UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Universitat Politècnica de Catalunya. UPC

The School of Industrial, Aerospace and Audiovisual Engineering of Terrassa. ESEIAAT

Master's degree in Aerospace Engineering

Incompressible Navier-Stokes solver in Python

Document: Report

Author: Jordi Galí Gimeno

Director: Manel Soria Guerrero

Date: 22 June 2021

Call: Spring

[This page intentionally left blank]

1 Resum

L'ús de mètodes numèrics per resoldre problemes de dinàmica de fluids ha estat un camp d'investigació des dels anys cinquanta del segle passat. La necessitat de desenvolupar avions militars d'alt rendiment va produir el descobriment d'una eina molt útil pel sector de l'enginyeria. Des d'aleshores, l'ús de la mecànica de fluids és un dels principals temes a tractar al màster d'enginyeria aeroespacial. Amb to això, aquesta tesina es centra en desenvolupar un codi per motius educatius capaç de resoldre casos incompressibles amb un programari de codi obert. S'ha dit desenvolupament d'un codi educatiu perquè el principal objectiu és construir un procediment per validar cada funció implementada en el codi. Un cop les funcions principals queden validades, es desenvoluparà un codi per resoldre casos periòdics i casos on els límits es formen per parets. Queda clar, doncs, que el primer objectiu és validar les diferents funcions que formaran part del codi principal. Un cop es compleix el primer objectiu, el següent és desenvolupar un codi capaç de resoldre els casos comentats. Tant el cas periòdic com el de parets als límits tenen diferents mètodes per validar els resultats. Un cop això s'ha fet, es podrà dir que el solver estarà correctament desenvolupat. En aquest estudi s'ha utilitzat el Fractional Step Method amb el teorema de descomposició de Ladyzhenskaya, també conegut com el teorema de Helmholtz–Hodge. Per l'estudi s'utilitzarà una malla staggered, on les variables escalars com la pressió es trobaran al centre de les cel·les mentre que els vectors de velocitats es localitzen a les cares de la cel·la. Aquest mètode és molt útil per casos incompressibles per motius acadèmics per la seva senzillesa.

2 Abstract

The use of numerical methods to solve flow dynamics problems has been a major issue of study since the fifties of the last century. The need of developing high performance military planes produced the discovery of a very powerful tool for the engineering sector. Since then, the use on computational fluids dynamics is one of the fundamental topics of study in the Aerospace engineering master's degrees. With that, this thesis is focused on developing an educational code able to solve incompressible cases with an open source programming language. It is said that the code developed is for educational purposes because the main objective is to build a procedure to validate each function added at the code. Once the main functions are validated, a code will be build to solve periodic and wall bounded cases. It is clear then that the first objective is to validate all the different functions that will be part

of the main code. Once this objective is achieved, the second one is to solve a periodic case and a wall bounded case. Both studies must have a way of validating the results, when this is done, the solver will be developed correctly. The flow dynamics method used is the fractional step method that will need the Ladyzhenskaya decomposition theorem, also known as the Helmholtz–Hodge theorem. For the study a staggered mesh is used, this means that it contains the scalar variables at the center of the cells and the velocity vectors are located at the face of the cells. This method is very useful for incompressible cases with educational purposes because of its simplicity.

Contents

List of Figures

List of Tables

3 Introduction

3.1 Aim

The main objective of the project is the implementation of the Navier-Stokes equations for incompressible flow in a python code for educational purposes. With this, a secondary objective will be set in order to achieve the main one that is validating the main functions correctly. The method of manufactured solutions will be used to do so and three different codes will be developed in order to validate the Poisson solver, the time-step integration and the functions that evaluate the convective and diffusive terms. Once this is reached, the aim is to implement boundary conditions and validate them.

3.2 Scope

In order to reach the main objective of the project it will be necessary for the student to understand the Navier-Stokes equations and the mathematical theory to implement them in the code. Developing skills in numerical methods and understand and dominate the Python programming language. As explained, it will be necessary for the student to dominate the method of manufactured solutions in order to validate the different functions during the implementation of the equations.

3.3 Requirements

At the beginning of the project some requirements are stated and it must be said by the student if they have been fulfilled or not. If they are not well reached or not reached at all, the student must justify the reason why this has happened. The requirements are placed chronologically so that one is reached the next one can be developed.

The first requirement is understanding the Navier-Stokes equations and the different methods to implement them in the code. The following three requirements consist of the first validation part of the code. First, it will be necessary to implement and validate the diffusive and convective functions. Then, the Poisson equation will also be implemented and validated. And finally, the time integration function will be implemented and validated by the method of manufactured solutions. These three parts will be called part A, B and C respectively. Once this is reached, a periodic case can be solved by the code. The next requirement consist of implementing the boundary

conditions. The Dirichlet boundary conditions will be implemented in the code and with that, the driven cavity case can be studied.

3.4 Background

Navier-Stokes equations describe how the pressure, velocity, temperature and density are related in a moving fluid. For this reason, this equations are very important for the design in many engineering fields, but mostly in the aerospace. Since the fifties of the last century, this method has been improved a lot to give very accurate results for complex problems [6]. At this context, this study does not intend to improve the state of the art of the computational fluid dynamics but to make a educational and understandable method to solve incompressible cases.

3.5 Acknowledgments

A special thanks to the director Manel for the teaching and Marina for helping with the boundary conditions.

4 Theoretical introduction

4.1 General description of the problem

The case of study will consist of a incompressible flow (less than 0.3 Mach velocity), which means that the density is constant for the path of a fluid element. For the validation code, the Reynolds number used and the density will be 10 and 1 respectively. Then, once the part A, B and C are well implemented, some cases will be studied and the Reynolds number will be changed accordingly.

Navier-Stokes equations describe the behaviour of the flow and states the relation between its properties like pressure, velocity, density and temperature. The flow described must be Newtonian which means that the viscous stresses and the strain rate are proportional. In the next section, it will be explained the derivation of Navier-Stokes equations for incompressible cases.

In the theoretical introduction and in the code validation sections the lecture notes [7] are used. The discretization method will follow the same structure and they will be the main guide for the validation of the different functions of the code as is seen later.

4.2 Navier-Stokes equations

The Navier-Stokes equations are the mass conservation equation, the momentum equations and the energy equation. The latest won't be used in this study because it is not needed for incompressible flows.

The mass conservation equation is also called the continuity equation and states that the mass increment in a control volume has to be equal to the mass that leaves it minus the mass that enters it. It is expressed as [8]:

$$\frac{\partial \rho}{\partial t} + \rho \frac{\partial u_i}{\partial x_i} = 0 \tag{1}$$

And as explained, for incompressible flows the derivative of the density along time is zero so the mass conservation equation stays as:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2}$$

The momentum equations relates the sum of the forces acting on a fluid element to its acceleration. So, for a control volume and focusing on the

forces at the X direction the equation is stated as:

$$F_x = \left(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} \right) dxdydz \tag{3}$$

From the second Newton's law, the force can be expressed for an infinitesimal control volume as:

$$F_x = \rho \left(\frac{Du_x}{Dt} \right) dxdydz \tag{4}$$

Where the term:

$$\frac{Du_x}{Dt} = \frac{\partial u_x}{\partial t} + u_j \frac{\partial u_x}{\partial x_j} \tag{5}$$

Is the acceleration of the flow which is a substantial derivative. It is a tensor that depends on position and time so it will consist of a time derivative term and a convection term. The convection term is the reason why the equation is not linear [9]. As can be seen, the acceleration of the flow is related to the change of the velocity with the position.

Leaving the equation for the X direction as:

$$\rho \left(\frac{\partial u_x}{\partial t} + u_j \frac{\partial u_x}{\partial x_j} \right) = \left(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} \right) \tag{6}$$

The stress tensor must be expressed as function of pressure and velocity. This relation can be seen in the equations 7, 8, 9 and 10. The stress tensor is described in function of the pressure and the viscous stresses τ, which are described in function of the velocity.

$$\sigma_{ij} = \begin{bmatrix} -p + \tau_{xx} & \tau_{yx} & \tau_{zx} \\ \tau_{xy} & -p + \tau_{yy} & \tau_{zy} \\ \tau_{xz} & \tau_{yz} & -p + \tau_{zz} \end{bmatrix} \tag{7}$$

$$\tau_{xy} = \tau_{yx} = \mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \quad \tau_{xx} = 2\mu \frac{\partial u}{\partial x} \tag{8}$$

$$\tau_{yz} = \tau_{zy} = \mu \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \right) \quad \tau_{yy} = 2\mu \frac{\partial v}{\partial y} \tag{9}$$

$$\tau_{zx} = \tau_{xz} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \quad \tau_{zz} = 2\mu \frac{\partial w}{\partial z} \tag{10}$$

So taking equations 7, 8, 9 and 10, the momentum equation stays as follows:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \tag{11}$$

From this, it can be obtained the Euler equations and the equations of potential flow if the flow is considered non-viscous but they won't be used in this study.

The momentum equation must be commented in detail. In the following section it will be explained how each term is implemented in the code. Looking at the equation 11, it can be found the time derivative:

$$\frac{\partial u_i}{\partial t}$$

The advective (or convective) term:

$$u_j \frac{\partial u_i}{\partial x_j}$$

The pressure gradient:

$$-\frac{1}{\rho}\frac{\partial p}{\partial x_i}$$

And viscous (or diffusive) term:

$$\nu \frac{\partial^2 u_i}{\partial x_j \partial x_j}$$

From now on, the advective and viscous terms will be aforementioned as convective and diffusive respectively. Those two terms are the divergence of a vector so the divergence theorem must be used as will be explained later.

It will be also explained how to treat the source of non-linearity, the convective term. This term is called the material derivative, that consist of a physical quantity, in this case the velocity, that is linked to the variation of time and space of a macroscopic velocity field.

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial v} + w\frac{\partial \phi}{\partial z}$$

It can be seen that in this case, the velocity depends on itself, which means that is transporting itself.

4.3 Divergence theorem

As explained, some terms are the divergence of a vector. For those cases, the divergence theorem will be used. The divergence theorem is also called the Gauss's theorem or Ostrogradsky's theorem [10].

$$\int_V \nabla \cdot J \, dV = \int_{dV} J \, n \, dS$$

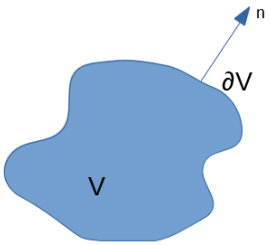

Figure 1: Divergence theorem

At the left site of the equation, it is described the imbalance of mass and at the right it is described the flux that enters and leaves through the close surface limit.

4.4 Mesh arrangement

In order to understand the discretization explained in the following sections, the mesh nomenclature must be very clear. As can be seen in the figure 2 this study is working with a staggered and centered mesh. The centered mesh will contain the pressure values and the staggered the velocities.

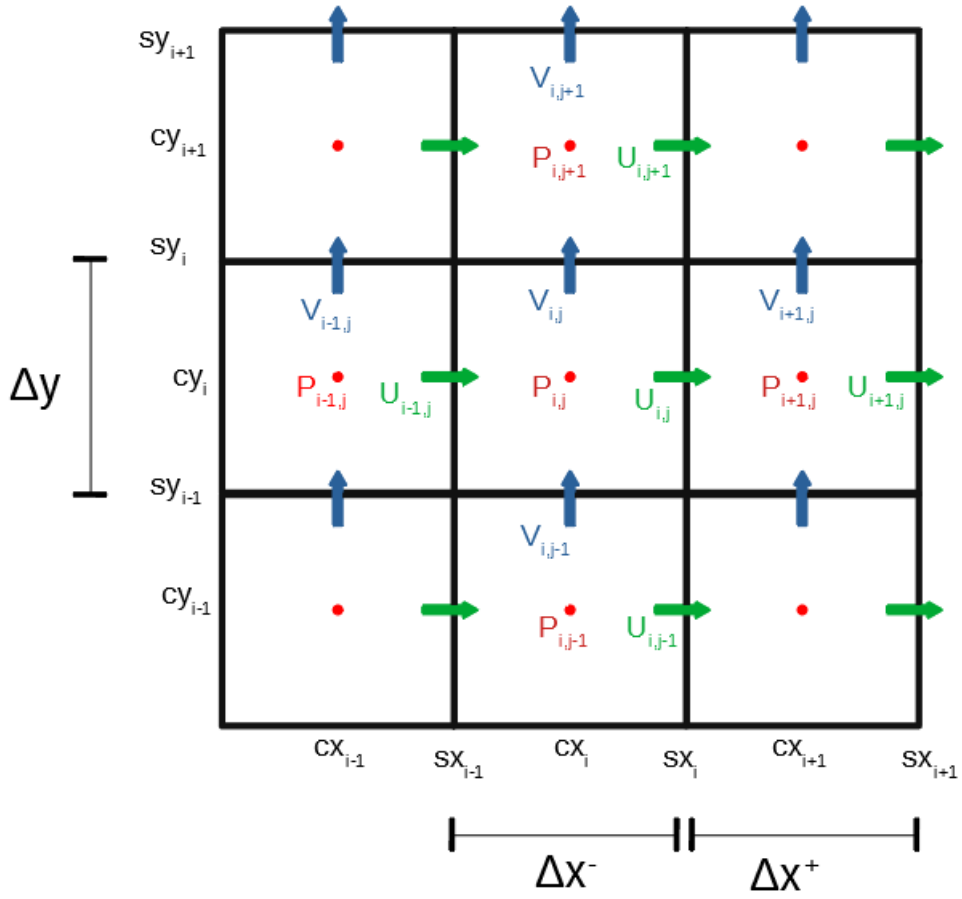

Figure 2: Mesh description

Then, in order to calculate the velocities variation a control volume must be set as will be explained later. But the nomenclature for this control volume will follow the one seen in the figure 3, using north, south, est and west regions (n, s, e, w).

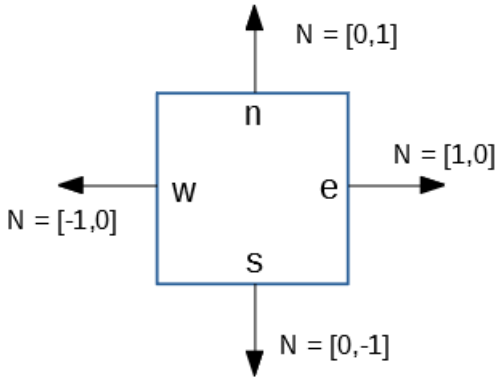

Figure 3: Coordinates nomenclature of an element

As this study is working with a staggered mesh, the control volumes used will be placed as in the figure 4. There, it can be seen that the control volume for a certain element in the X direction in green and in the Y direction in blue.

