Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Luís Henrique Carnevale da Cunha

ALE FINITE ELEMENT METHOD FOR SIMULATING FLOWS WITH THE STREAMFUNCTION-VORTICITY FORMULATION

Rio de Janeiro, Brazil
2020

Luís Henrique Carnevale da Cunha

ALE FINITE ELEMENT METHOD FOR SIMULATING FLOWS WITH THE STREAMFUNCTION-VORTICITY FORMULATION

Master's Thesis presented to the Mechanical Engineering Graduate Program of the Universidade do Estado do Rio de Janeiro as a partial requirement to obtain the degree of Master in Sciences. Field of concentration: Transport Phenomena.

Advisor: Prof. Gustavo Rabello dos Anjos, Ph.D

Co-Advisor: Prof. Norberto Mangiavacchi, Ph.D

Rio de Janeiro, Brazil

2020

Página da Ficha Catalográfica:

A biblioteca deverá providenciar a ficha catalográfica. Salve a ficha no formato PDF.

Substitua esse arquivo ***Ficha.pdf*** na pasta ***B.PreTextual*** pelo pdf da sua ficha catalográfica enviado pela biblioteca.

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

_____ _____

Assinatura Data

Luís Henrique Carnevale da Cunha

ALE FINITE ELEMENT METHOD FOR SIMULATING FLOWS WITH THE STREAMFUNCTION-VORTICITY FORMULATION

Master's Thesis presented to the Mechanical Engineering Graduate Program of the Universidade do Estado do Rio de Janeiro as a partial requirement to obtain the degree of Master in Sciences. Field of concentration: Transport Phenomena.

Approved on December 15th, 2020.

Examining Committee:

Prof. Gustavo Rabello dos Anjos, Ph.D (Advisor)
COPPE/UFRJ

Prof. Norberto Mangiavacchi, Ph.D (Co-Advisor)
PPG-EM UERJ

Prof. Cristiane Oliveira de Faria, D.Sc.
PPG-EM UERJ

Prof. Álvaro L.G.A. Coutinho, D.Sc.
COPPE/UFRJ

Prof. Panagiotis Theodorakis, Ph.D
IFPAN

Rio de Janeiro, Brazil

2020

Luís Henrique Carnevale da Cunha

ALE FINITE ELEMENT METHOD FOR SIMULATING FLOWS WITH THE STREAMFUNCTION-VORTICITY FORMULATION

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências, ao Programa de Pós-Graduação em Engenharia Mecânica da Universidade do Estado do Rio de Janeiro. Área da concentração: Fenômenos de Transporte

Aprovada em 15 de dezembro de 2020.

Banca Examinadora:

Prof. Gustavo Rabello dos Anjos, Ph.D (Advisor)
COPPE/UFRJ

Prof. Norberto Mangiavacchi, Ph.D (Co-Advisor)
PPG-EM UERJ

Prof. Cristiane Oliveira de Faria, D.Sc.
PPG-EM UERJ

Prof. Álvaro L.G.A. Coutinho, D.Sc.
COPPE/UFRJ

Prof. Panagiotis Theodorakis, Ph.D
IFPAN

Rio de Janeiro, Brasil

2020

ABSTRACT

CARNEVALE DA CUNHA, Luís Henrique. *ALE Finite Element Method for simulating flows with the streamfunction-vorticity formulation.* 2020. 91 f. Master's Thesis (Master in Mechanical Engineering) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil, 2020.

Fluid Dynamics is an important area of research in many fields of science and engineering. It is present in the study of aerodynamics, heat exchanges, meteorology and even in biological systems. Due to the complex behavior of flow in different circumstances, being able to accurately describe the fluid motion is a challenge. The purpose of this dissertation is to implement a Finite Elements algorithm capable of simulating one phase two-dimensional flow using movable computational mesh grids. The mathematical model of fluid flow used was the stream function-vorticity formulation which is a substitute for the Navier-Stokes equation and is written for the Arbitrary Lagrangian-Eulerian referential to accommodate the mesh movement. A first order semi-Lagrangian method was used to approximate the advective term found in the formulation. This is an unconditionally stable method which allows the algorithm to take larger time steps. A third party open source software called GMSH was used to generate the computational mesh with linear triangular elements. To avoid large distortions, a Laplacian smoothing technique was implemented to help maintain the mesh quality during its movement. Code development was based on Python script language where highly demanding methods were implemented in its compiling version namely Cython. Several test cases were successfully used as benchmarks to verify and validate the proposed new methodology including the cavity driven and Poiseuille flows. Additionally, the ow past an object was chosen as study case for several interesting geometries including the classical cylinder.

