\frac{Pr}{Ra^{1/2}}\left[\frac{v_E - v_P}{\delta x_e} S_e - \frac{v_P - v_W}{\delta x_w} S_w + \frac{v_N - v_P}{\delta y_n} S_n - \frac{v_P - v_S}{\delta y_s} S_s\right] + Pr\, T \sin(\vartheta)\, V_P \tag{13.4.7}$$

where the velocity of the interface nodes is calculated as it has been explained before. Nevertheless, the temperature term needs some considerations. The temperature field is obtained at the central nodes (see Figure 9.4.1), so an approximation is needed to obtain the temperature value for the staggered grid in y-direction and x-direction.

$$T_{staggered} = CDS\left(T_N, T_S\right) \tag{13.4.8}$$
$$T_{staggered} = CDS\left(T_E, T_W\right) \tag{13.4.9}$$

With the exception of the changes detailed before, the Fractional step method explained in section 9.4 can be applied without any problem. But this time, a conjugate gradient solver is used in order to solve faster the pressure Poisson equation. On the other hand, a biconjugate gradient stabilized solver is used in order to solve the convection-diffusion system.

Therefore, the program used to solve the case of study is the same one as the implemented for the differentially heated cavity, but this case also requires the angle ϑ, as well as the mesh concentration parameters for the creation of the non-uniform mesh.

13.5. The results

As it has been explained before, the most important parameter is the heat flow through any line parallel to y-axis, which is given by

$$Nu_x = \frac{1}{L_y}\int_0^1 Q(x,y)\,dy \tag{13.5.1}$$

where Q is defined as $Q = uT - \frac{\partial T}{\partial x}$. It must be constant in each line parallel to the y-axis. For that reason, it is only necessary to compute the Nusselt value in the left boundary where the horizontal component of the velocity is null.

The Nusselt number will be compared with some experimental correlations defined by Hollands[18] for inclined air layers considering the radiative heat transfer. For that reason, the results are not expected to be the same ones that they exposed. Equation 13.5.2 can be applied for aspect ratios greater than 40, Ra lower than 10^5 and θ lower than 60º. Nevertheless, for higher angles, a linear interpolation (Equation 13.5.3) between Nu(60º) from Hollands approximation and Nu(90º) from Shewen approximation (Equation 13.5.4) is done. The expressions are shown below:

$$Nu = 1 + 1.44\left[1 - \frac{1708}{Ra\cos(\theta)}\right]^{\cdot}\left(1 - \frac{1708\,(\sin(1.8\theta))^{1.6}}{Ra\cos(\theta)}\right) + \left[\left(\frac{Ra\cos(\theta)}{5830}\right)^{1/3} - 1\right]^{\cdot} \tag{13.5.2}$$

where$[x]^{\cdot} = max\,(0\,,\,x)$.

$$Nu = \frac{90^{º} - \theta}{30^{º}} Nu_{60^{º}} + \frac{\theta - 60^{º}}{30^{º}} Nu_{60^{º}} \tag{13.5.3}$$

$$Nu_{90^{º}} = \left[1 + \left[\frac{0.066Ra^{1/3}}{1 + (900/Ra)^{1.4}}\right]^2\right]^{1/2} \tag{13.5.4}$$

All the results can be found in Annex C.

According to Hollands [18], the movement of the flow can be split into three different groups based on the Rayleigh number.

- The first one is called the Conductive Regime, which includes a Ra range from 0 to $1708/\cos\theta$.
- The second one is called the Intermediate-Postconductive Regime, which includes a Ra range from $1708/\cos\theta$ to $10^4/\cos\theta$.
- The last one is called the High Rayleigh Number Regime which includes higher Ra numbers.

13.5.1. The Conductive Regime

On the first range, the fluid motion consists in a parallel flow to the boundaries, rising near the hot surface and falling near the cold surface. The heat transfer in this regime is consequently purely conductive, so the Nusselt number will be 1 in any line parallel to y' axis, except in the surroundings of the upper and lower surfaces, where a convective term exists.

The fluid motion for Ra=1708 and $\vartheta = 60^{\circ}$ is shown below:

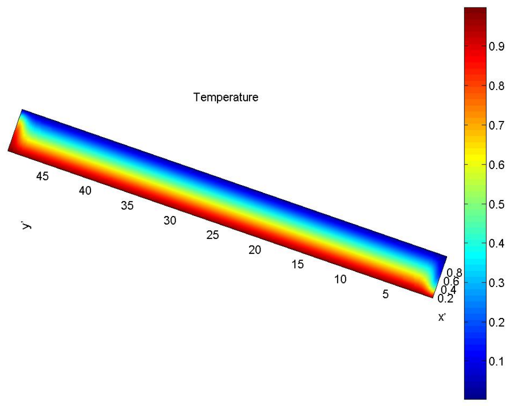


Figure 13.5.1.: Temperature for Ra=1708 and $\vartheta = 60^{\circ}$

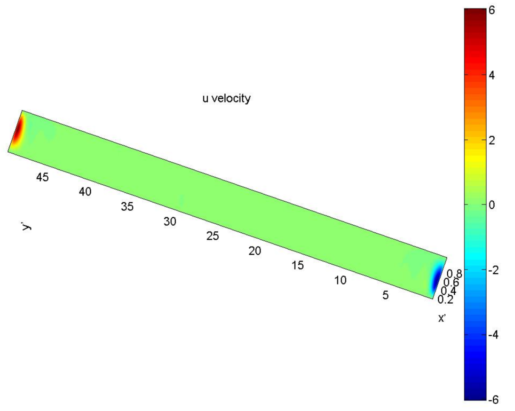


Figure 13.5.2.: U-velocity for Ra=1708 and $\vartheta = 60^{\circ}$

As it can be observed on the previous images (Figure 13.5.1 to Figure 13.5.6), the fluid has a purely conductive behaviour. For that reason, the temperature distribution along x'-axis is a linear distribution between 1 on the left side and 0 on the right side. Except on the top and the bottom surfaces where the horizontal velocity are different than 0, so the convective term is not equal to 0 and the temperature