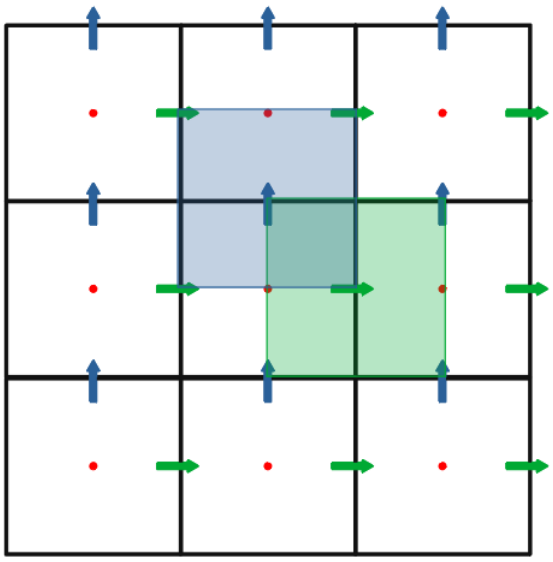

Figure 4: View of the control volume in the X and Y direction

5 Code validation

As explained, the validation of the code is divided in three parts, the implementation of the convective and diffusive terms, the coupling of the pressure and velocity and the time integration. Each one must be explained differently [7].

The results of the validation are plotted with the same Python code that has performed the study. The study cases like the periodic and driven cavity will plot the results using a Matlab code as is explained later. So, for the validation case, the pictures of the velocity field are plotted using a Python function *matplotlib.pyplot.quiver* and the streamliner with the function *matplotlib.pyplot.streamplot*.

5.1 Part A: convective and diffusive terms

In order to validate the convective and diffusive terms, the method of manufactured solutions (MMS) will be used. For this case, a velocity field will be set and it must be solved analytically and numerically. Then, the error between the solutions must be compared in function of the element size. It is obvious that the smaller the element size, the smaller the error should be.

The velocity field chosen is the following:

$$U = \cos(2\pi x)\sin(2\pi y)$$
$$V = -\cos(2\pi y)\sin(2\pi x)$$

The symbolic evaluation has been performed using the differentiation Python function *sympy.diff*.

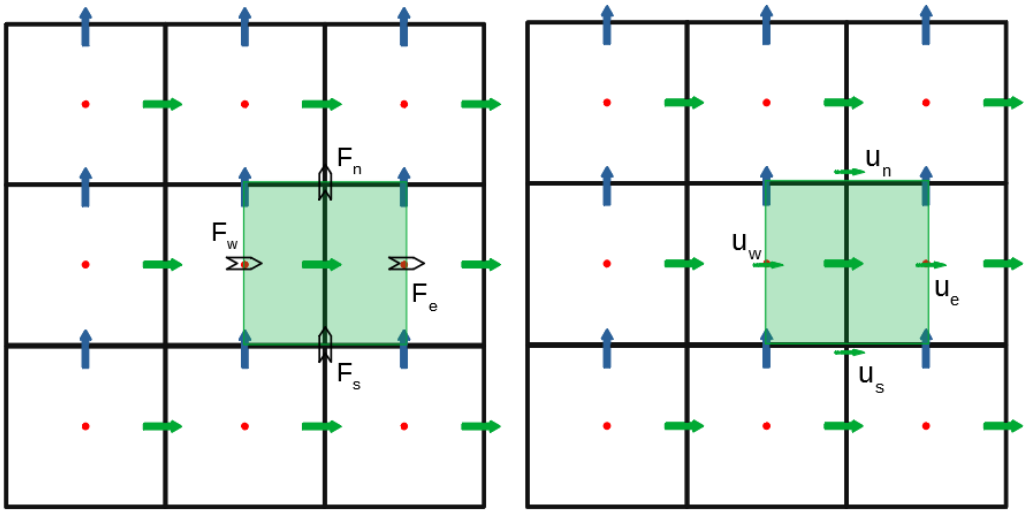

Figure 5: Velocity and flow terms in the U control volume

The numerical evaluation is more complex because each term must be discretized correctly. In order to understand the location of the control volume and the positions and directions of the velocities and flow terms, the figure 5 shows the case for the X direction.

5.1.1 Convective term discretization

Starting from the convective term and applying the divergence theorem for the X direction the equation can be derived as follows:

$$\int_V \nabla \cdot (u_1 u)\, dV = \int_S (u_1 u)\, n\, dS =$$
$$\int_e u_1 u_1 dS - \int_w u_1 u_1 dS + \int_n u_1 u_2 dS - \int_s u_1 u_2 dS \simeq$$

Which can be written as:

$$\simeq u_e F_e - u_w F_w + u_n F_n - u_s F_s$$

Where u is the velocity term and F is the flow term. They are discretized as:

$$u_e = \frac{u_{(i+1,j)} + u_{(i,j)}}{2} \qquad F_e = \frac{u_{(i+1,j)}\Delta y + u_{(i,j)}\Delta y}{2}$$

For the surface east (e) and X direction.

As can be seen, the derivation of the convective term is performed for each direction of the velocity, for a two dimensional case, for X and Y directions. It also requires the velocity at each direction independently of the direction that is being calculated.

5.1.2 Diffusive term discretization

From the diffusive term it can be developed as follows applying the divergence theorem.

$$\int_V \nabla \cdot (\nabla \cdot u_1)\, dV = \int_S \nabla \cdot u_1\, n\, dS =$$
$$\int_e \frac{\partial u_1}{\partial x} dS - \int_w \frac{\partial u_1}{\partial x} dS + \int_n \frac{\partial u_1}{\partial y} dS - \int_s \frac{\partial u_1}{\partial y} dS \simeq$$

Which can be written like:

$$\simeq \Delta y \left.\frac{\partial u_1}{\partial x}\right|_e - \Delta y \left.\frac{\partial u_1}{\partial x}\right|_w + \Delta x \left.\frac{\partial u_1}{\partial y}\right|_n - \Delta x \left.\frac{\partial u_1}{\partial y}\right|_s$$

Where each term are discretized as:

$$\left.\frac{\partial u_1}{\partial x}\right|_e \simeq \frac{u_{1(i+1,j)} - u_{1(i,j)}}{sx_{(i+1)} - sx_{(i)}}$$

Using a second order Taylor's expansion and for the east (e) surface at X direction.

It is observed that for the diffusive term it must also computed for each velocity direction. However, for this case each calculation only requires the velocity at the direction that is being calculated.

5.1.3 Part A results

After the implementation of the code explained before, it has been possible to validate the two functions that compute the convective and diffusive terms.

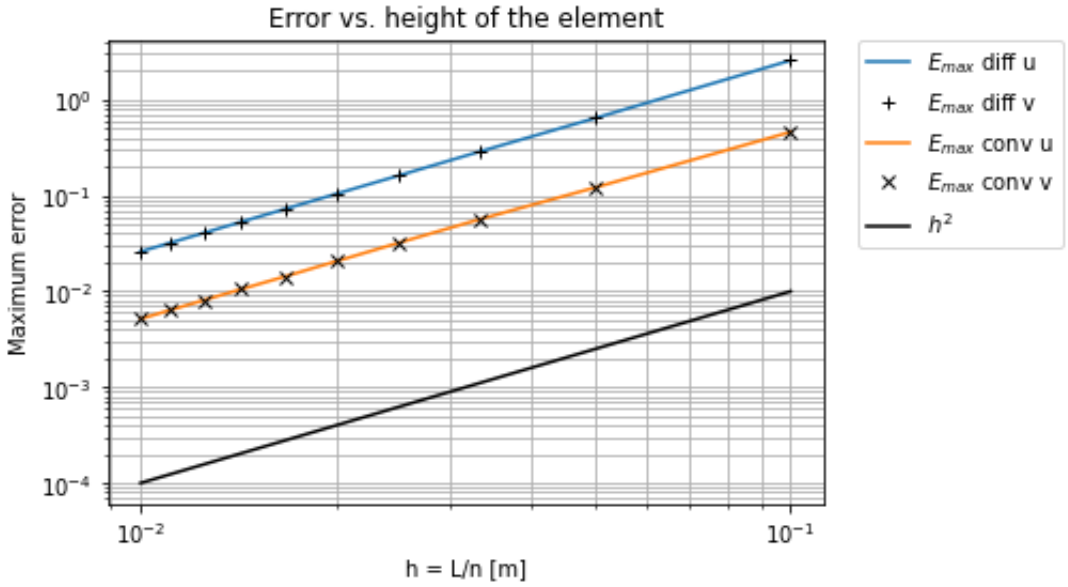

Figure 6: Convective and diffusive terms error versus element height

In the figure 6 it can be observed that the error decreases with the element size. This means that the numerical discretization is well performed because the higher the number of elements, the diffusive and convective calculation is performed with more precision.

It can be seen that the error found for the diffusive term is higher than for the convective term. This can be because a second order Taylor's approximation must be used to discretize the velocity derivative at each face of the control volume.

5.2 Part B: Pressure-velocity coupling

The second part consist of coupling the pressure and the velocity. This part of the code will be divided in three sub-codes to validate three cases.

The first case, and the most simple one will consist of building the pressure-velocity coupling for a periodic case. The second one will consist of a case where the domain is surrounded by solid walls and the third one a wall case but with inlets and outlets in the wall.

All three cases are going to be validated with the same procedure. First, a non divergence-free field will be set. It will consist of a zero velocity field

with an imposed velocity at a certain point of the domain. Then, from this velocity field the Poisson equation must be solved as will be explained to compute the new velocity step. It must be said that the boundary conditions must be taken into account for each three cases. Then, the new velocity field computed must be divergence-free in order to validate the code. If this is achieved, the Poisson solver and the boundary conditions will be well implemented.

5.2.1 Code implementation

In order to understand the pressure-velocity coupling, the code will be explained for a periodic case and at the following sections the modifications performed will be explained.

As can be seen from the momentum equation, the velocity and pressure are related:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j}$$

The momentum equation can be derived and expressed as follows:

$$\frac{du}{dt}V + C(u) = -\frac{V}{\rho}\nabla \cdot p + D(u)$$

Where u is the velocity vector with both components, C and D are the convective and diffusive terms, respectively. Then p is the pressure and V is the volume of the control volume at the corresponding direction.

Now, the velocity in function of time has to be isolated.

$$\frac{du}{dt} = R - \frac{1}{\rho}\nabla \cdot p$$

Where R is defined like:

$$R = -C/V + D/V$$

Now, the velocity must be discretized along time to be able to compute the next velocity step from the previous one. In order to solve this case, the Adams-Bashforth explicit method for ordinary differential equations is used [11]. The discretization can be observed in the equation 12.

$$\frac{u^{n+1} - u^n}{\Delta t} = \frac{3}{2}R^n - \frac{1}{2}R^{n-1} - \frac{1}{\rho}\nabla \cdot p^{n+1} \tag{12}$$

The velocity and R terms use an explicit method and the pressure gradient uses an implicit method.

Now, the velocity of the next step must be isolated.

$$u^{n+1} = u^n + \Delta t \left(\frac{3}{2} R^n - \frac{1}{2} R^{n-1} \right) - \frac{\Delta t}{\rho} \nabla \cdot p^{n+1}$$

Now, the equation will be grouped into three terms. The left term is the unknown velocity (velocity of the next step), the right term is the effect of the pressure and the rest of the equation: $u^n + \Delta t \left(\frac{3}{2} R^n - \frac{1}{2} R^{n-1} \right)$ will be the predictor velocity.

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

As can be seen, the predictor velocity can be obtained from the known velocity and the convective and diffusive terms. Then, rewriting the equation it stays as:

$$u^{n+1} = u^p - \frac{\Delta t}{\rho} \nabla \cdot p^{n+1}$$

Now, the mass conservation equation will be imposed to the velocity of the next step.

$$0 = \nabla \cdot u^{n+1} = \nabla \cdot u^p - \frac{\Delta t}{\rho} \nabla^2 \cdot p^{n+1}$$

With this, it is obtained that:

$$\nabla \cdot u^p = \nabla^2 \left(\frac{\Delta t}{\rho} p^{n+1} \right)$$

Where the right term is the divergence of a gradient, which will be the laplacian. This will be called the pseudo-pressure and will be expressed as follows.

$$\tilde{p} = \frac{\Delta t}{\rho} p^{n+1}$$

From this, it is easy to obtain the Poisson equation.

$$\boxed{\nabla^2 \tilde{p} = \nabla \cdot u^p}$$

From the predictor velocity the pseudo-pressure can be found. with this, it will be easy to obtain the pressure and then the new velocity step.

$$u^{n+1} = u^p - \nabla \cdot \tilde{p}$$

It is known that from a given vector field w defined in a bounded domain Ω with smooth domain $\partial\Omega$ is uniquely decomposed in a pure gradient field and a divergence-free vector parallel to $\partial\Omega$ [12].

$$w = a + \nabla \cdot \phi$$

For this case:

$$u^{n+1} = u^p - \nabla \cdot \tilde{p}$$

If u^{n+1} is divergence-free, the scalar field has to be the pseudo-pressure.

Discretization of the Poisson equation From the Poisson equation described previously, it is clear that the velocity is known and the objective is to find the pressure [13].

$$\nabla^2 \tilde{p} = \nabla \cdot u^p$$

To do so, for a certain point of the centered mesh and assuming that the mesh is uniform.

$$\Delta\frac{p_e - p_p}{\Delta} - \Delta\frac{p_p - p_w}{\Delta} + \Delta\frac{p_n - p_p}{\Delta} - \Delta\frac{p_p - p_s}{\Delta} = \Delta v_p - \Delta v_s + \Delta u_p - \Delta u_w \quad (13)$$

Where the velocities are located in the staggered mesh and the Δ is the distance of the control volume and between centered nodes because the mesh is considered uniform.

$$p_e + p_w + p_n + p_s - 4p_p = \Delta(v_p - v_s + u_p - u_w) \quad (14)$$

Which can be expressed in a matrix way.

$$Ap = b$$

Where the pressure is the unknown and the vector b and the matrix A are known.