Keywords: Finite Element Method; Arbitrary Lagrangian-Eulerian; Semi-Lagrangian Method; Stream Function-Vorticity Formulation.

RESUMO

CARNEVALE DA CUNHA, Luís Henrique. *Método de Elementos Finitos ALE para Simulação de Escoamentos Utilizando a Formulação Corrente-Vorticidade*. 2020. 91 f. Master's Thesis (Master in Mechanical Engineering) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil, 2020.

Dinâmica de fluidos é um importante tópico de pesquisa em diversas áreas da ciência e engenharia. Ela está presente no estudo de aerodinâmica, trocadores de calor, meteorologia, e até mesmo em sistemas biológicos. Devido ao comportamento complexo de escoamentos sob diferentes circunstancias, descrever o movimento de um fluido com suficiente acurácia é um desafio. A proposta dessa dissertação é implementar um algoritmo capaz de simular escoamentos monofásicos em duas dimensões através do método de Elementos Finitos e usando uma malha computacional móvel. O modelo matemático utilizado é uma alternativa às equações de Navier-Stokes, conhecido como formulação corrente-vorticidade, que foi escrita no referencial arbitrário Lagrangiano-Euleriano para o movimento de malha. A discretização do termo advectivo da formulação foi feita através da aplicação de um método semi-Lagrangiano de primeira ordem, que é incondicionalmente estável e com isso permite o uso de maiores passos no tempo. O software open source GMSH foi usado para gerar as malhas computacionais com uma triangulação de elementos lineares. Para evitar grandes distorções, uma técnica de suavização Laplaciana foi implementada para ajudar a manter uma boa qualidade de malha durante o movimento. O código foi escrito principalmente com a linguagem de programação Python e algumas funções foram otimizadas usando Cython. O problema clássico de escoamento em torno de um cilindro foi escolhido como um caso de estudo, primeiro para uma posição fixa e depois para um movimento oscilatório com outras geometrias.

Palavras-chave: Método de Elementos Finitos; Lagrangiano-Euleriano Arbitrário; Método Semi-Lagrangiano; Formulação Função Corrente-Vorticidade.

LIST OF FIGURES

LIST OF TABLES

SUMMARY

INTRODUCTION

Fluid Dynamics is an important are of research in many fields of science and engineering. It is present in the study of aerodynamics, heat exchanges, meteorology and even in biological systems. Due to the complex behavior of flow in different circumstances, being able to accurately describe the fluid motion is a challenge and the main equations that model this motion, namely the Navier-Stokes equations, are still an unsolved problem.

Since the advent of computer machines, an array of methods started being developed to approximate the solution of mathematical models using computational algorithms however, there are different numerical methods, each with its advantages and drawbacks. In this context emerged the field of Computational Fluid Dynamics (CFD) in an attempt to understand and solve problems regarding fluid flow.

Nowadays there are two main methods found in most CFD commercial softwares: the Finite Volume Method (FVM) and the Finite Elements Method (FEM). The purpose of this dissertation is to implement a FEM algorithm capable of simulating single-phase two-dimensional flows using moving computational mesh nodes. Moreover, the mathematical model of fluid flow used was the stream function-vorticity formulation which is a substitute for the classical Navier-Stokes equation where primitive variables are used. This formulation is written for the Arbitrary Lagrangian-Eulerian referential to accommodate the mesh motion.