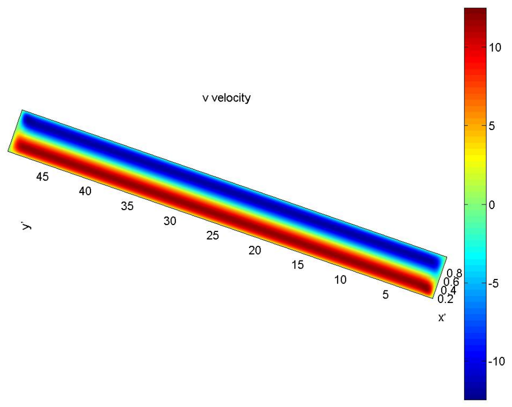


Figure 13.5.3.: V-velocity for Ra=1708 and $\vartheta = 60^{\circ}$

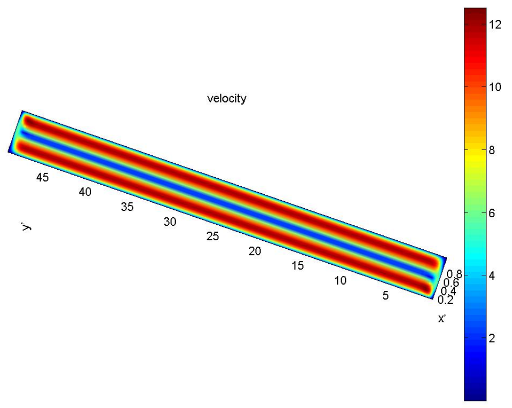


Figure 13.5.4.: Velocity for Ra=1708 and $\vartheta = 60^{\circ}$

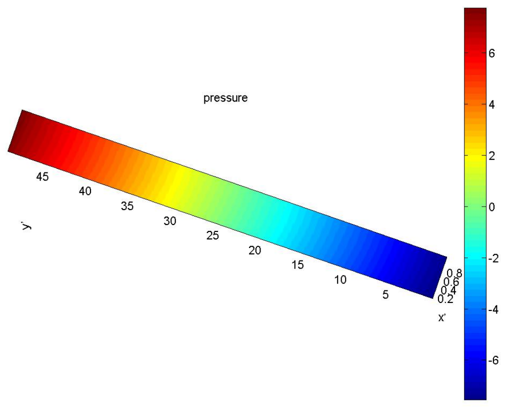


Figure 13.5.5.: Pressure for Ra=1708 and $\vartheta = 60^{\circ}$

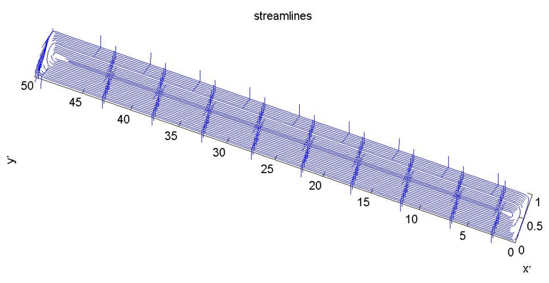


Figure 13.5.6.: Streamlines for Ra=1708 and $\vartheta = 60^{\circ}$

distribution changes.

The streamline graphic shows a laminar fluid flow, which is parallel to the boundaries. There is not any convective roll.

13.5.2. The Intermediate-Postconductive Regime

On the second range, the fluid motion becomes slightly unstable, advection takes over conduction as the dominant mode and a circulation patter engulfs all the enclosure, which are called Rayleigh-Bernand Cells. For higher values, the flow becomes turbulent, although these unsteady rolls persist. As the flow turns turbulent, the program computes the solution according to a DNS simulation.

The fluid motion for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$ is shown below:

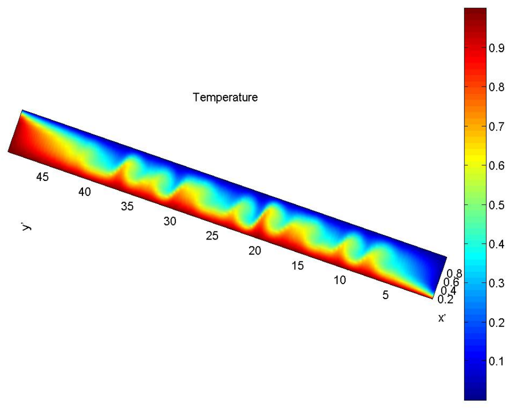


Figure 13.5.7.: Temperature for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

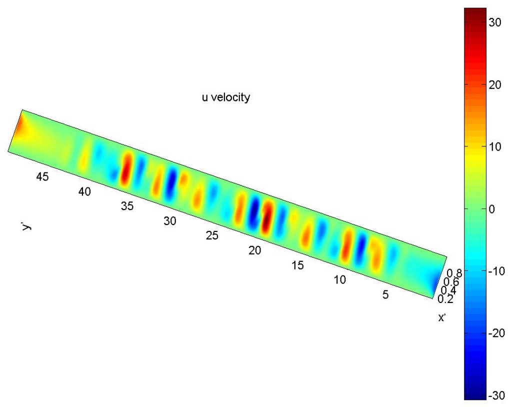


Figure 13.5.8.: U-velocity for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