If the mesh is not uniform, the coefficients at the pressure shown in the equation 14 will depend on the distances of the element. So for a mesh that is not uniform, the equation 13 will be written as:

$$C_e p_e + C_w p_w + C_n p_n + C_s p_s - C_p p_p = \Delta(v_p - v_s + u_p - u_w) \quad (15)$$

Where as said, the coefficients C_e, C_w, C_n, C_s and C_p now must be computed.

$$\begin{aligned} C_e &= Dy/dxp \\ C_w &= Dy/dxn \\ C_n &= Dx/dyp \\ C_s &= Dx/dyn \\ C_p &= C_e + C_w + C_n + C_s \end{aligned}$$

For the case of the coefficient C_e, each distance is observed in the figure 7. If the element of the center of the figure is the one that is being studied, the coefficient C_e will use the seen distances. It is clear that for a non uniform mesh, the distances will be affected changing the corresponding coefficients.

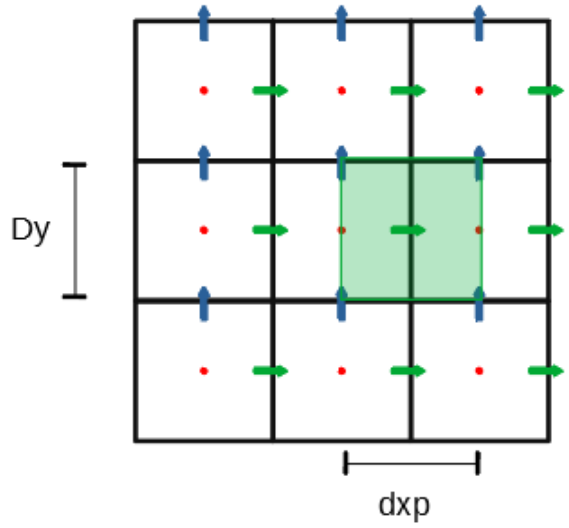

Figure 7: Coefficient C_e geometry

Matrix *A* implementation As said, the pressure will be found from the known velocity and the matrix A that will be square, singular and symmetric. It will have a shape of $Nx*Ny$ by $Nx*Ny$. As seen before in the equation 14, the matrix A will be composed by the coefficients that multiply the pressure and relate them with the corresponding nodes.

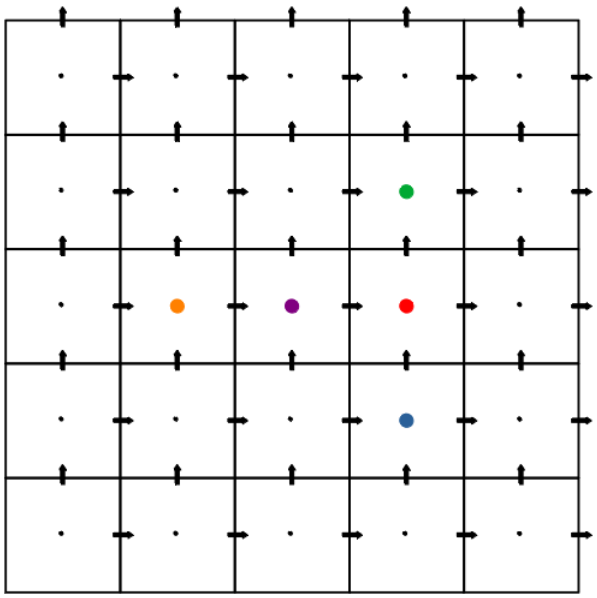

Figure 8: Node positions for the building of the matrix A

From this, it is easy to see that each row of the matrix A will have the -4 coefficient in the corresponding node and a coefficient of value one in the four nodes of the surroundings. For the nodes in the borders, it will be necessary to use the halo nodes to set the correct position in the matrix. With this, it is easy to see that the sum of each row and column of the matrix is zero. As long as the boundary conditions are consistent, the sum of the vector b also has to be zero. For this reason, the matrix A has to be perturbed in order to be able to solve the system. In this case, in the nodes *A(1,1)* is will be set the value -5.

In the figure 8 it is represented with red a certain node and the nodes in their boundaries. It is clear that the node at the east position is located in the halo so the node taken must be the orange one.

To obtain the pressure vector a very high computational power will be needed because of the dimensions of the matrix A. To solve this, the matrix has been set as a scattered matrix to save the locations of the zeros. And the pressure vector has been obtained with a solve function from the Python code. With this, a big amount of time is saved.

5.2.2 Part B1: Periodic case

For the periodic case, the implementation will be as explained before, without modifying the coefficients of the matrix A. In order to validate the code, it must be observed if the velocity field after the step is divergence-free.

```
Divergence value of velocity field: 7.37e-02
Divergence value after solving Poisson: 5.08e-17
```

Figure 9: Picture of the command window with the divergence values

It can be observed 9 that the maximum value of the divergence of the new velocity field is zero.

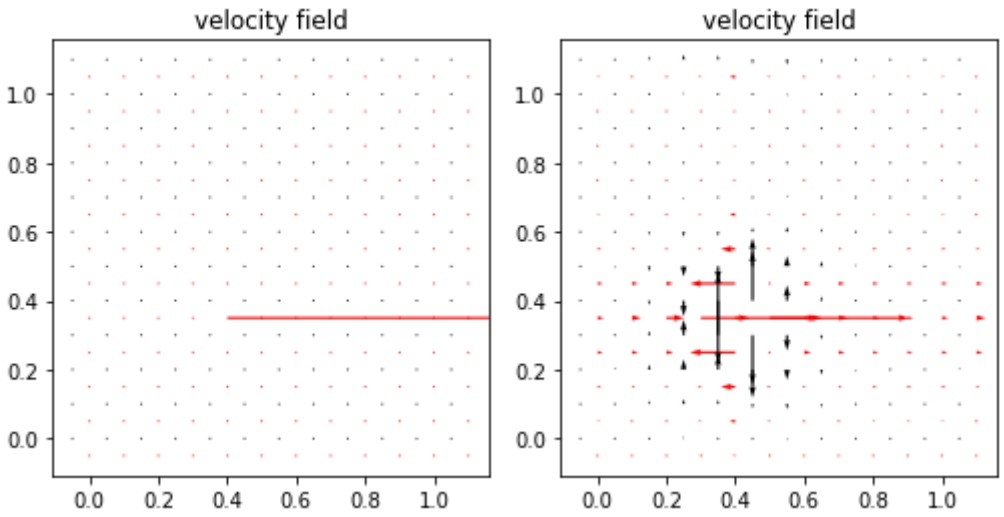

Figure 10: Velocity field before and after the step

The figure 10 shows the non divergence-free field at the left and the divergence-free field at the right. As the theory said, the velocity placed in the first field has perturbed the whole domain.

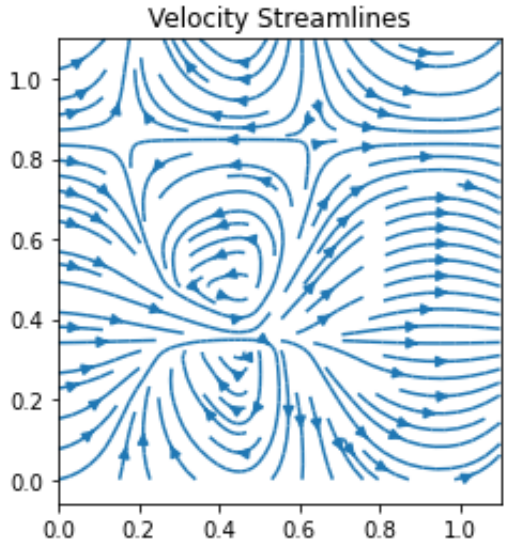

Figure 11: Streamlines of the velocity

The figure 11 shows the next velocity step once the Poisson equation is solved. The velocity streamlines are very useful to observe that the boundary conditions have been correctly implemented as the lines follow the periodic domain.

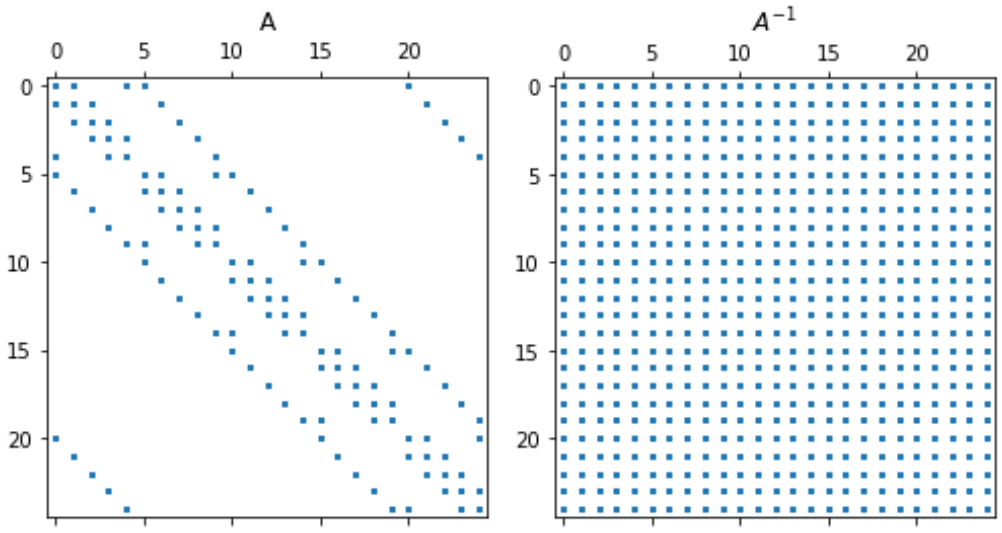

Figure 12: A and A^{-1} matrix for N = 5

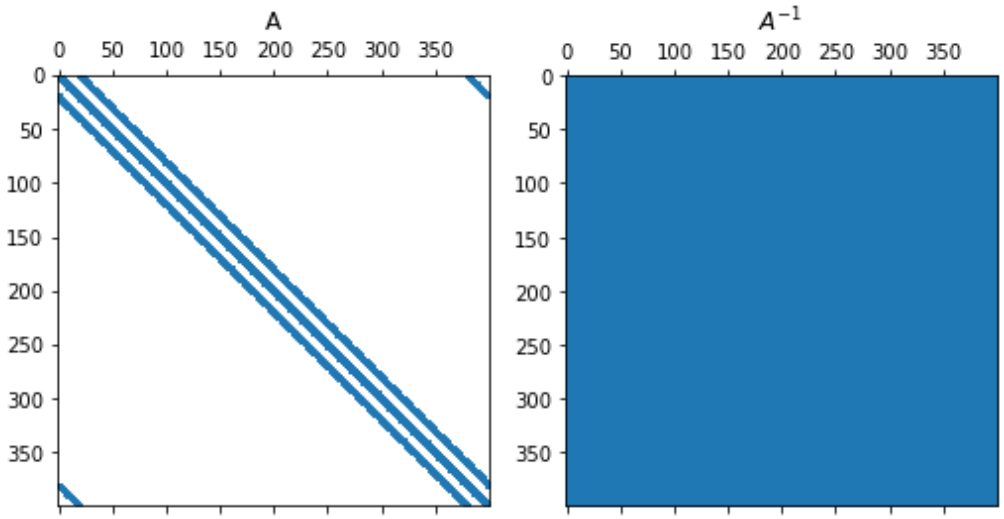

Figure 13: A and A^{-1} matrix for N = 20

In the figures 12 and 13 the matrix A and A^{-1} are represented. The blue dots are the positions in the matrix that are different from zero. It can be visually observed that the computation of this matrix will increase exponentially with the number on nodes.

5.2.3 Part B2: Wall case

For the wall case, a small change has to be added in order to make the code understand that the step must be performed with walls. It is clear that when the new velocity field is computed the values in the boundaries must be zero at each direction. This will be imposed making a modification in the pressure and in the building of the matrix A.

Starting from the equation 13, and using a node next to the east wall, the modifications are explained. In the example proposed, the pressure at the node will have the same value than the pressure at the east position to achieve a zero velocity in the wall.

The equation 14 has been used for a periodic case with uniform mesh. But now, with a wall case, the pressure at the east position has the same value than the pressure at the point. This will transform the equation 14 to:

$$p_p + p_w + p_n + p_s - 4p_p = \Delta(v_p - v_s + u_p - u_w)$$

Which must be rearranged.

$$p_w + p_n + p_s - 3p_p = \Delta(v_p - v_s + u_p - u_w)$$

And for a non uniform mesh, from the equation 15, the pressure at the east position must be equal to the pressure at the node.

$$C_e p_p + C_w p_w + C_n p_n + C_s p_s - C_p p_p = \Delta(v_p - v_s + u_p - u_w)$$

Then, the equation is written as:

$$C_w p_w + C_n p_n + C_s p_s - (C_w + C_n + C_s) p_p = \Delta(v_p - v_s + u_p - u_w)$$

It is clear that implementing the code for a non uniform mesh can be used for both meshes as it is more general. In the code of the study, the non uniform mesh has been implemented.

After that, and when the pressure is computed, the modification in the borders must be performed. It will be imposed that the pressure of the last and next-to-last node are the same. With this, the velocity computed for the next step will be zero at the wall.

```
Divergence value of velocity field: 5.00e-02
Divergence value after solving Poisson: 3.71e-17
```

Figure 14: Picture of the command window with the divergence values

The figure 14 shows the maximum values of the divergence of the velocity field before and after the computation of the Poisson equation.

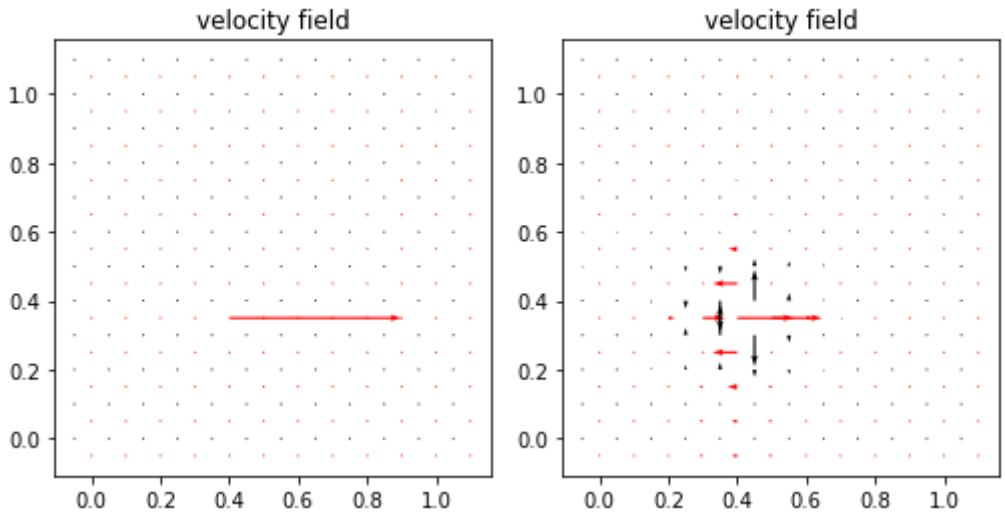

Figure 15: Velocity field before and after the step

In the figure 15 it can be seen a small difference from the periodic case. This is that the velocity at the walls is zero.