In dealing with unsteady fluid simulations, numerical instabilities may spoil the accuracy and validity of the solutions and this problem is accentuated for higher Reynolds number, which is a relation between inertial and viscous forces in the flow. To counter act this effect, a first order semi-Lagrangian method was used to approximate the advective term found in the formulation. This is an unconditionally stable method which allows the algorithm to take larger time steps, however it has a numerical diffusion problem as its drawback.

The unstructured computational mesh used in the this work consisted only of linear triangular elements and was generated by a third party open source software called GMSH. For the mesh motion, a few restrictions need to be observed to prevent large distortions in the elements or their overlapping which would introduce too much error in the simulation. To avoid having such a problem, a Laplacian smoothing technique was implemented to help maintain a good mesh quality during its movement.

Code implementation was done by using mainly Python as its programming language. Python offers a great number of well established packages, such as Scipy and Numpy, that deal with mathematical operations and numerical methods. It also has a package called

Matplotlib, used to plot some results and the package Meshio that is able to read the mesh files from GMSH. The chosen language still had a problem of efficiency when calling the implemented methods and to address this issue, the most computational demanding functions were implemented on Cython.

As the main goal of this work is to computationally solve flow problems with a moving mesh, a classical problem of fluid-structure interaction was chosen as a study case. First the flow around a fixed cylinder is investigated for different Reynolds numbers and compared to results found in the literature, then the simulations for an oscillating cylinder, square and triangle are presented.

This dissertation is organized as follows:

- Chapter 1: Literature review of relevant papers and books used as reference for this work;
- Chapter 2: A deduction of the main equations that govern fluid flow in a continuum medium;
- Chapter 3: Presents the Finite Elements Method discretization of the mathematical model;
- Chapter 4: Describes the algorithm and methodology used;
- Chapter 5: Shows a verification of the implemented code and the simulations for a moving mesh.
- Conclusion.

1 LITERATURE REVIEW

1.1 Numerical Method

Many physical phenomena can be described by a mathematical model from which it is possible to make predictions and study the problem at hand. In more complex physical systems the dynamics are modeled by partial differential equations (PDE's) and it is not always possible to obtain an analytical solution for those equations. Numerical methods were developed to solve a discrete version of the equations with the aid of computational algorithms and find an approximate solution. The three main numerical methods are:

- Finite Differences;
- Finite Volumes;
- Finite Elements.

The Finite Differences Method consists of trying to find approximated schemes for the derivatives of a function in a differential equation by using a Taylor series expansion of those terms and then rearranging the PDE's as a linear system of equations. The simplest example of approximation is the forward difference: Let $f(x) : \mathbb{R} \rightarrow \mathbb{R}$ be a scalar function, its Taylor series around the point $x + \Delta x$ is

$$f(x+\Delta x) = f(x) + (\Delta x)\frac{df(x)}{dx} + \frac{(\Delta x)^2}{2!}\frac{d^2f(x)}{dx^2} + \frac{(\Delta x)^3}{3!}\frac{d^3f(x)}{dx^3} + \cdots, \tag{1}$$

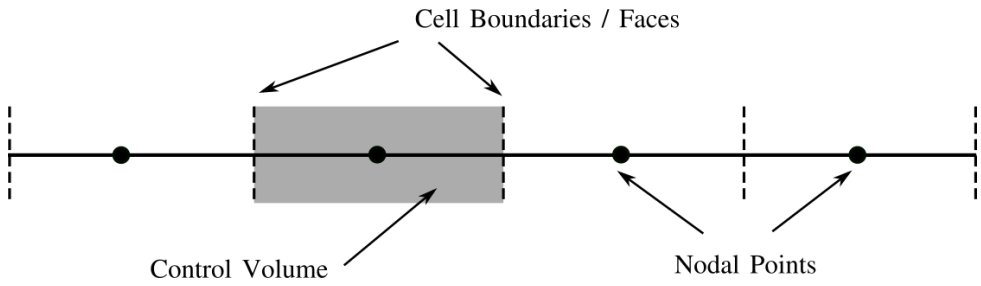
rearranging the terms to isolate the first derivative and representing the sum of all the terms with factors Δx and higher by $O(\Delta x)$ (meaning of order Δx),

$$\frac{df(x)}{dx} = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x). \tag{2}$$

Graphically, Eq. 2 is approximating the first derivative of $f(x)$ at a point x by the slope of the function between the points x and $x + \Delta x$. The error of this approximation is of order $O(\Delta x)$ and decreases linearly as Δx decreases (HOFFMANN; CHIANG, 2000).