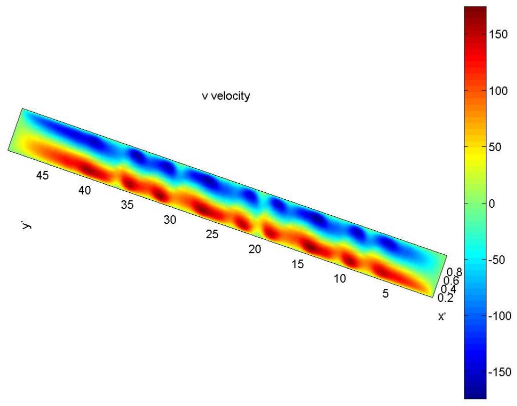


Figure 13.5.9.: V-velocity for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

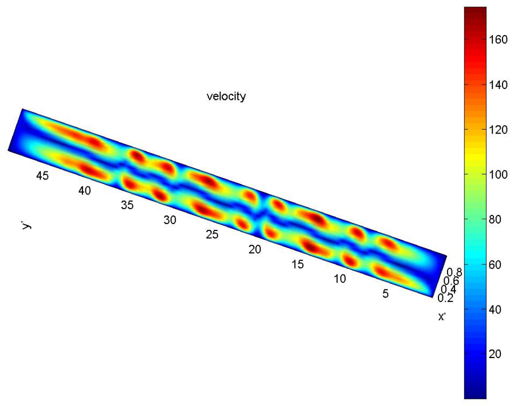


Figure 13.5.10.: Velocity for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

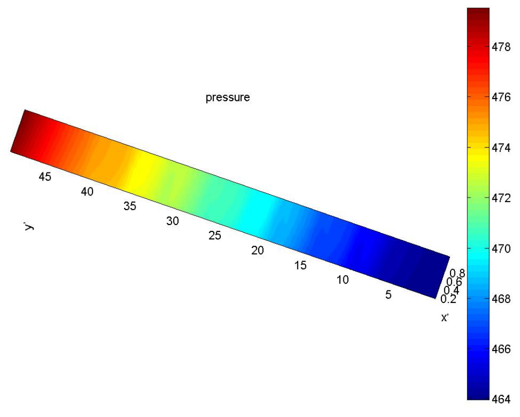


Figure 13.5.11.: Pressure for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

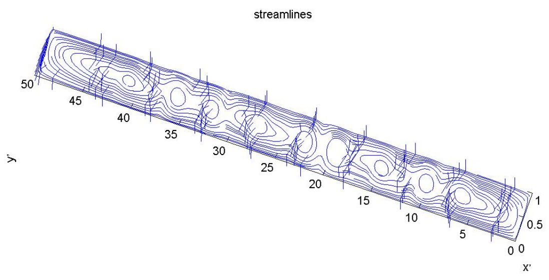


Figure 13.5.12.: Streamlines for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

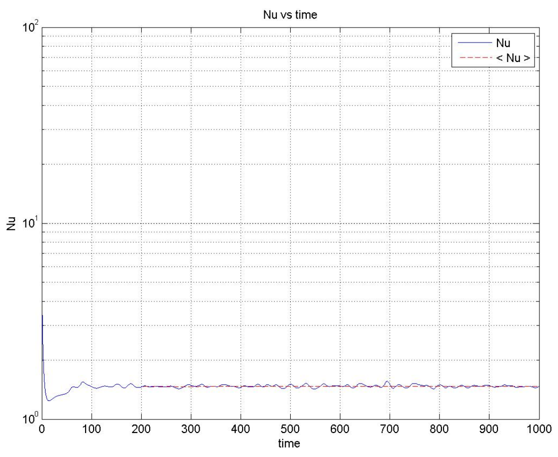


Figure 13.5.13.: Nusselt number for Ra=$2.47 \cdot 10^4$ and $\vartheta = 60^{\circ}$

As it can be observed on the previous images (Figure 13.5.7 to Figure 13.5.12), the fluid motion consists of many convective rolls which increase the Nusselt number with reference to the conductive regime. These convective rolls are due to the effect of density variation as a function of temperature. The flow rises and accelerates along x'-axis as it can be seen in Figure 13.5.8. When the fluid collides with the right boundary, it gets colder and it descends. This effect creates pairs of regions where the fluid ascends and then it descends.

The streamline graphic shows a convective fluid flow with small cells along y' axis. There are several convective rolls (circles) centred along the x=0.5 line.

The Nusselt number versus the tilt angle is shown below:

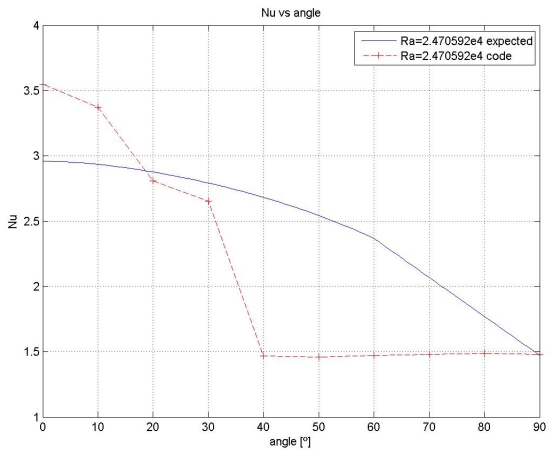


Figure 13.5.14.: Nusselt number with the tilt angle for Intermediate-Postconductive regime

Numerical values can be seen in Table 13.5.1.