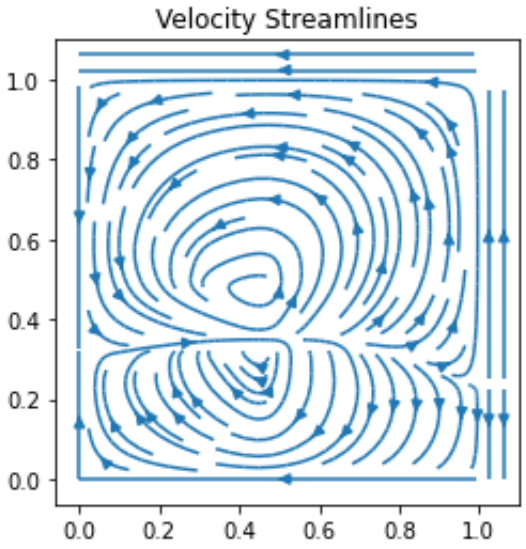

Figure 16: Streamlines of the velocity

In the streamlines figure 16 it can be seen how the vortex are closed in the domain. This means that the flow must adapt to the new boundaries.

5.2.4 Part B3: Inlet and outlet case

The case will consist on a wall domain with an inlet and outlet. The boundary conditions must be set like the previous case but here, with an imposed velocity in the inlet and outlet.

```
Divergence value of velocity field: 1.00e-02
Divergence value after solving Poisson: 3.05e-17
```

Figure 17: Picture of the command window with the divergence values

And for this case, the maximum divergence value is also zero after the new velocity field is computed.

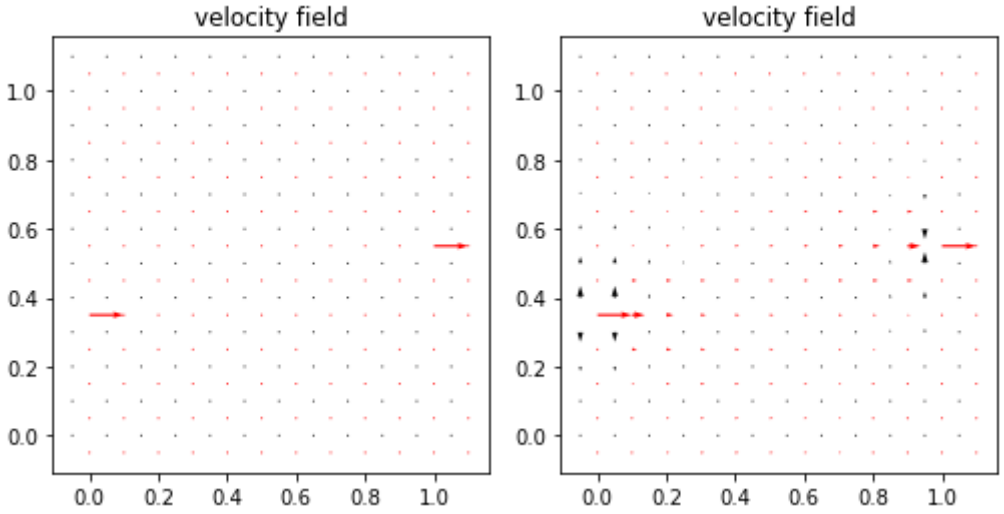

Figure 18: Velocity field before and after the step

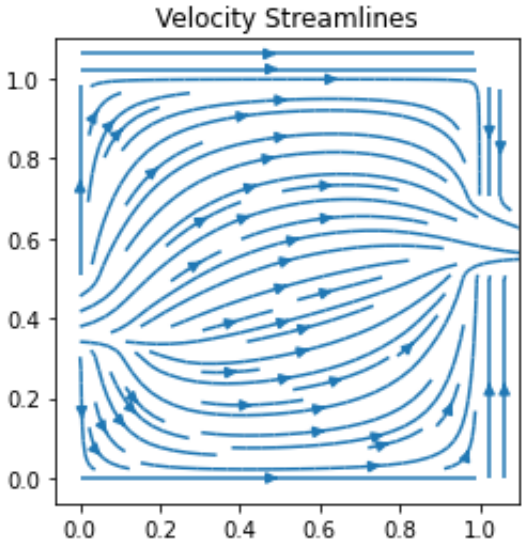

Figure 19: Streamlines of the velocity

In order to obey mass conservation, the sum of the inlet and outlet velocities must be zero. For example, if is set inlet velocity of 1 and an outlet velocity of 2, the mass conservation will not be fulfilled and the divergence of the next step field will not be zero. This is also valid for more than two inlet and outlets. If the inlet velocity is 1 and two outlets have a velocity of 0.5, the step will be performed correctly.

As said before, the sum of the vector b must be zero as the columns of the matrix A. In this case, this can be used to see if the step is performed correctly.

5.3 Part C: Time integration

The main objective of the part C is to implement a code that can perform a flow simulation by applying an iterative process where each step is the velocity computation seen in the previous section. It must be taken into account that each time step must be evaluated before calculating the next velocity field.

To validate the code, this part consist of taking a flow field that can be solved analytically and compare the results with the numerical solution. The flow chosen is the following:

$$\begin{bmatrix} u \\ v \end{bmatrix} = F \begin{bmatrix} cos(2\pi x)sin(2\pi y) \\ -cos(2\pi y)sin(2\pi x) \end{bmatrix}$$

Where F contains the time variable.

$$F = e^{-8\pi^2 \nu t}$$

In this part, it will be obtained as a result the comparison between numerical and analytical velocity. A certain point will be chosen and the velocity will be obtained at each step.

5.3.1 Code implementation

The implementation of the code consist of the iterative process, the time step evaluation and saving the data.

The time step evaluation is very important, since the mesh dimensions are constant and the velocity is constantly changing, the time is the parameter that relate both previous variables. Considering one control volume, if the time step is very big, the velocity of the flow will span more than one control volume, making it impossible for the code to converge. For this reason, the time step must be below the convective and diffusive time limit.

$$\Delta t_c \leq Min\left(\frac{\Delta}{|u|}\right)$$

$$\Delta t_d \leq \frac{1}{2} Min\left(\frac{\Delta^2}{\nu}\right)$$

$$\Delta t = f * Min(\Delta t_c, \Delta t_d)$$

Where f must be between 0.2 to 0.5. Of course, this evaluation is easy if the mesh is constant but if the mesh is changed, this must be taken into account in the time evaluation.

In order to compare the analytical and numerical flow evolution, the velocities will be evaluated from the starting point to the moment that the velocity has been faded. And for each step, the error from both velocity field will be saved. This computation is performed for different numbers of elements to observe if the error between both field is lower or higher. It is assumed that the lower the element size, the numerical computed velocity field must be closer to the analytical one.

5.3.2 Part C results

The results of this part can be observed in the following figures. This code validates the function that iterates the problem until the convergence is reached.

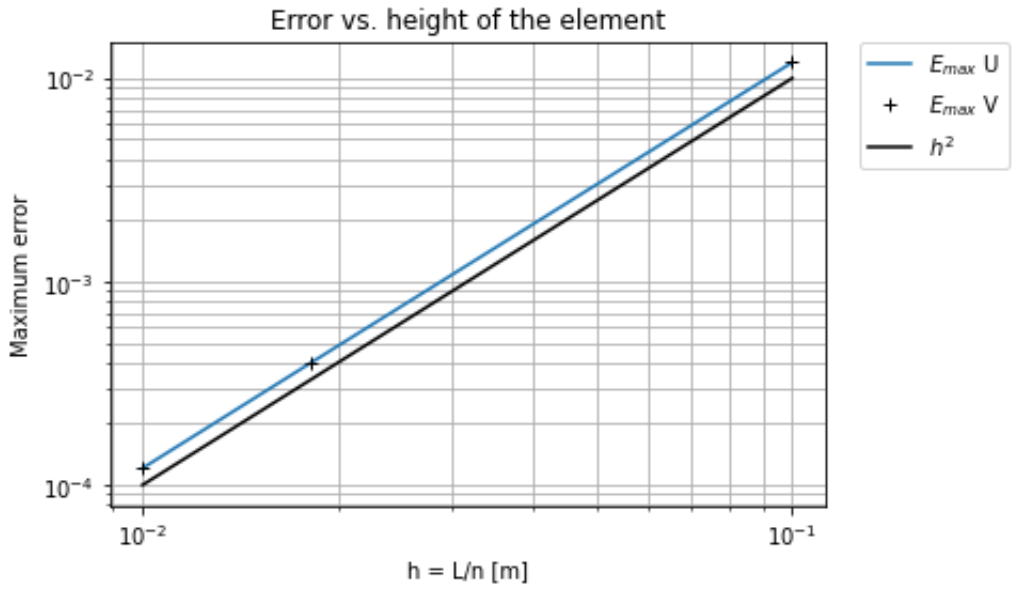

Figure 20: Velocity error versus element height for N = 10, 55 and 100

As expected, the error between velocity field becomes lower when the number of elements increases. This means that a lower element size will be able to compute better the real flow field.

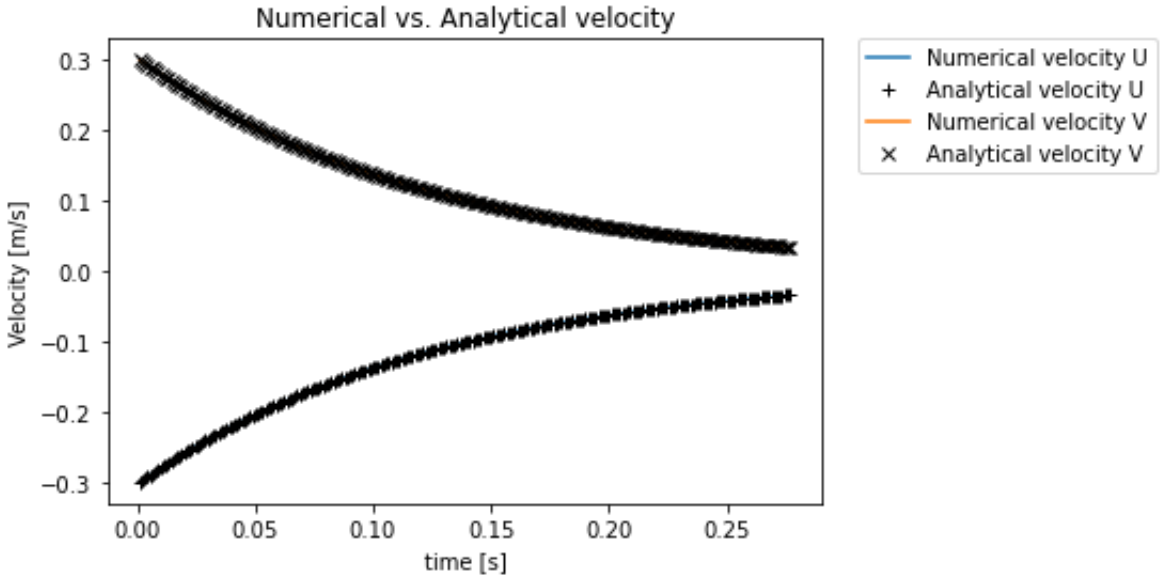

Figure 21: Velocities comparison for 200 steps

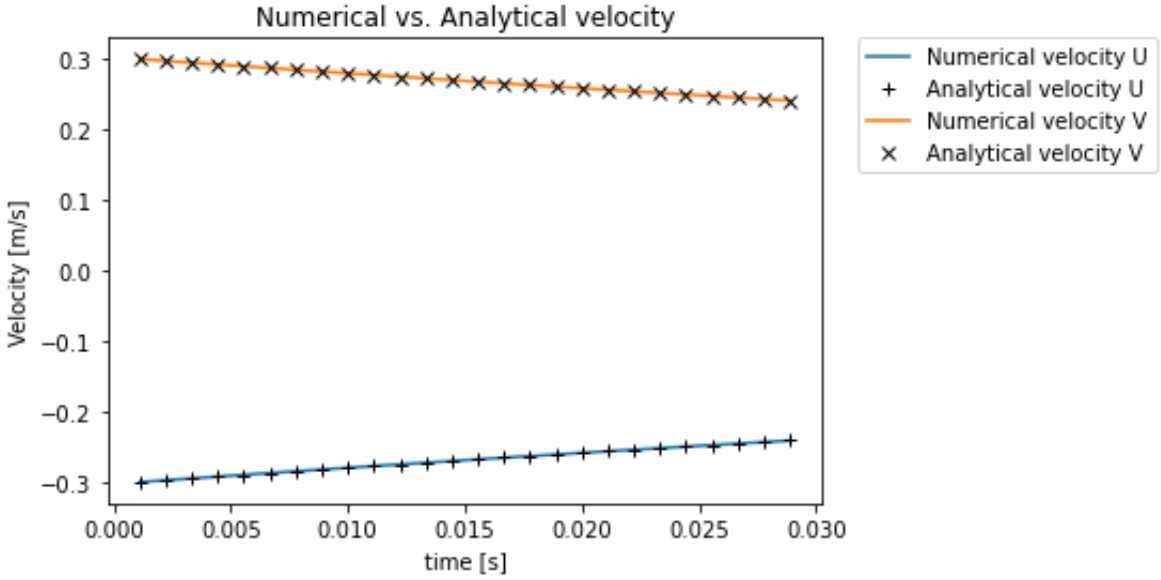

Figure 22: Velocities comparison for 25 steps

Two things must be observed from the figures 21 and 22. The first one is that is very well represented how the velocity fades into a zero value along with time. Is is also very clear that the numerical calculations are very precise, which can also be concluded from the figure 20. Both figures show

the same tendency but it has been plotted for different number of steps in order to see more details.