One of the first instances of this method being employed in solving unsteady incompressible flows with the stream function-vorticity formulation is discussed by Fromm and Harlow (1963) in which they studied the development of a vortex street behind a rectangular plate for Reynolds number range $15 < Re < 6000$.

Figure 1 - One dimensional domain discretized in control volumes used in the Finite Volume Method.



The Finite Volume Method consist of finding a discretized equation by integrating the differential equations over what is called a control volume, or mesh cell. These cells are generated by distributing the discrete nodal points over the domain and then positioning the boundaries (or faces) of the cells between adjacent nodal points, so that each node is surrounded by a control volume as illustrated in Fig. 1 for a one dimensional grid.

The discretized equation is solved for the variables defined and evaluated at the nodal points by balancing the flux of those quantities that go through the cell boundaries. And this flux is approximated by the nodal values of neighbor control volumes. This is a great advantage of the method because the resulting expressions guarantee local conservation of relevant physical properties (VERSTEEG; MALALASEKRA, 2007).

The choice of numerical method for this dissertation is the Finite Elements Method. According to Fish (2007) the FEM was developed during the 1950s, although in 1943, Courant published a paper in which he used a variational formulation to solve a problem over a domain discretized in triangular elements (COURANT, 1943). In 1956, a paper by Turner et al. (1956) was published discussing the foundations for the method.

To solve a differential equation with the FEM the problem is stated in its variational form, in which a weak solution is found for a defined function space. The domain of interest is divided in a finite number of elements connected by nodal points where the equations are integrated for a finite number of basis functions chosen from the defined space (HUGHES, 2000). More details of the method and its application on this dissertation are presented in Chapter 3.

Even though the FVM being largely found in commercial CFD softwares, there is still a debate in which method is more suitable for solving flow problems. Idelsohn and Oñate (1994) argues that FVM is a good choice for convective dominant equations and that FEM is a more general and mathematically consistent approach, it also shows a case in which both methods are equivalent.

1.2 **Stream Function-Vorticity Formulation**

The stream function-vorticity formulation is an alternative way of writing the Navier-Stokes equations, which describes fluid motion in a continuum medium. The formulation characterizes the flow by using the stream function ψ and the vorticity ω as its variables instead of using the primitive variables velocity and pressure (PEETERS; HABASHIT; DUECK, 1987). A strong advantage of using this formulation is the removal of the pressure-velocity coupling problem that requires a special treatment in a FEM implementation. However, this approach is best suited for two dimensional models since 3D simulations require the solution of a larger number of equations than solving the primitive variables directly.

Some examples of the stream function-vorticity formulation being solved with the FEM can be found in the literature. In Cesini et al. (1998) it is used to study the natural convection around a heated cylinder inside a rectangular cavity and compared the results with experimental observations. Tezduyar, Glowinski and Liou (1988) uses the FEM to solve the stream function-vorticity formulation on multiply connected domains and addresses some difficulties associated with the presence of the convective term in the vorticity transport equation by employing a streamline-upwind/Petrov-Galerkin procedure to minimize numerical oscillations. In Tezduyar and Liou (1991), the authors also discuss about the difficulties in defining downstream boundary conditions for the vorticity and show results for their proposed procedure for the flow around a cylinder case.

A drawback in using this approach is the lack of boundary conditions for the vorticity at no-slip boundaries. This is usually solved by calculating the vorticity value from the stream function solution using a finite difference scheme as shown by Vynnycky et al. (1998) in studying conjugated heat transfer problems with forced convection. The no-slip boundary condition can also be estimated by using a FEM approximation as done by Comini, Cortella and Manzan (1995), which used the formulation in problems of mixed, forced and natural convection of laminar flow. In this dissertation it is proposed an estimation of the no-slip boundary condition by calculating a Finite Element approximation of the vorticity definition form the velocity field.