Tilt angle	Code value	Expected value	Relative error
0	3,5496	2,9587	19,9713%
10	3,3719	2,9346	14,9020%
20	2,8070	2,8772	-2,4379%
30	2,6546	2,7922	-4,9264%
40	1,4684	2,6816	-45,2411%
50	1,4605	2,5435	-42,5772%
60	1,4730	2,3670	-37,7681%
70	1,4796	2,0690	-28,4904%
80	1,4873	1,7711	-16,0224%
90	1,4798	1,4732	0,4483%

Table 13.5.1.: Nusselt number comparison between code values and Hollands correlation for Intermediate-Postconductive regime

Figure 13.5.14 shows how the Nusselt number changes with the tilt angle. In that case, it decreases with the increase of tilt angle as expected. However, there is a sudden dip for tilt angle equals 30º-40º. Consequently, there is a considerable error between the value determined by Hollands [18] and the value computed by the program. This divergence will be explained in section 13.6.

13.5.3. The High Rayleigh Number Regime

On the third range, the fluid becomes stratified due to the fact that two boundary layers appear, one on each bounding surface which will define the resistance to heat transfer through the collector. On the other hand, the enclosure's core becomes stagnant although additional cells can develop in the corners. [17]

The fluid motion for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60º$ is shown below:

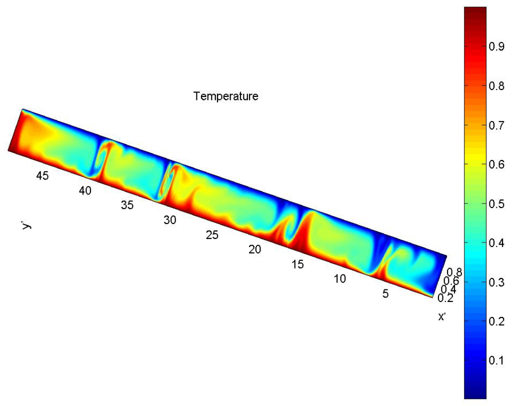


Figure 13.5.15.: Temperature for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60º$

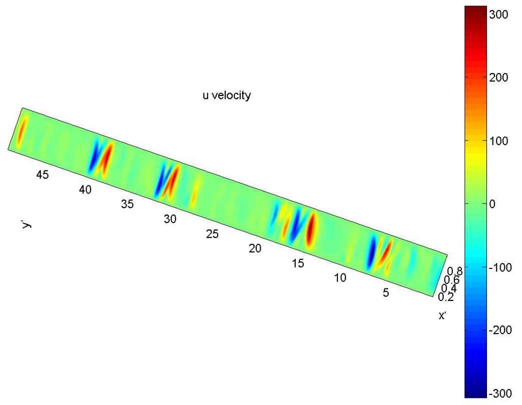


Figure 13.5.16.: U-velocity for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60º$

As it can be observed on the previous images (Figure 13.5.15 to Figure 13.5.20),

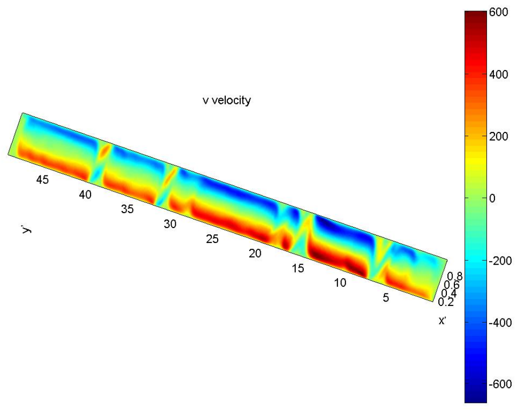


Figure 13.5.17.: V-velocity for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60^{\circ}$

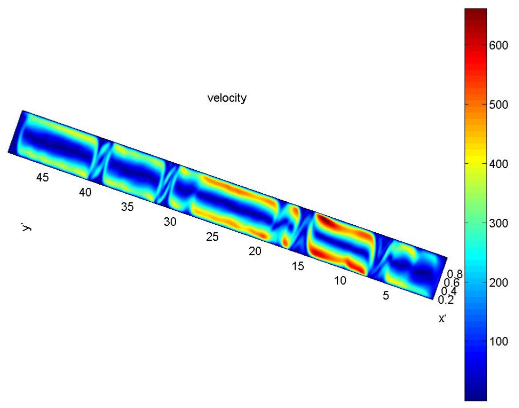


Figure 13.5.18.: Velocity for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60^{\circ}$

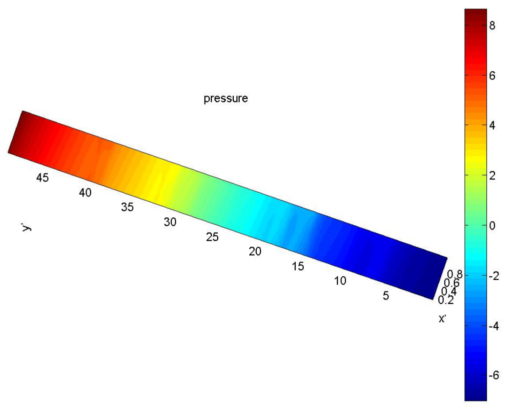


Figure 13.5.19.: Pressure for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60^{\circ}$