6 Boundary conditions implementation

Depending on the case that is studied, different boundary conditions must be applied to the domain. The most used ones are the inlet, outlet and non-slip condition. And to implement them, Dirichlet or Neumann boundary conditions are used. The Dirichlet method consist on imposing a certain value, for example the velocity. And Neumann imposes the value of the derivative at the boundary. For this thesis, the Dirichlet boundary conditions are used [14]. The code will be able to set a non slip condition and a moving wall. This means the case where the velocity is zero at each direction and the case where the normal velocity is zero while the tangential has a certain value.

The figures 23 and 24 show how the velocities are distributed in each of the four walls. As can be seen, each one must be treated differently in function of the velocity direction because of its location in the staggered mesh.

It is important to say that when the boundary conditions are implemented, the calculation of the matrix A must be changed as in the second case of the part B of the validation. The modification explained in this section must be added in order to implement correctly the boundary conditions.

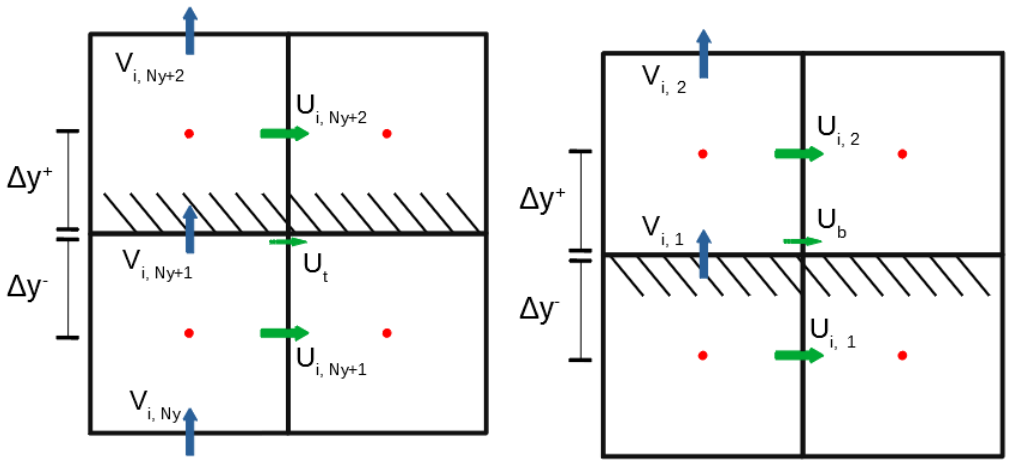

Figure 23: Top and bottom wall elements

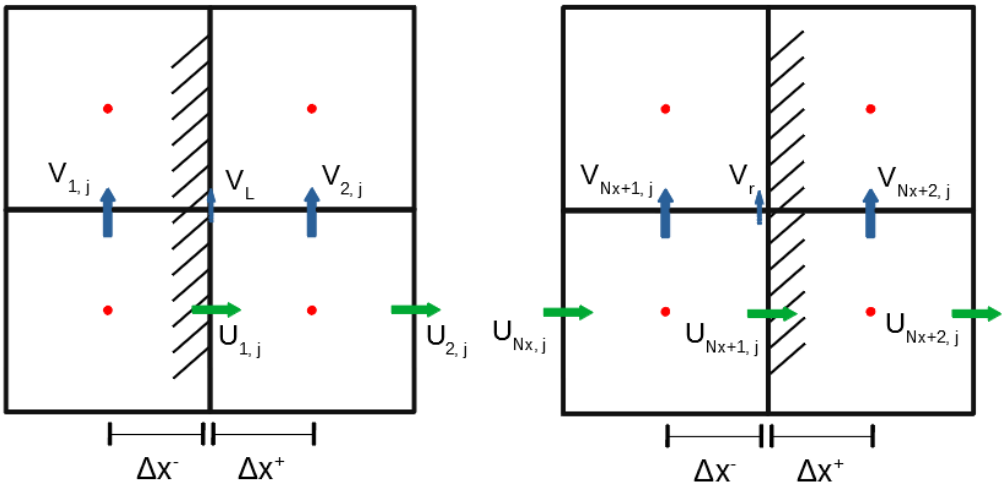

Figure 24: Left and right wall elements

In order to set the boundary conditions, the velocities in the halo will be modified in order to obtain the desired value when computing the convective and diffusive terms. Then, a different modification must be performed before the calculation of the convective and diffusive terms. Each modification will take into account the velocity direction and the boundary. In order to perform the mathematical explanation, it must be said that the Δy^+ and Δy^- distances must be equal and the same will happen with the Δx^+ and Δx^- distances. This will be reached for a uniform mesh, but if the code works with a concentrated mesh at the walls, it is very important to impose that the elements of the halo have the same size than the element next to them.

6.1 Convective term

As explained previously, the convective term calculation is reached once the velocity and flow terms are obtained. The main objective of the boundary conditions implementation is modifying the velocity terms to obtain the desired value of the face velocities.

6.1.1 Velocity U

Top and bottom walls : The top and bottom walls will be analysed in this paragraph. If the objective is to impose a certain value of the U_t and U_b velocities shown in the figure 25, it must be observed how the convective term computes this velocity term.

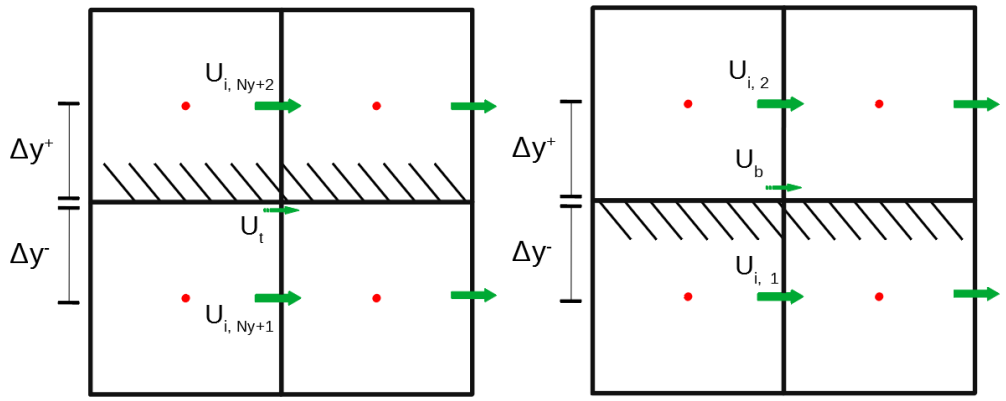

Figure 25: Top and bottom wall for U velocity

The convective term finds the mentioned U_t and U_b velocities interpolating from the known velocities of the staggered mesh.

$$U_t = \frac{U_{i,Ny+2} + U_{i,Ny+1}}{2} \qquad U_b = \frac{U_{i,2} + U_{i,1}}{2}$$

If U_t and U_b are the top and bottom velocities at the face that must be found when computing the convective term, it is imposed then (top and bottom respectively):

$$\underline{U_{i,Ny+2}} = 2U_t - U_{i,Ny+1} \qquad \underline{U_{i,1}} = 2U_b - U_{i,2}$$

It is clear then, that the velocities $U_{i,Ny+2}$ and $U_{i,1}$ are the ones that will be modified previous to the convective calculation for the top and bottom case.

Left and right walls The left and right walls have a different geometry due to the staggered mesh so the case must be treated differently. In the figure 26 it is easy to see that the velocities that must be imposed are the ones at the face of the wall.

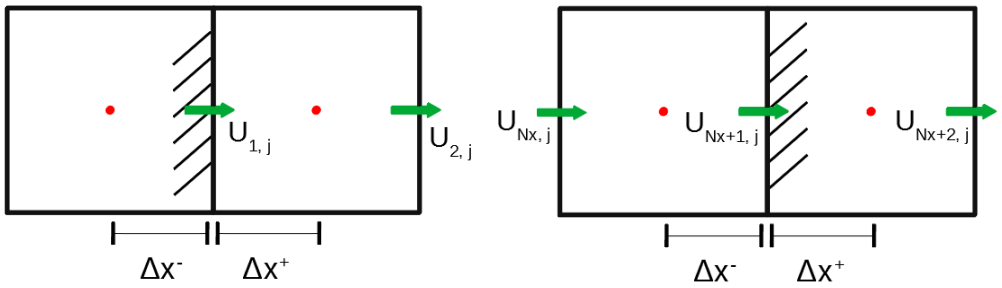

Figure 26: Left and Right wall for U velocity

It is imposed then (left and right respectively):

$$\underline{U_{1,j}} = U_l \qquad \underline{U_{Nx+1,j}} = U_r$$

For the case of the right wall, the velocity $U_{Nx+2,j}$ is neglected as it will not affect the velocities around the wall.

6.1.2 Velocity V

Top and bottom wall The procedure to find the imposed velocity for the top and bottom walls for the velocity in Y direction is analogous to the left and right walls for velocity in X direction.

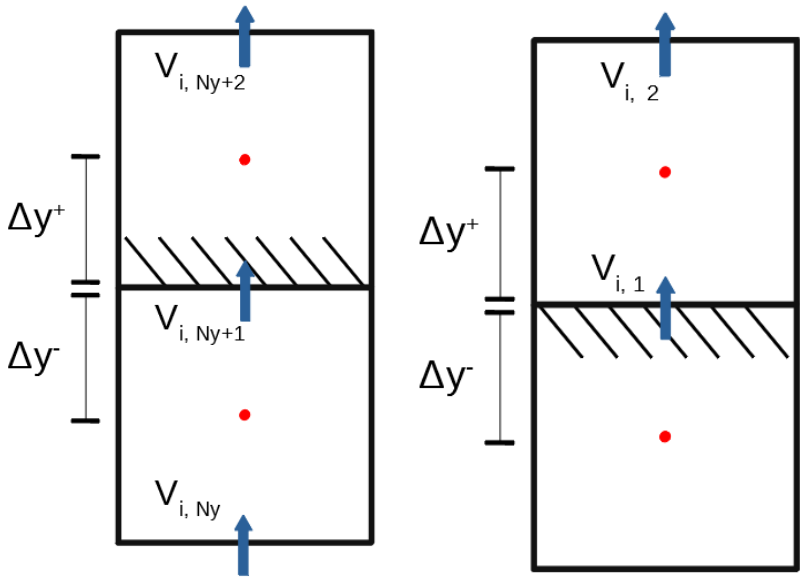

Figure 27: Top and bottom wall for V velocity

It is imposed (top and bottom respectively):

$$\underline{U_{i,Ny+1}} = U_t \qquad \underline{U_{i,1}} = U_b$$

Left and right walls And the same for the left and right walls in Y direction.

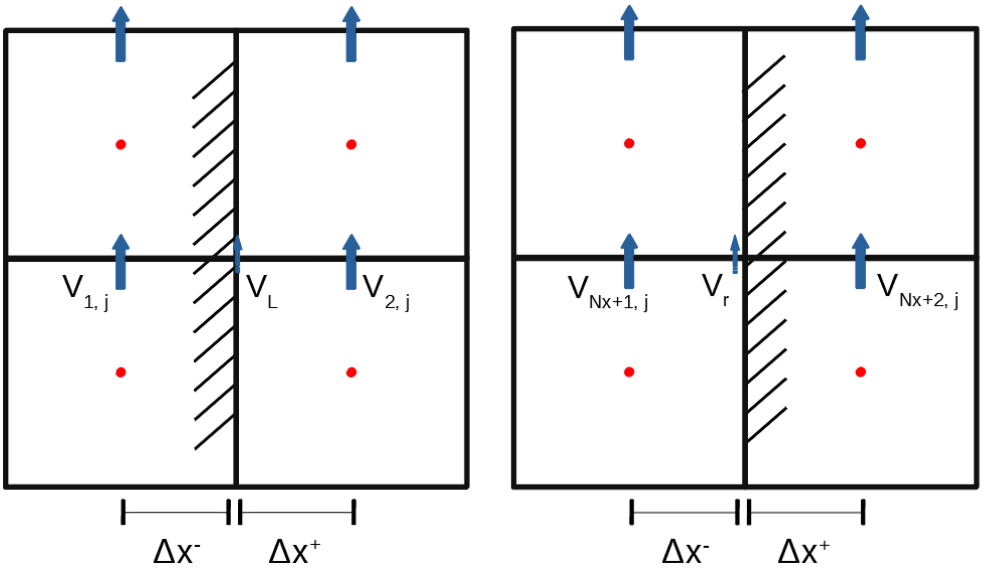

Figure 28: Left and Right wall for V velocity

$$V_l = \frac{V_{1,j} + V_{2,j}}{2} \qquad V_r = \frac{V_{Nx+2,j} + V_{Nx+1,j}}{2}$$

It is imposed then (top and bottom respectively):

$$\underline{V_{1,j}} = 2V_l - V_{2,j} \qquad \underline{V_{Nx+2,j}} = 2V_r - V_{Nx+1,j}$$

6.2 Diffusive term

In the diffusive term the same procedure is used than in the convective one. A previous modification of the velocities at the halo are going to be used in order to obtain the desired values of the velocities at the walls once the evaluation of the diffusive term is performed.