The stream function-vorticity formulation can also be solved with different methods as shown by Abdellatif, Touihri and Amin (2016) using a spectral element discretization of the equations written in axisymmetric coordinates for a Stokes flow.

1.3 **Arbitrary Lagrangian-Eulerian**

The Arbitrary Lagrangian-Eulerian method (ALE) is an important tool in simulating a wide array of problems in fluid dynamics where boundary movement needs to be accounted for. Its main feature is the possibility of describing the fluid motion in an arbitrary frame of reference which allows the computational mesh to move in an arbitrary manner.

Problems regarding fluid-structure interactions are an example of applicability of the ALE method. In the paper from Donea, Giuliani and Halleux (1982) they use an ALE kinematical description of the fluid domain which leads to an accurate treatment of the fluid-structure interface and permits significant fluid movement without producing excessive distortion on the computational mesh. In the paper, they study non-linear response of fluid-structure systems exposed to transient dynamic loading.

This method can also be used in solving problems that need a sharp numerical representation of an interface, such as in two phase flow simulations. An ALE/FE method has been used to simulate two-phase flows by Anjos et al. (2014) where the interface between fluids was subjected to large spatial deformations. To accurately address such a change in topology, an adaptive remeshing technique was proposed along with the Laplacian Smoothing operator to regularize the finite element mesh. Additionally, the Semi-Lagrangian method was adopted to discretize the convective operator, allowing the simulation of high velocity flows and avoiding spurious numerical oscillations due to the discretization of such an operator. Several test cases were used to demonstrate the accuracy of the ALE/FE combined with the SL methods.

As presented by Hughes, Liu and Zimmermann (1981), an ALE/FE method is implemented for a two-dimensional 4-node quadrilateral element and used it to solve the Navier-Stokes equation in primitive variables to study a free-surface wave propagation problem. However they argue the method has an increased computational cost when compared to to a purely Eulerian case.

All the aforementioned papers use the Navier-Stokes equations with the primitive velocity and pressure variables, however not many papers can be found with an alternative formulation written in the ALE framework being used. Lo and Young (2014) discuss the use of an ALE/FE velocity-vorticity formulation in studying free-surface flow, where a boundary-fitted coordinate system is employed to solve boundary equations for kinematic and dynamic conditions at the free surface using a finite difference method.

1.4 **Vortex Shedding**

The classical problem of vortex formation in the wake of a cylinder is used in this dissertation to demonstrate the method developed, first for a fixed cylinder and then for different moving geometries. This is a complex problem as different flow regimes can occur depending on the Reynolds number, defined in Eq. 3.

$$Re = \frac{UD}{\nu} \tag{3}$$

where U is a reference velocity, D a characteristic length, in this case the diameter, and ν the fluid's kinematic viscosity.

For the fixed cylinder case, when $Re < 47$, a stationary solution is observed with the formation of two recirculation zones attached to the cylinder in its wake. For $47 < Re < 180$, vortices start being shed periodically in a staggered alternate manner in what is known as Von Kármán streets. The frequency in which they are formed is known as Strouhal frequency and is associated with the non-dimensional Strouhal number, defined in Eq. 4 which increases with an increase in the Reynolds number. For $180 < Re < 230$ the flow goes through a wake transition phase caused by instability modes where there's a discontinuity in the relation between the Strouhal and Reynolds number. Other flow regimes can also be observed for $Re > 360$ characterized by different instability modes (FEY; KöNIG; ECKELMANN, 1998).

$$St = f_s \frac{d}{U} \tag{4}$$

where f_s is the Strouhal frequency and d is the cylinder diameter.

When the cylinder is put into forced oscillation, the vortex shedding frequency diverges from the fixed case, and for a certain range of amplitudes this becomes equal to the oscillation frequency. Placzek, Sigrist and Hamdouni (2009) studies this phenomena with an ALE finite volume method by simulating various ranges of frequency and amplitude of oscillation and they also carry out simulations of vortex-induced vibrations, where the cylinder movement is directly induced by the vortex shedding process in the wake.