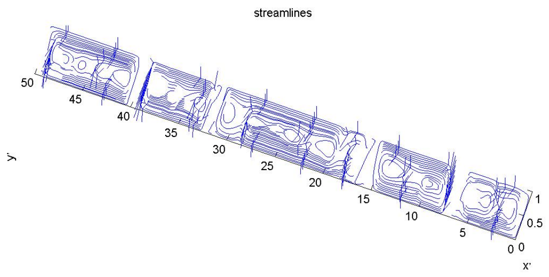


Figure 13.5.20.: Streamlines for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60^{\text{o}}$

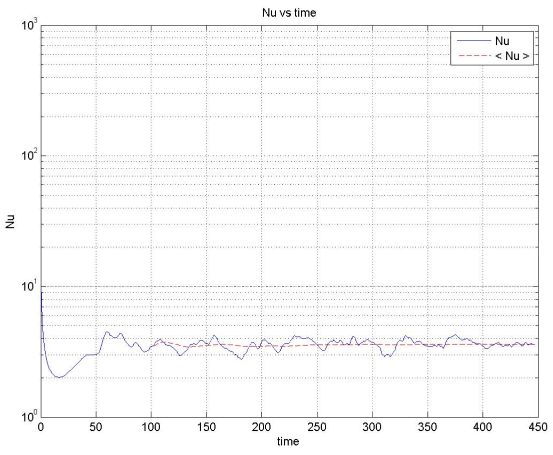


Figure 13.5.21.: Nusselt number for Ra=$2.47 \cdot 10^5$ and $\vartheta = 60^{\text{o}}$

the fluid motion consists of few convective rolls which increase the Nusselt number with reference to the conductive regime. The streamline graphic shows a convective fluid flow with big cells along y' axis. The core of the enclosure is virtually stagnant because the norm of the velocity is almost 0.

The Nusselt number versus the tilt angle is shown below:

Numerical values can be seen in Table 13.5.2.

Figure 13.5.22 shows how the Nusselt number changes with the tilt angle. In that case, it decreases with the increase of tilt angle as expected. However, there is a sudden dip for tilt angle equals 40º as it happens in intermediate-postconductive regime. Consequently, there is a considerable error between the value determined by Hollands [18] and the value computed by the program. This divergence will be explained in section 13.6.

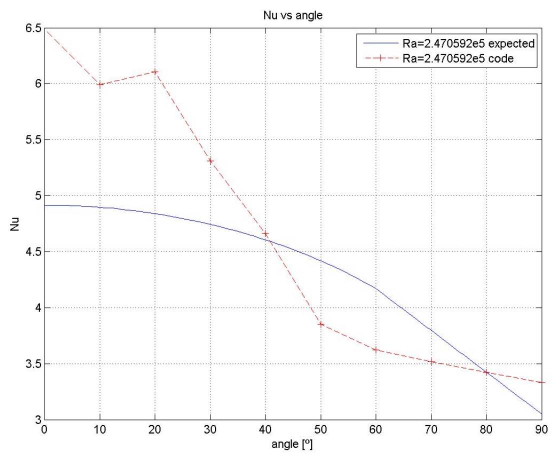


Figure 13.5.22.: Nusselt number with the tilt angle for high Rayleigh number regime

Tilt angle	Code value	Expected value	Relative error
0	6,4835	4,9164	31,8739%
10	5,9939	4,8970	22,3996%
20	6,1048	4,8398	26,1391%
30	5,3099	4,7436	11,9393%
40	4,6628	4,6052	1,2509%
50	3,8513	4,4180	-12,8276%
60	3,6210	4,1691	-13,1482%
70	3,5173	3,7963	-7,3499%
80	3,4243	3,4235	0,0227%
90	3,3294	3,0508	9,1332%

Table 13.5.2.: Nusselt number comparison between code values and Hollands correlation for high Rayleigh number regime

13.6. Conclusions

Focusing on the angle, when the enclosure is horizontal or tilted from the horizontal with small angle, convective rolls form. However, when the tilt angle is increased, the numerous rolls develop on a single cell at the critical tilt angle which depends on the Rayleigh number and the aspect ratio.[28]

For that reason, the maximum heat transfer occurs at the smallest angles due to the fact that the convective rolls are aligned with x direction. When the tilt angle is increased, the convective rolls rotates its main direction in order to align with vertical axis, so the Nusselt number along any perpendicular line of x' axis decreases.[28]

Nevertheless, the minimum heat transfer value is associated with the tilt angle where the flow changes its motion from multiple Rayleigh-Bernand cells to a single cell. This angle, which is called the critical angle, is dependent on the Rayleigh number and the aspect ratio as well as the Prandtl number. The critical angle gets closer to 90º when the aspect ratio increases.[17]

After running the program for the different tilt angles and Rayleigh numbers, a time analysis of the Nusselt value has been done in order to determine it when the case has reached a steady or steadily oscillating mode. This steady mode of the flow is obtained after a certain time depending on the tilt angle and the Rayleigh number. Thus, as it can be noticed in Figure 13.5.13 and Figure 13.5.21, a first transitional region and then a second steadily oscillating stage are observed. Nonetheless, the second region is only taken into account to determine the mean Nusselt number.

Only two cases have been computed because of its high computational time requirements. Each tilt angle simulation has taken about 30 hours. Thus, there is not enough information to verify how well the program simulates the fluid motion inside a high aspect ratio cavity.

From the previous studies, a smooth decrease in the Nusselt number with the increase angle θ can be expected (Equation 13.5.2). Nonetheless, the results in Annex C indicate 2D CFD results deviating from the true behaviour defined by Hollands et. al. [18]. In all two cases, a sudden dip can be observed instead of a gradual decrement in the Nusselt number, which is characteristic of 2D CFD analysis. Therefore, it does not mean that the program computes the parameters inside the cavity in a wrong way, but that these erroneous results emphasize the need to treat the case as a 3D case.