6.2.1 Velocity U

Top wall In this case, the figure 25 is used as a reference and the derivative term from the momentum equation is taken. Then, by making a first order approximation the derivative can be written as:

$$\left.\frac{\partial u_1}{\partial y}\right|_t = \frac{U_{i,Ny+2} - U_{i,Ny+1}}{\Delta y^+ + \Delta y^-} \simeq \frac{U_t - U_{i,Ny+1}}{\Delta y^-}$$

Arranging the equation to isolate the imposed velocity:

$$U_{i,Ny+2} = U_{i,Ny+1} + \frac{\Delta y^+ + \Delta y^-}{\Delta y^-}(U_t - U_{i,Ny+1})$$

As said previously, the element of the halo must have the same size than the element on its side. With this, the distances are the same so the equation is simplified as:

$$\underline{U_{i,Ny+2}} = 2U_t - U_{i,Ny+1}$$

Bottom wall To set the velocity for the bottom wall the figure 25 will also be the reference. Starting from the velocity derivative and performing a first order approximation the equation is written as:

$$\left.\frac{\partial u_1}{\partial y}\right|_b = \frac{U_{i,2} - U_{i,1}}{\Delta y^+ + \Delta y^-} \simeq \frac{U_b - U_{i,1}}{\Delta y^-}$$

The terms are rearranged:

$$U_{i,2} = U_{i,1} + \frac{\Delta y^+ + \Delta y^-}{\Delta y^-}(U_b - U_{i,1})$$

Considering the same element sizes and rearranging the terms it is obtained the following equation:

$$U_{i,2} = 2U_b - U_{i,1}$$

In this case, though, the velocity to be imposed is the one in the halo so the equation must be isolated differently.

$$\underline{U_{i,1}} = 2U_b - U_{i,2}$$

Left wall For the left wall the derivative of the velocity U is performed along the X direction as seen in the figure 26.

$$\left.\frac{\partial u_1}{\partial x}\right|_l = \frac{U_{2,j} - U_{1,j}}{\Delta x^+ + \Delta x^-}$$

In this case, the velocity $U_{1,j}$ of the halo is the same velocity of the left wall. So it is easy to see that imposing the velocity of the halo the wall velocity is imposed. It is clear then that it must be set:

$$\underline{U_{1,j}} = U_l$$

Right wall The right wall for the velocity U must be set differently as can be seen in the figure 26.

$$\left.\frac{\partial u_1}{\partial x}\right|_r = \frac{U_{Nx+2,j} - U_{Nx+1,j}}{\Delta x^+ + \Delta x^+} \simeq \frac{U_{Nx+1,j} - U_{Nx,j}}{\Delta x^- + \Delta x^-}$$

Then:

$$\underline{U_{Nx+2,j}} = 2U_r - U_{Nx,j}$$

6.2.2 Velocity V

For the velocity V of the diffusive term, the procedure to obtain the imposed velocities is analogous to the procedure for the U velocities. In order to simplify the report, they have not been added. As will be seen later, for the case of the driven cavity, this imposed velocities will be simpler to implement. This is because the imposed velocities for perpendicular directions to the wall are going to be zero.

7 Periodic Case

In this section, the results of a periodic case are shown. As said in the code validation section, the part C consist of taking a periodic case and comparing the velocity with the analytical results. In this case, the results of the simulation are shown in order to observe the movement of the flow.

As explained before, the periodic and driven cavity case results are plotted using a Matlab code. The maps of colors are plotted using the *contourf* function, the velocity vectors using the *quiver* function and the streamlines using the *streamline* function as can be seen in the attachments.

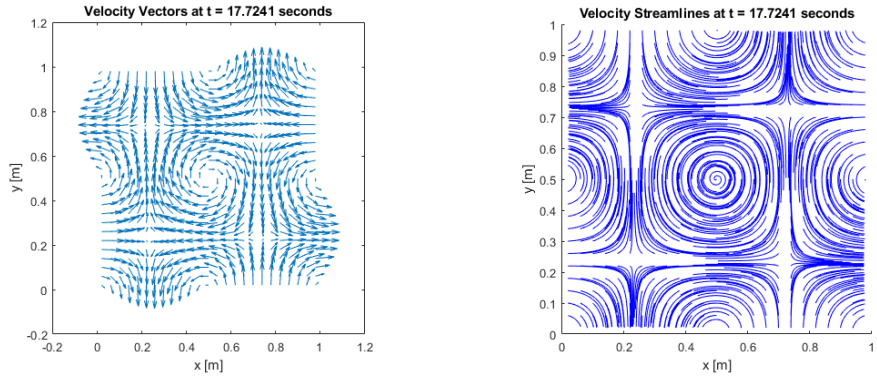

Figure 29: Velocity vectors and velocity streamlines for the periodic case

As seen in the figure 29, the eddy in the center is very clear moth in the velocity vectors and velocity streamlines figure. From the figure 29 it is very easy to observe that the case is periodic because a streamline that goes to the top boundary will appear at the same position in the bottom boundary.

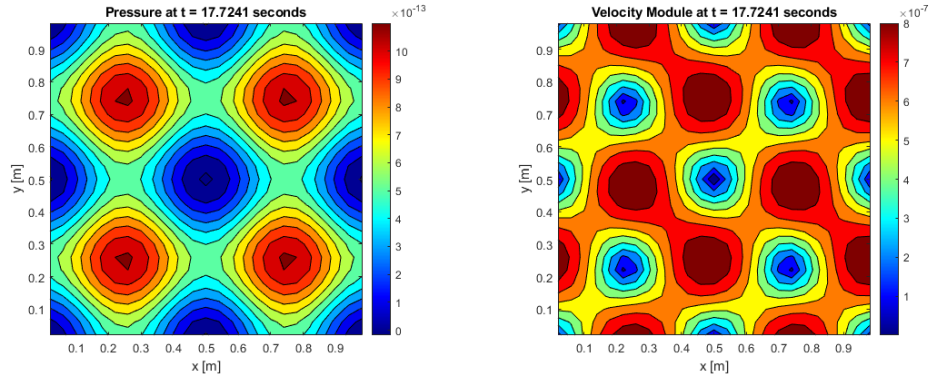

Figure 30: Velocity vectors and velocity streamlines for the periodic case

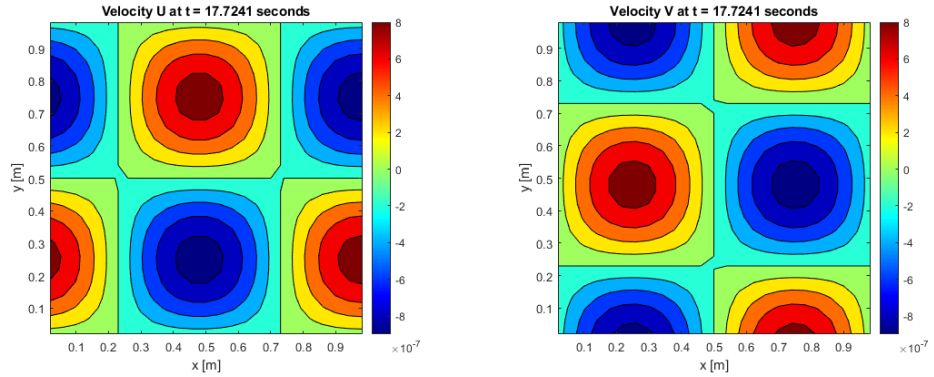

Figure 31: Velocity vectors and velocity streamlines for the periodic case

Let's refresh, the velocity flow field imposed initially:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} cos(2\pi x)sin(2\pi y) \\ -cos(2\pi y)sin(2\pi x) \end{bmatrix}$$

It is clear, and this has been proved in the validation section, that the flow velocity will fade until a negligible value. This can be observed in the figures 30 and 31, where the flow shape is the same but the velocity and pressure values are very low.

8 Case study: Driven cavity case

Once the boundary conditions are implemented in the code as explained, it is possible to perform the simulation of the driven cavity case. To implement the case, it is necessary to have a square domain that in the study is set as one meter length and height. Then for the bottom, left and right boundary it is necessary to set a wall condition where the velocity of the wall is zero. And in the top boundary, the wall must be set with a velocity of one meter per second. It must also be implemented the changes in the building of the matrix A and the modification of the pressure halo as explained in the second case of the part B of the validation of the code.

As said, the driven cavity case results are shown using the *contourf*, *quiver* and *streamline* Matlab functions.

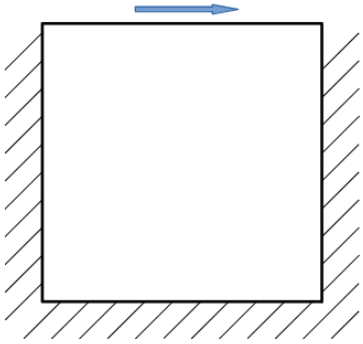

Figure 32: Geometry of the driven cavity case

Before starting the case, the convergence criteria must be set. It will consist of evaluating the velocity change between one step and the next one. The maximum error of each node and velocity direction will be the reference value. When this error has reached a certain value, the simulation is considered converged.

Once the simulation is completed, it is important to compare it with experimental results and other simulations from articles in order to validate the code.

8.1 Results validation

In order to validate the Driven Cavity case, different results of experiments and simulations are used.

Reference	Study	Number of elements
Ghia 1982 [1]	Experiment	-
Bruneau 2006 [2]	Simulation	1024x1024
Botella 1998 [3]	Simulation	160x160
Vanka 1986 [4]	Simulation	321x321
Yapici 2013 [5]	Simulation	768x768

Table 1: Mesh size and number of elements for each Reynolds number

Some of the mentioned articles results are going to be used in order to validate the code developed in this report. Other articles are going to be used to compare the streamlines of the flow.

As the experiment of the article [1] have results for low Reynolds, they are going to be used to validate the code. It is considered that these results are accurate enough because as seen in [2], the experimental results are coincident with the simulations.

The four simulations used to compare the results use a direct method with high number of elements.

8.2 Mesh study

In order to make a good simulation, the mesh must be constructed carefully. As known, the most critical point of the domain is the region near the wall, where the element size must be small, but increasing the number of elements will increase the computational time. For this reason, the element size near the wall must be studied.

In the following sections two studies of the mesh will be performed. The first one, for a uniform mesh and Reynolds number of 100, the number of elements will be changed. And the second case for Reynolds number 1000 and 20 number of elements each direction, the concentration to the walls will be changed.

8.2.1 Uniform mesh study

As said, the mesh in this section will be uniform and the purpose of the study is to evaluate the size of the elements that must be used to perform a correct simulation of the driven cavity case.

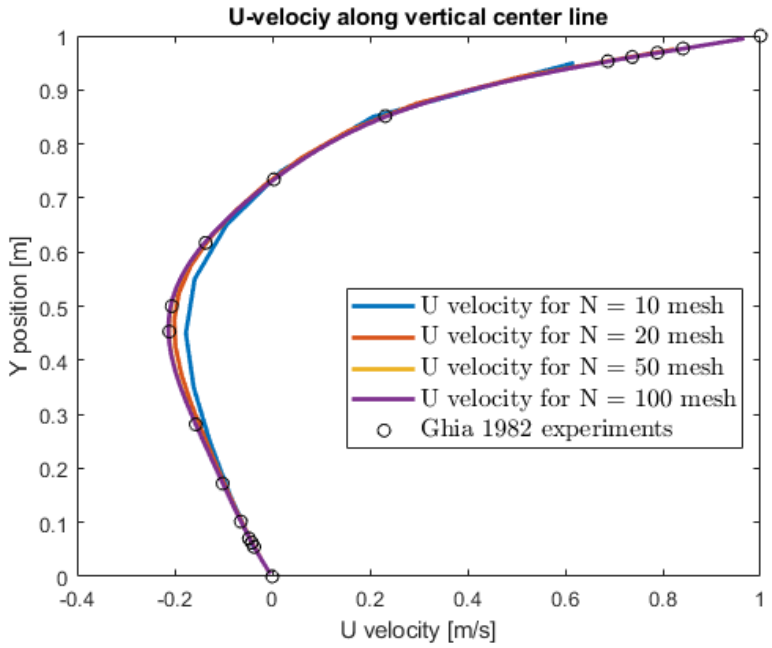

Figure 33: U velocity for different number of elements

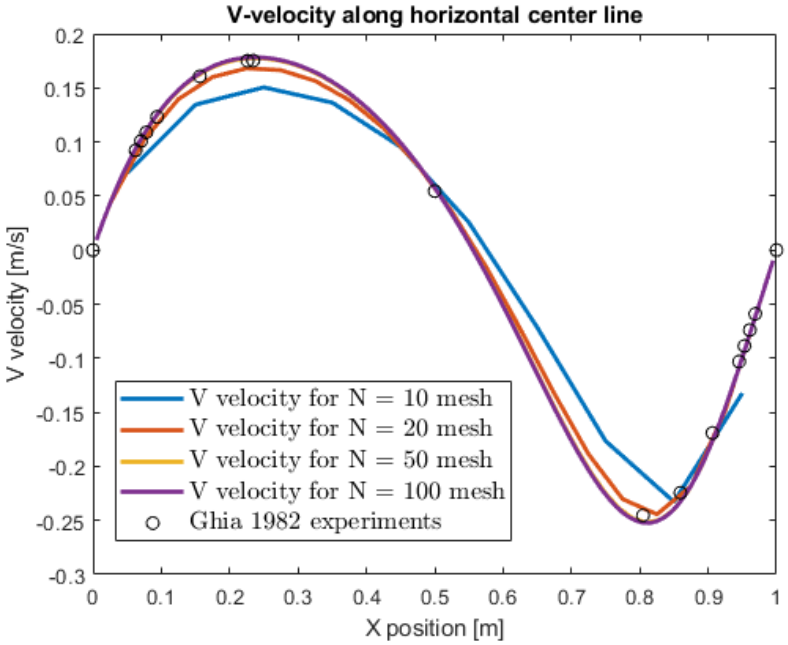

Figure 34: V velocity for different number of elements

As can be seen in the figures 33 and 34, the number of elements must be

higher than 20 and if possible closer to 50 in order to obtain a good result for this Reynolds number.

This study also shows that the discretization method is correct because as the number of elements gets higher, the results converge to the correct solution. It is clear in the images 33 and 34 that from 50 to 100 elements, the change of results are very small.

Once the Reynolds number is increased, it is clear that the number of elements needed are higher. This must be taken into account when performing simulations at higher Reynolds number.

8.2.2 Concentrated mesh study

In this section, it will be studied how the concentration of the mesh at the walls affect the result of the simulation. As said before, the Reynolds number chosen is Re = 1000 and number of elements 20.

In order to create a concentrated mesh an hyperbolic tangential function has been used.

$$f(x) = \frac{\tanh\left(\gamma\left(\frac{2x}{N} - 1\right)\right) - \tanh\left(\gamma\left(\frac{2(x-1)}{N} - 1\right)\right)}{2\tanh(\gamma)}$$

Where N is the number of elements and γ is the concentration factor to the walls. For a γ value close to zero the mesh is uniform and for a γ value higher, the mesh is more concentrated to the walls.