Nguyen (2010) uses an ALE discontinuous Galerkin method to simulate flows over variable geometries, including deformable domains or moving boundaries. A mesh smoothing technique is also developed based on element size functions to handle large grid deformations. This approach was used to study the laminar flow over a forced oscillating cylinder and also for the flow over a flapping elliptical wing.

An experimental study with cylinder forced oscillations made by Williamson and Roshko (1988) was able to observe various modes of synchronization between the vortex shedding frequency and the oscillation frequency for different amplitudes.

2 MATHEMATICAL FORMULATION

This chapter presents the main differential equations used to model the fluid flow in their traditional description. The derivation of those equations stem from the continuum hypothesis which establishes that the fluid medium is homogeneous in its smallest region of space dV. By applying the conservation of mass and momentum to a control volume dV we get the equations:

- Continuity equation
- Momentum equation
- Stream function-Vorticity equation

Afterwards, the Arbitrary Lagrangian-Eulerian description is introduced and the equations are then written in the ALE reference frame. In the end of this chapter the non-dimensional form of the equations are also presented.

2.1 Continuity Equation

The continuity equation comes from the conservation of mass and establishes the rate of change in time of fluid's density ρ on a given point in space. Assuming the volume dV has a mass $dm = \rho dV$, and the flow is described by the velocity field $\mathbf{v} = (v_x,\ v_y)$, the mass flux that passes through the control volume are represented in Fig. 2.

Let the rate of accumulation of mass inside dV be $\Delta\dot{m}$ and the mass flux be $\dot{m}$, conservation of mass states:

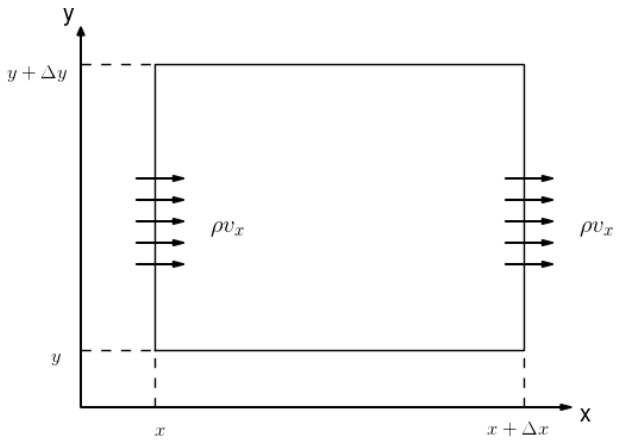
$$\Delta\dot{m} = -\dot{m}. \tag{5}$$

By integrating over the volume, the mass rate of change can be expressed as:

$$\int_V \frac{\partial}{\partial t} dm = \int_V \frac{\partial}{\partial t}(\rho dV) = \int_V \frac{\partial \rho}{\partial t} dV. \tag{6}$$

The mass flux that goes through dV only considers the velocity component that is

Figure 2 - Two dimensional representation of a control volume dV with a mass flux going through its boundaries in the x direction



normal to the boundary. If **n** is the outwards unit vector perpendicular to the boundary, then the total mass flux is:

$$\oint_A \rho \mathbf{v} \cdot \mathbf{n} dA. \tag{7}$$

Eq. 7 is a surface integral that can be rewritten as a volume integral by using Gauss' Theorem, thus:

$$\oint_A \rho \mathbf{v} \cdot \mathbf{n} dA = \int_V \nabla \cdot (\rho \mathbf{v}) dV. \tag{8}$$

From Eq. 5, Eq. 6 and Eq. 8 we obtain the continuity equation in its integral form:

$$\int_V \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) \right] dV = 0. \tag{9}$$

Eq. 9 has to be valid for any control volume, therefore we can write it in its differential form as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{10}$$