To conclude, it is important to insist on 3D characteristic motion of the fluid, which should have been considered from the beginning. However, the purpose of this study is to have a first contact with numerical methods. Thereupon, a 3D turbulent simulation goes further than the initial target.

Part VIII.

Economic and environmental study

14. Economic and environmental study

14.1. Economics

The main target of this study is the creation of a CFD&HT software that is able to solve a large number of laminar cases, as it has been mentioned above. It is for this reason that no economic profit is expected to be made by selling the product due to the fact that it would be for own use or for a commercial use.

This study has been carried out by a single engineer along a 4 months semester (it has been completed in exactly 19 weeks), working an average of 7 days per week, 4 hours per day. Assuming the average wage of an undergraduate engineer is 15€/hour, the overall human cost can be estimated in Table 14.1.1.

Weeks/Semester	days/week	hours/day	Total hours [h]	Price [€/h]	Cost [€]
19	7	4	532	15	7980

Table 14.1.1.: Human cost estimation

The hardware used to perform the project was a laptop with a cost of 780€. Considering a 25% annual amortization and a period of 19 weeks of 52weeks/year, the hardware cost will be computed as $780 \cdot 0.25 \cdot 19/52$, which is equal to 71.25€. Although extra calculations have been done using a desktop computer with a cost of 650€. Considering a 25% annual amortization and a period of 5 weeks of 52weeks/year, the hardware cost will be computed as $650 \cdot 0.25 \cdot 5/52$, which is equal to 15.625€.

$$Hardware\ cost = 71.25€ + 15.625€ = 86.875€ \tag{14.1.1}$$

Several software have been used, including Microsoft Office package (80€), Matlab with student license (69€), LyX Latex (free software), Mendeley (free account) and Dev-C++ compiler (free software).

$$Software\ cost = 80€ + 69€ = 149€ \tag{14.1.2}$$

Apart from the hardware and software, an electricity cost of 0.15€/kWh has to be included. Taking into account an electrical power of 120W for the personal computer and an electrical power of 500W for the desktop computer, the study has consumed 483.84 kWh. The total electrical cost is shown in Equation 14.1.3.

$$\textit{Electrical cost} = 0.15€/kWh \cdot 483.84\,kWh = 72.576€ \tag{14.1.3}$$

A 20% overhead cost has been included.

The total cost is shown in Table 14.1.2.

Type cost	Units	Criteria	Cost (€)
Human resources cost	532 h	15€/h	7980
Hardware cost	19 weeks	780€ in 4 years	71.25
	5 weeks	650€ in 4years	15.63
Software cost	-	-	149
Electrical cost	483.84kWh	0.15€/kWh	72.58
Cost			8288.45
Overhead	-	20%	1657.69
Total Cost			9946.14

Table 14.1.2.: Total cost estimation

14.2. Environmental impact

The environmental impact of this study is basically related to the study itself and it is almost null. However, the electricity consumed by the computer has generated CO_2 emissions. According to the budget, the study has lasted 532 hours, even though it has needed some extra computational calculations, which have been completed in approximately 840 hours. During this period, a personal computer with a mean consumption of 120W has been used, while the extra calculations have been performed on a desktop computer with a mean consumption of 500W. Therefore, this study has consumed 483.84 kWh. According to the Spanish ministry of energy [1], each kWh produces approximately 0.29 kg CO_2 . Thus, this study has produced a 140.314 kg of CO_2 .

Also, it has to be remarked that a CFD software consumes much less electricity than a real wind tunnel test. The computer program does not use extra materials to test the prototype, which can have a negative effect on the environment. It also requires less personnel to perform the simulation, so it means that the human cost decreases.

Moreover, future researches of this study might suppose an improvement in the efficiency of thermal equipment, specifically in solar collectors. Thus, it would suppose a reduction of the necessary area to obtain the same flow rate of the warm fluid at the same temperature. In turn, this would reduce the amount of materials used to manufacture these solar collectors.

This also might correspond to a decrement in the cost of solar collectors, which in turn, it would mean an increase in the purchase and installation of this equipment by consumers.

Part IX.

Conclusions and future actions

15. Conclusions and future actions

15.1. Conclusions

The aim of the study is considered to have been achieved since an extensive study of how a laminar fluid behaves has been done. The Navier-Stokes equations have been solved for different flows inside cavities. Although, it is necessary to emphasize that the case of study could not be finished successfully because of its turbulent nature.

The following sections were necessary to be developed in order to carry out this study:

- Study and understanding of the differential equations which govern the fluid motion.
- Transformation of differential equations into algebraic equations by means of different numerical methods.
- Continuous development of the code from the resolution of benchmarking cases:
  - Introduction to the fluid dynamics phenomenology problems thanks to convection-diffusion problems as the Smith-Hutton problem.
  - Introduction and study of convection phenomenon using Driven-cavity case.
  - Introduction and study of the heat transfer phenomenology through the Differentially heated cavity case.
- Code verification and implementation of new features to reduce computational costs.
- Develop a case for engineering purpose in the thermal field. The selected case is an inclined solar collector for the purpose of reducing energy losses by heat transfer.

The code developed along this study that has been used to obtain the final results can be found in Annex D.

The project has resulted for the author as an introduction in the field of heat transfer (CFD&HT), and an understanding of the mathematical formulation and numerical solution of the equations. It has also been useful to have a first approach to the application and understanding of turbulence.