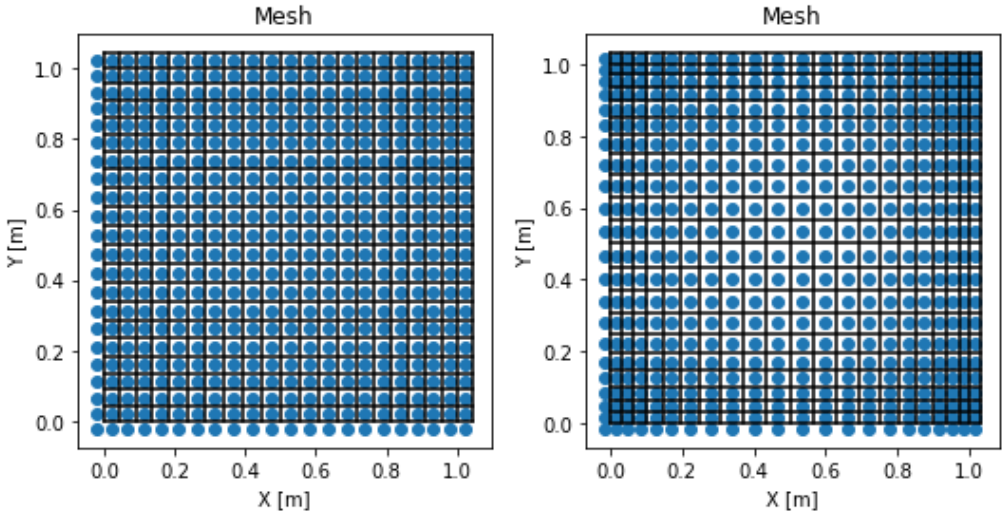

Figure 35: Mesh for concentration factors of $\gamma = 0.5$ and $\gamma = 1$

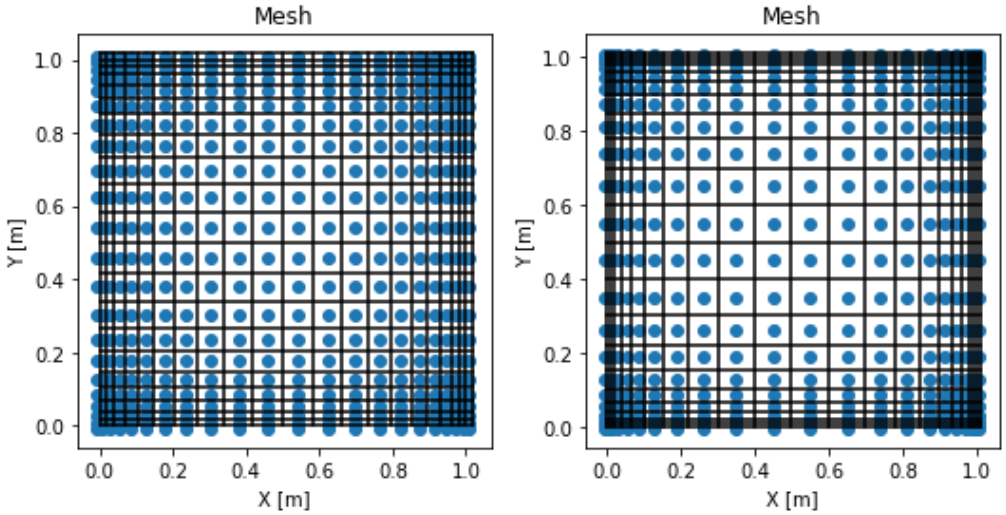

Figure 36: Mesh for concentration factors of $\gamma = 1.5$ and $\gamma = 2$

γ **value**	Δ**y at the wall [m]**
0.00001	0.025
0.5	0.022
1	0.015
1.5	0.009
2	0.004

Table 2: Element size at the wall for different values of γ

Comparing the table 2 with the theoretical results of the element size, it is clear that for a γ value of 1.5 and 2, the mesh will be fine enough to solve the case correctly.

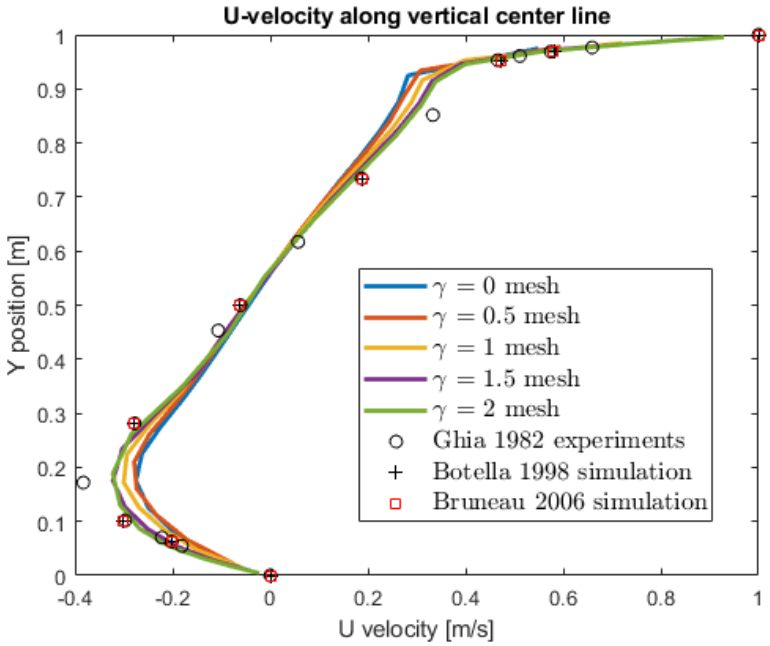

Figure 37: U velocity for different concentration factors [1] [2] [3]

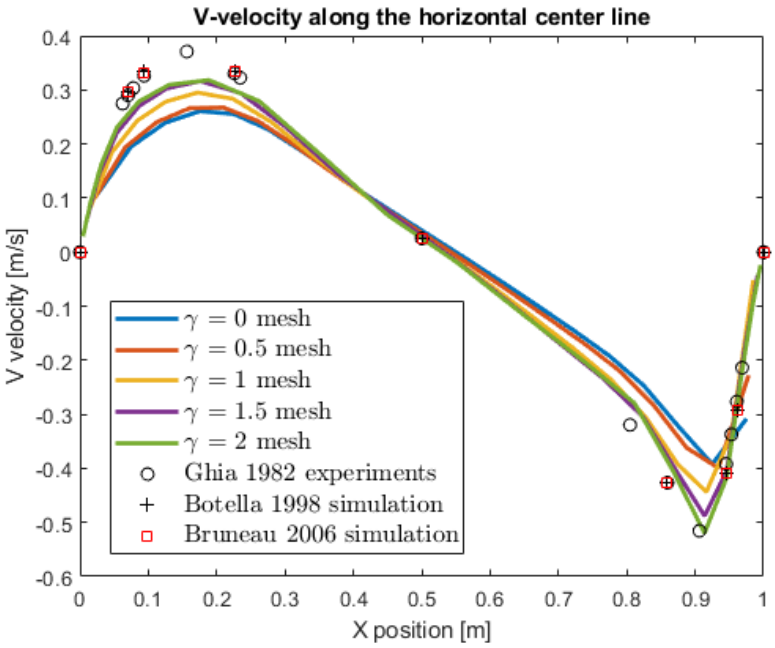

Figure 38: V velocity for different concentration factors [1] [2] [3]

Once the case is solved, it is clear that as the concentration to the walls get higher, the result is closer to the correct value: 37, 38. It is also clear that the number of elements is still a bit small for the number of Reynolds chosen because the case is not well solved. But in this study, the most important conclusion is to observe that as the concentration to the walls gets higher, the flow will be better computed in the wall region.

In the figures 37 and 38 it is clear the point made previously which said that the [1] results are coincident with high number of elements simulations.

8.3 Reynolds number study

Once the mesh study is performed, the case can be solved for different Reynolds numbers: 100, 400 and 1000. The Reynolds used are small because of the computational power that must be used to solve the cases.

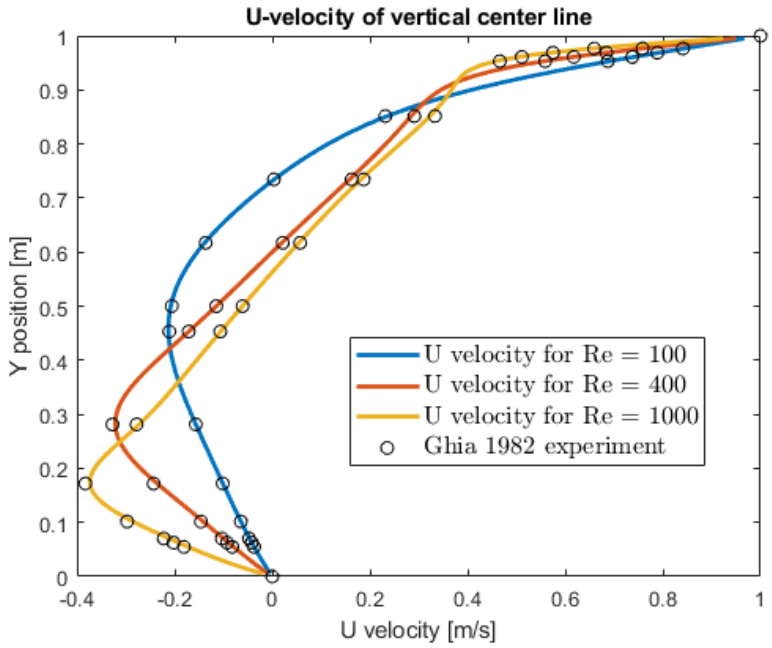

Figure 39: U velocity for different Reynolds number [1]

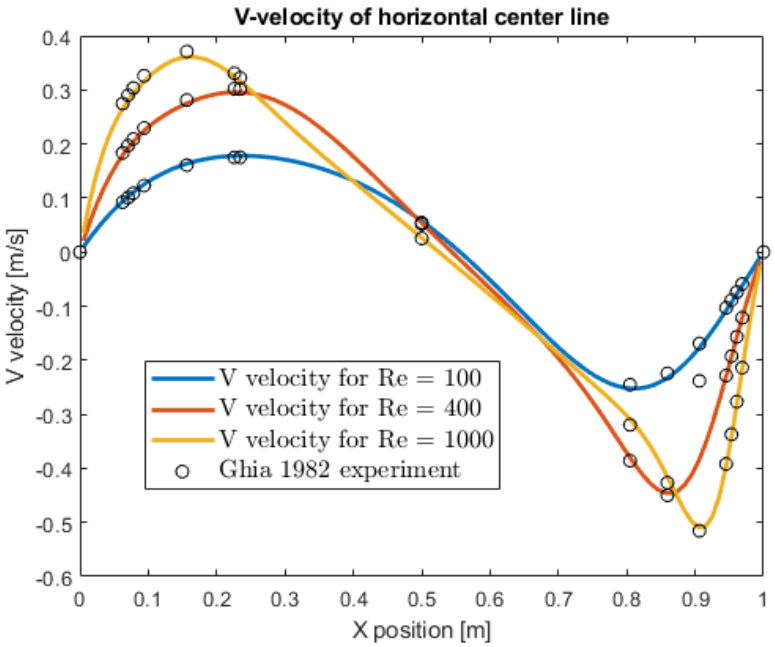

Figure 40: V velocity for different Reynolds number [1]

The figures 39 and 40 show how the velocity will be modified as the

Reynolds number increases. It is clear that the velocity will increase near the walls when the Reynolds number increases.

This study shows the the code implemented is able to compute the case for different numbers of Reynolds. This means that it is able to adapt to the different turbulence values of the flow.

8.4 Streamlines comparison

In this section the streamlines of different articles are going to be compared with the obtained in the study. Two different articles are going to be used for the simulations with fine meshes [4] [5].

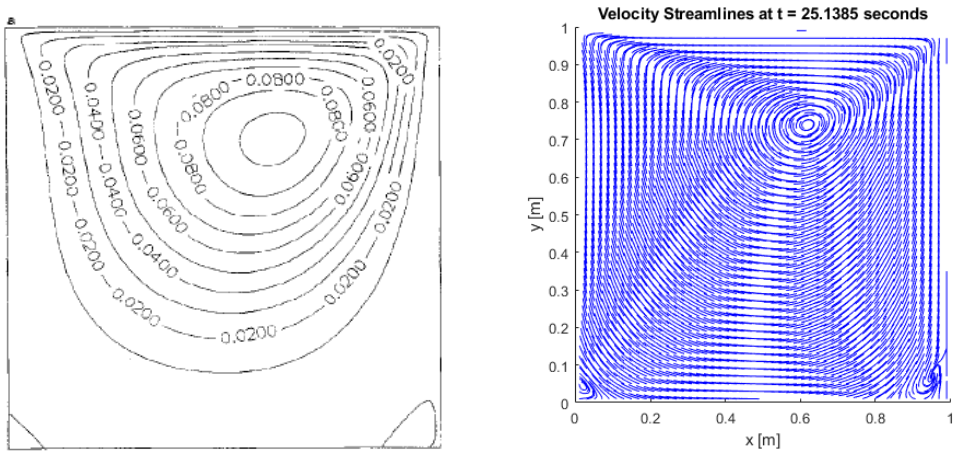

Figure 41: Streamlines comparison for a Reynolds number of Re = 100 [4]

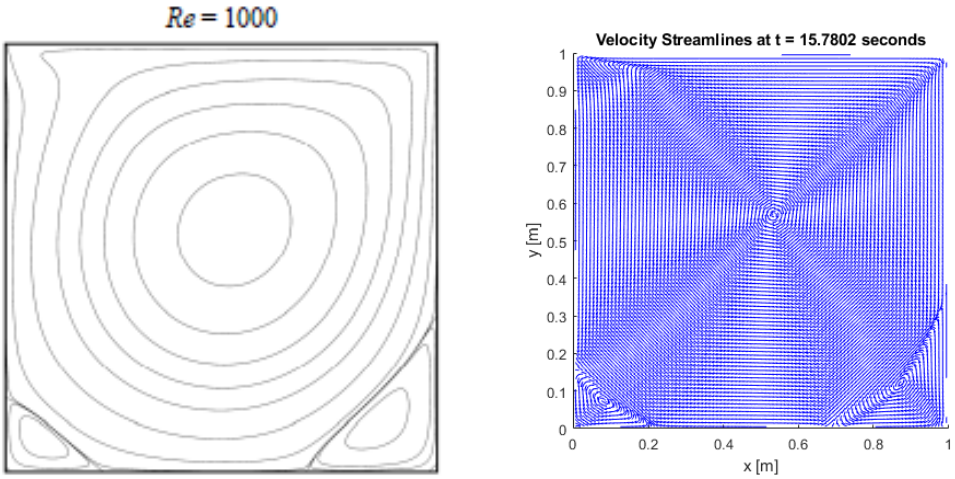

Figure 42: Streamlines comparison for a Reynolds number of Re = 1000 [5]

As seen in the figure 41 and 42 the streamlines of both cases match with the corresponding simulation. Some aspects of the flow can be observed comparing both images.