On the other hand, the realization of this study has been an opportunity to go into detail about the C++ programming language.

15.2. Future actions

As it is already stated in the justification and in the objective of this study, its functionality was to learn and go in depth into knowledge of fluid dynamics. In this context, it makes sense to discuss on the future lines of research if the project would go forward in a hypothetical case or if it was about to continue in the master's degree. Therefore, the actions to consider would be:

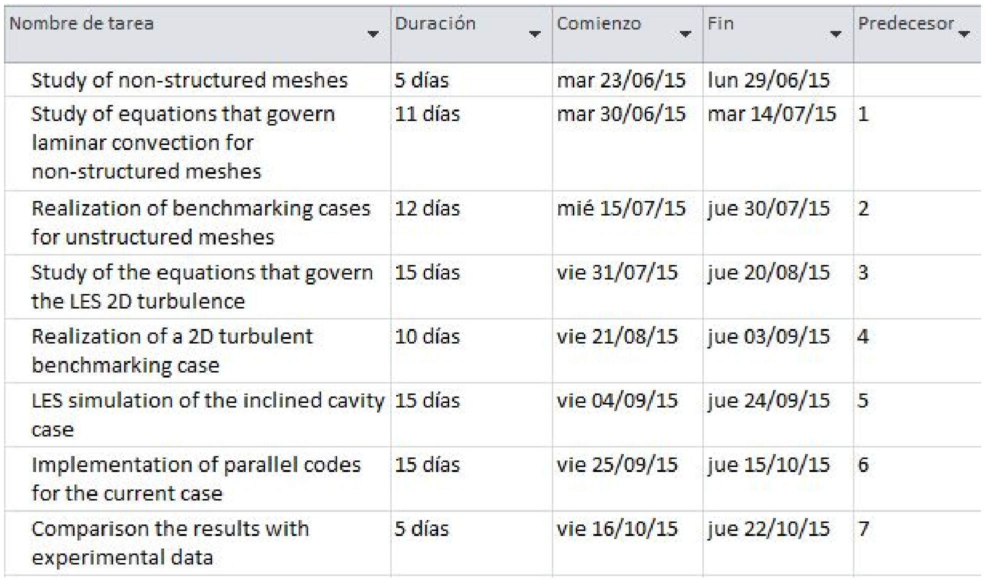
First, improve the skills in C++ object-oriented programming, with what one would be capable of developing more efficient codes applicable to more complex cases.

Secondly, learn how to implement parallel codes in C++, as well as the theory of resolution algorithms that can be solved in different computing threads.

In the third place, extrapolate the N-S equations, as well as the numerical resolution methods for unstructured meshes, thus allowing the resolution of much more complex geometries. Another possible improvement would be to solve the N-S equations using temperature dependent variables.

Finally, it would go in depth into the phenomenology that governs the turbulence in order to extrapolate the results obtained for the studied cases to turbulent flows using LES models.

The temporary line of the tasks that must be done in order to accomplish this study is detailed below.



Nombre de tarea	Duración	Comienzo	Fin	Predecesor
Study of non-structured meshes	5 días	mar 23/06/15	lun 29/06/15	
Study of equations that govern laminar convection for non-structured meshes	11 días	mar 30/06/15	mar 14/07/15	1
Realization of benchmarking cases for unstructured meshes	12 días	mié 15/07/15	jue 30/07/15	2
Study of the equations that govern the LES 2D turbulence	15 días	vie 31/07/15	jue 20/08/15	3
Realization of a 2D turbulent benchmarking case	10 días	vie 21/08/15	jue 03/09/15	4
LES simulation of the inclined cavity case	15 días	vie 04/09/15	jue 24/09/15	5
Implementation of parallel codes for the current case	15 días	vie 25/09/15	jue 15/10/15	6
Comparison the results with experimental data	5 días	vie 16/10/15	jue 22/10/15	7

Figure 15.2.1.: Temporary line of future tasks

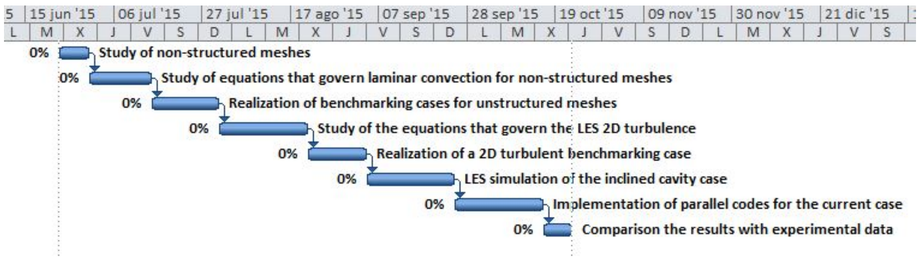


Figure 15.2.2.: Future Gantt chart

Part X.

Bibliography

Bibliography

[1] Factores de emisión de co_2 y coeficientes de paso a energía primaria de de diferentes fuentes de energía final consumidas en el sector edificios en españa. Tech. rep., Ministerio de industria, energía y turismo, May 2014.

[2] A. Cengel, Y., and M. Cimbala, J. *Fluid Mechanics: Fundamentals and applications.* McGraw-Hill, 2006.

[3] Brandt, A., and Yavneh, I. Inadequacy of first-order upwind difference schemes for some recirculating flows. *Journal of computational physics* (1991), 128–143.

[4] Colomer Rey, G. *Numerical methods for radiative heat transfer.* Phd thesis, ETSEIAT - UPC, Terrassa, June 2006.

[5] CTTC, D. Numerical solution of convection. Tech. rep., ETSEIAT, 2010.

[6] CTTC, D. Introduction to the fractional step method. Tech. rep., ETSEIAT, 2011.

[7] CTTC, D. Navier-stokes equations: symmetry and conservation. Tech. rep., ETSEIAT, June 2011.

[8] CTTC, D. Introduction to the dns of turbulent flows. Tech. rep., ETSEIAT, June 2012.

[9] CTTC, D. Burgers equation in fourier space. Tech. rep., ETSEIAT, November 2014.

[10] CTTC, D. Large-eddy simulation of turbulent incompressible flows. Tech. rep., ETSEIAT, December 2014.

[11] de Vahl Davis, G. Natural convection of air in a square cavity a bench mark numerical solution. *International journal for numerical methods in fluids 3* (1983), 249–264.

[12] de Vahl Davis, G., and Jones, I. P. Natural convection in a square cavity: a comparison exercise. *International journal for numerical methods in fluids 3* (1983), 227–248.

[13] Ferziger, J. H., and Peric, M. *Computational Methods for Fluid Dynamics*, 3rd edition ed. No. ISBN 3-540-42074-6. Springer-Verlag, 2002.

[14] Ghia, U., Ghia, K. N., and Shin, C. T. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of computational physics* (1982), 387 – 411.

[15] H. Lienhard IV, J., and H. Lienhard V, J. *A heat transfer textbook*, third ed. Phlogiston Press, Cambridge, Massachusetts, U.S.A., 2008.

[16] Heggelien Refsnaes, R. A brief introduction to the conjugate gradient method. 2009.

[17] Henderson, D., Junaidi, H., Muneer, T., Grassie, T., and Currie, J. Experimental and cfd investigation of an icsswh at various inclinations. *Renewable and Sustainable Energy Reviews*, 11 (2007), 1087–1116.

[18] Hollands, G. T., Unny, T. E., Raithby, G. D., and Konicek, L. Free convective heat transfer accors inclined air layers. *Journal of Heat Transfer*, 98(2) (May 1976), 189–193.

[19] Kizildag, D., Ventosa, J., Rodríguez, I., and Oliva, A. V european conference on computational fluid dynamics. In *Non-oberbeck-boussinesq natural convection in a tall differentially heated cavity* (2010).

[20] M. Elsherbiny, S. Free convection in inclined air layers heated from above. *Elsevier: International Communications in Heat and Mass Transfer 39*, 18 (1996), 3925–3930.

[21] Minkowycz, W. J., and Sparrow, E. M., Eds. *Series in Computational and Physical Processes in Mechanics and Thermal Sciences.* No. ISBN 978-1-4398-9440-8. McGraw-Hill, 2006.

[22] Omari, R. Cfd simulations of lid driven cavity flow at moderate reynolds number. *European Scientific Journal vol.9*, No.15 (May 2013).

[23] Online, C. Mesh classifiaction. webpage, May 2007.

[24] Pérez Segarra, C., Castro, J., Oliva, A., and Capdevila, R. Apunts de transferència de calor per radiació. April 2013.

[25] Siegel, R., and R. Howell, J. *Thermal radiation heat transfer*, vol. 3. NASA, 1971.

[26] Stroustrup, B. *The C++ programming language*, fourth ed. No. ISBN 978-0-321-56384-2. Addison-Wesley, 2013.

[27] Talens Oliag, S. Curso de programación en c++.

[28] Tari, I. Natural convection simulations and numerical determination of critical tilt angles for a parallel plate channel. *Energy Conversion and Management 51*, 4 (2010), 685–695.

[29] Trias, F. X., Gorobets, A., Soria, M., and Oliva, A. Direct numerical simulation of a differentially heated cavity of aspect ratio 4 with rayleigh numbers up to 10^{11} part i: Numerical methods and time-averaged flow. *International Journal of Heat and Mass Transfer* (2010), 665–673.

[30] Trias, F. X., Gorobets, A., Soria, M., and Oliva, A. Direct numerical simulation of a differentially heated cavity of aspect ratio 4 with rayleigh numbers up to 10^{11} part ii: Heat transfer and flow dynamics. *International Journal of Heat and Mass Transfer* (2010), 674–683.

[31] V. Patankar, S. *Numerical heat transfer and fluid flow.* No. ISBN 0-07-048740-5. McGraw-Hill, 1980.

[32] Versteeg, H. K., and Malalasekera, W. *An introduction to computational fluid dynamics: the finite volume method*, second ed. No. ISBN: 978-0-13-127498-3. Pearson Education Limited, 2007.

[33] Xhafa, F., Gatius Vila, M., Martín Prat, n., Esquerra Llucià, I., Valverde Ruiz, A., López-Herrera, J., Moral Romero, O., and Amirian Basiri, G. *Programació pràctica en C++.* Edicions UPC, 2010.

[34] Xu, F. Convective instability of a vertical thermal boundary layer in a differentially heated cavity. *International Communications in Heat and Mass Transfer* (2014), 8–14.

[35] Xu, F., Patterson, J., and Lei, C. 16th australasian fluid mechanics conference. In *Transient Natural Convection in a Differentially Heated Cavity with a Thin Fin of Different Lengths on a Sidewall* (2007).

[36] Y. Murthy, J. *Numerical Methods in Heat, Mass, and Momentum Transfer.* School of Mechanical Engineering, Purdue University, 2002.

[37] Yang, H. X., and Zhu, Z. Numerical study on transient laminar natural convection in an inclined parallel-walled channel. *Elsevier: International Communications in Heat and Mass Transfer 30*, 3 (2003), 359–367.