First of all, as the Reynolds number increases, the main eddy of the flow is located closer to the center of the domain. And it can also be observed that the two eddies at the bottom corners of the domain are greater as the Reynolds number increases. It is obvious that as the flow is more turbulent, those two eddies will stay at a certain size but then more small eddies will appear in both regions.

8.5 Qualitative results

In this section, the picture of the final step of the case for Reynolds number of 1000 is taken to observe the behaviour of the flow in the different regions. This pictures 43 and 44 are very useful to observe the behaviour of the flow as said, but is also very important to observe the transitory state. When the simulation starts, the flow of the domain is at zero velocity, and the different steps saved from the simulation can show its evolution to the steady flow velocity domain.

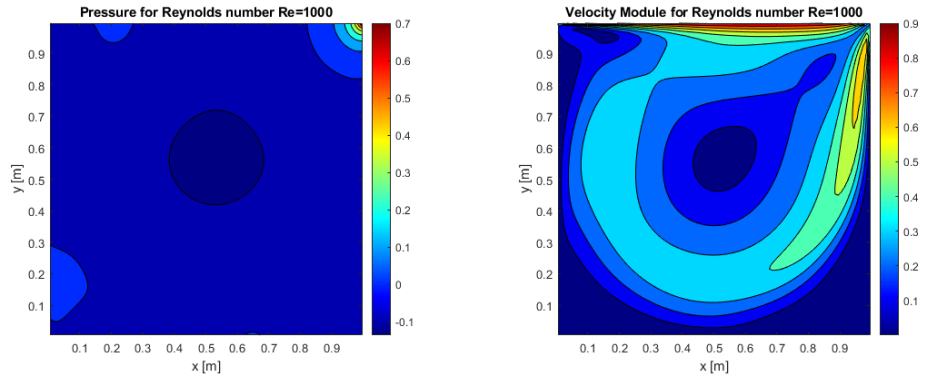

Figure 43: Pressure and velocity module for Reynolds = 1000

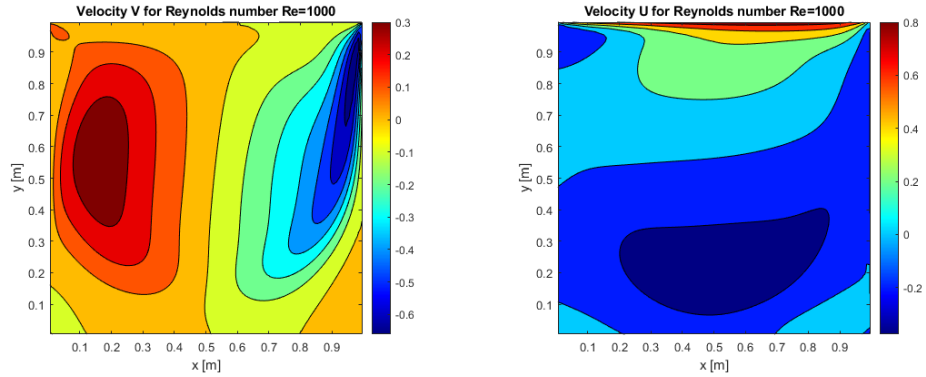

Figure 44: Horizontal and vertical velocity for Reynolds = 1000

9 Performance study

In this section a brief study of the code performance is done. The computational effort of the code is very important in order to be optimized for future cases.

In this study, the main function to be studied is the one that implements the time step integration because a single simulation can have a huge amount of steps so each calculation that is placed inside it will be performed so many times.

So in order to obtain the computational effort, the time that each part of the function will be obtained. Then, the most critical parts have to be analysed.

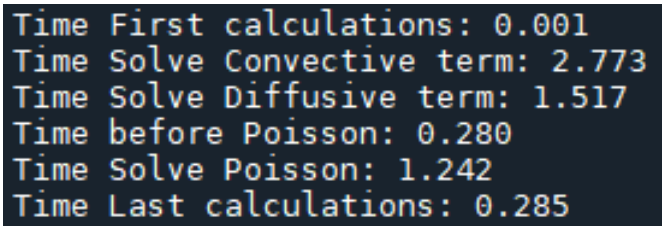

Figure 45: Picture obtained from the command window

In order to perform the study, a driven cavity case is run with a uniform mesh of 200x200. In the figure 45 the time that each part of the step last is observed.

Part of the step	**Computational time [seconds]**	**Percentage of time [%]**
First calculations	0.001	0.02
Convective term function	2.773	45.5
Diffusive term function	1.517	21.7
Calculations between Diffusive and Poisson	0.280	4.6
Poisson function	1.242	20.4
Last calculations	0.285	4.7

Table 3: Mesh size and number of elements for each Reynolds number

As seen in the table 3, it is easy to conclude that the calculation of the convective and diffusive terms and the calculation of the pseudo-pressure consume almost all the time. The three mentioned functions consume the 87.6 % of the time. Those three functions are the ones to be optimized for future cases then.

It is also clear that the first function to be optimized is the one that computes the convective term as it consumes almost half of the time.

10 Future improvements

In this section some improvements of the code are explained. It is clear that the code developed is used for educational purpose and it can solve simple cases. As each step must be understood and validated, the code obtained is used for incompressible two dimensional cases. For this reason, this section shows some improvements that can be made in order to have a more complex code.

10.1 Three dimension domain

It is clear that the code of two dimension flow is not exactly correct as the turbulence eddies are developed in the three dimensions. For this reason, if it is desired a more realistic code, the third dimension must be implemented. In order to obtain this, the following modifications must be performed.

It is clear that a third direction for the velocity must be set at the momentum conservation equation. As a consequence, the building of the matrix A will need a third dimension to compute the pseudo pressure. This will also modify the explained convective and diffusive terms calculation functions.

The change of the code will also consist on applying the halo of the new two boundaries (front and back) or setting the boundary conditions at those boundaries.

10.2 Neumann boundary conditions

In order to solve more cases, an outlet must be set by the Neumann boundary conditions. The Neumann boundary conditions consist on imposing a certain value of the derivative normal to the boundary. For the case of fluid dynamics, this means that the gradient of the velocity and pressure is set to zero.

The imposing of the derivative normal to the wall will have a similar effect as the Dirichlet conditions explained on previously. The velocities of the halo must be modified in order to achieve the desired derivative value (zero for an outlet) previous to the cenvective and diffusive terms calculation.

10.3 Suspended object

In order to solve a case with a suspended object, it is clear that the domain must have an inlet and an outlet [15]. Then, the top and bottom borders can be set as a periodic boundaries or a wall very far from the object that is studied .

Then, a map like the one in the figure 46 must be set to make the code distinguish between the object and the flow. This example shows that a uniform mesh of 200X200 does not have a good precision when discretizing the geometry of a certain profile.

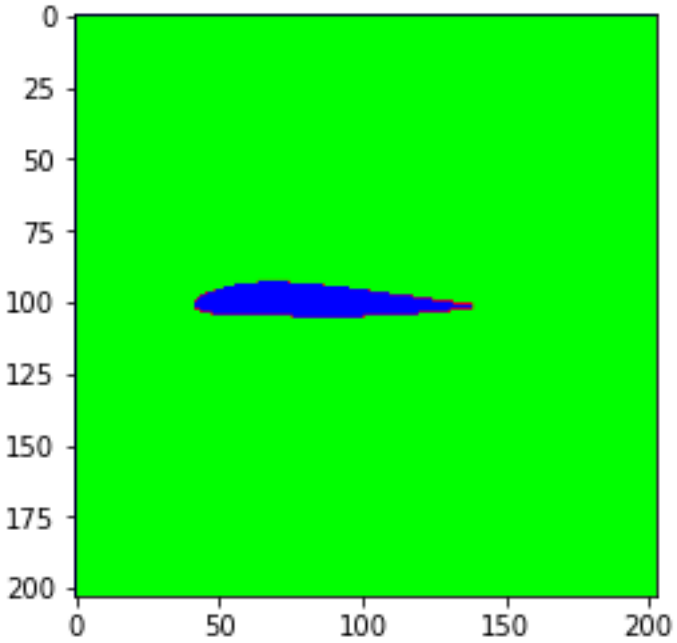

Figure 46: Map of the airfoil

Once this is done, the Dirichlet conditions of zero velocity must be set to the object. It is clear that for this case, the mesh must be concentrated around the object in order to have more precise results.

It is clear that the shape of the airfoil is obtained from the internet. The example of of the figure 46 corresponds to a NACA25112 airfoil.

10.4 Higher number of elements simulations

If the code is modified to simulate more complex cases like 3D studies or turbulent cases, the first improvement to be done is an optimisation of the code. It is important to prepare it for managing a big amount of data.

It will also be important to consider improving the simulation time. To do so, for high number of elements, an iterative method can be used to solve the Poisson equation. An example of this could be the Gauss-Seidel method.

The other functions that have a huge computational effort are the ones that compute the convective and diffusive terms. In this case, the functions must also be optimized in order to give a good performance.

11 Conclusions

To conclude, different topics must be commented. First of all, the requirements set at the beginning of the study must be explained. It must be said if the requirement is reached and why.

The first requirement talks about understanding the Navier-Stokes equations and the procedure to implement the fractional step method. It must be said that the equations have been understood and implemented correctly. The fractional step method is very useful for educational purposes as it is very simple and can be implemented easily.

Talking now about the validation of the code, it is very important to mention the three different studies performed. Each study will be used to validate different functions so once this is done, it can be said that the corresponding function can be used. As explained, the Part A validates the functions that compute the convective and diffusive terms. They have been validated correctly so the are not going to be modified at least during this study. If a three dimensional domain is implemented, they should be validated again as they should be changed. The functions that solve the Poisson equation are validated in the Part B. This part is divided by three studies depending on the boundary conditions of the domain. And after that, is clearly seen that the code can compute all three cases. With that, in function of the study that wants to be done, each one of the different codes mentioned must be set. And at last, the function that implements the time integration is validate correctly for a periodic case. With the objectives reached, it is possible to run a simple case with periodic boundaries. With this, the first part of the thesis is completed satisfactorily and is possible to continue with the implementation of the boundary conditions. The second main requirement is to implement and validate a simple case using boundary conditions, the driven cavity. The results show that this case has been implemented correctly. So it can be said that the objective has been reached.

11.1 Results discussion

The results explained in the validation part of the thesis shows that the results are correct for a periodic case. As can be seen in the Part C of the validation, the flow is very well computed comparing with the analytical solution. It is clear that for a periodic case where an initial velocity field is imposed, its magnitude will be decreased as the time increases. This is be-

cause there is no external force or pressure that accelerates the flow. This is the reason why the velocity tends to a zero value.

For the driven cavity case, the boundary conditions imposed produce an external motion that will make the flow move along time. Then, after the initial transient period, the flow reaches a stationary movement. At this point the solution converges and the results are obtained. As can be seen, the driven cavity case show very good results for Reynolds number of Re = 100, Re = 400 and Re = 1000. It is clear that if the Reynolds number gets higher, the simulation will require more computational effort to be done. This is because the flow gets more turbulent so more eddies appear in the domain. More elements will be required and close to the walls they will have to be smaller to compute well their influence.

11.2 Continuation of the study

As commented before, some improvements can be performed to obtain a code able to solve more complex cases. However, the improvements are focused on the method used in the code itself. If the thesis is continued, some other paths could be explored. For example, it could be studied if the method used is the most optimal. It could also be set a very detailed document with the steps to develop the code from zero.

However, as explained and following the method already used in this thesis, the code could be improved by different paths. The three dimension domain would be the first step to perform because it could solve a case with all king of geometries. After that, the Neumann boundary conditions could be implemented in order to have an outlet at the domain. After these improvements, a suspended object can be studied which is very important because it can be very useful for the aerospace sector. To solve more complex cases a more optimal code should be used to manage the big amount of data and an iterative method to solve the Poisson equation.

References

[1] U. GHIA K. N. GHIA and C. T. SHIN. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *JOURNAL OF COMPUTATIONAL PHYSICS*, 48:387–411, 1982.

[2] Mazen Saad Charles-Henri Bruneau. The 2d lid-driven cavity problem revisited. *Computers & Fluids*, 35:326–348, 2006.

[3] R. PEYRET O. BOTELLA. Benchmark spectral results on the lid-driven cavity flow. *Computers & Fluids*, 27(4):421–433, 1998.

[4] S. P. VANKA. Block-implicit multigrid solution of navier-stokes equations in primitive variables. *JOURNAL OF COMPUTATIONAL PHYSICS*, 65:138–158, 1986.

[5] K. Yapici and Y. Uludag. Finite volume simulation of 2-d steady square lid driven cavity flow at high reynolds numbers. *Brazilian Journal of Chemical Engineering*, 30(4):923–937, 2013.

[6] Lajos T. Vad J. and Schilling R. *Modelling Fluid Flow*. Springer, 2004.

[7] Manel Soria Guerrero. Numerical solution of the incompressible ns equations, February 2021.

[8] C. Foias, O. Manley, R. Rosa, and R.Temam. *Navier-Stokes equations and turbulence*. Cambridge university press, 2004.

[9] R. Aris. *Vectors, Tensors, and the basic Equations of Fluid Mechanics*. Dover Publications, 1989.

[10] George B. Arfken. *Mathematical Methods for Physicists 3rd ed.* Orlando, FL: Academic Press, 1985.

[11] SAJAL K. KAR. An explicit time-difference scheme with an adams–bashforth predictor and a trapezoidal corrector. *I. M. Systems Group, Inc., Rockville, Maryland*, 140:307–322, 2012.

[12] O. A. Ladyzhenskaya A. Krzywicki. A grid method for the navier-stokes equations. *Soviet Physics Dokl.*, 11:212–213, 1966.

[13] R. R. Rosales D. Shirokoff. An effcient method for the incompressible navier-stokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary., 2013.

[14] Marc Andrés Carcasona. Study: Simulation of planetary atmospheres, 2020.

[15] Steven Allmaras, Venkat Venkatakrishnan, and Forrester Johnson. Farfield boundary conditions for 2-d airfoils. *43th AIAA Aerospace Sciences Meeting, AIAA.*, 2005.