The continuity equation can be further simplified by using the identity $\nabla \cdot (\rho \mathbf{v}) = \mathbf{v} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{v}$, and for an incompressible fluid, i. e. the density does not depend on the time variable, therefore the equation is reduced to:

$$\nabla \cdot \mathbf{v} = 0 \tag{11}$$

2.2 **Momentum Equation**

The Navier Stokes equations first appeared by the middle of 19^{th} century and are the most famous equations in fluid mechanics. To this day, different fields of research from applied sciences tor pure mathematics are constantly working with these equations. Its derivation follows from the conservation of linear momentum, which represents a balance of forces acting on the control volume dV. Let $\dot{l}$ be the rate of change in linear momentum, and F_S, F_B are surface and body forces respectively, the conservation of momentum states:

$$\Delta \dot{l} = -\dot{l} + F_S + F_B \tag{12}$$

The rate of accumulation and the flux of momentum through the boundary are mathematically described in a way similarly to the terms in the continuity equation and are presented in Eq. 13 and Eq. 14 respectively.

$$\int_V \frac{\partial}{\partial t}(\rho \mathbf{v}) dV \tag{13}$$

$$\oint_A \rho \mathbf{v}\mathbf{v} \cdot \mathbf{n} dA \tag{14}$$

According to Pontes and Mangiavacchi (2010), the sum of external forces acting on the volume surface can be represented by Eq. 15, where $\boldsymbol{\sigma}$ is a stress tensor. The only body

force considered comes from the gravitational field **g** and is shown in Eq. 16.

$$\oint_A \sigma \mathbf{n} dA \tag{15}$$

$$\int_V \rho \mathbf{g} dV \tag{16}$$

From the conservation of linear momentum and Eqs. 13, 14, 15 and Eq. 16:

$$\int_V \frac{\partial}{\partial t}(\rho \mathbf{v}) dV + \oint_A \rho \mathbf{v}\mathbf{v} \cdot \mathbf{n} dA = \oint_A \sigma \mathbf{n} dA + \int_V \rho \mathbf{g} dV \tag{17}$$

Once again using Gauss' theorem on the surface integrals and assuming the equation to be valid for any dV, the differential form of the conservation o linear momentum equation is:

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = \nabla \cdot \sigma + \rho \mathbf{g} \tag{18}$$

The left side of Eq. 18 can be further reduced by assuming an incompressible fluid, i.e. $\nabla \cdot \mathbf{v} = 0$, and also by rewriting $\nabla \cdot (\rho \mathbf{v}\mathbf{v}) = \mathbf{v} \cdot \nabla \cdot (\rho \mathbf{v}) + \rho \mathbf{v} \cdot \nabla \mathbf{v}$, thus:

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = \mathbf{v}\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \cdot (\rho \mathbf{v}) + \rho \frac{\partial \mathbf{v}}{\partial t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} = \rho \frac{D\mathbf{v}}{Dt} \tag{19}$$

where D/Dt is called the material derivative, generally represented in Eq. 20.

$$\frac{D(\cdot)}{Dt} = \frac{\partial(\cdot)}{\partial t} + \mathbf{v} \cdot \nabla(\cdot) \tag{20}$$

From Eqs. 18 and 19:

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}\nabla \cdot \sigma + \mathbf{g} \tag{21}$$

To get to the known form of the Navier-Stokes equations, the stress tensor $\boldsymbol{\sigma}$ is separated in a component related to the pressure p and a component related to the viscous stress tensor $\boldsymbol{\tau}$, so that:

$$\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\tau} \tag{22}$$

where $\mathbf{I}$ is the identity matrix.

$$\nabla \cdot \boldsymbol{\sigma} = \nabla p + \nabla \cdot \boldsymbol{\tau} \tag{23}$$

Considering a newtonian and incompressible fluid , the tensor $\boldsymbol{\tau}$ is described by the velocity and viscosity μ as

$$\boldsymbol{\tau} = \mu(\nabla \mathbf{v} + (\nabla \mathbf{v})^T) \tag{24}$$

and by applying the divergence operator $\nabla \cdot (\cdot)$ Eq. 24 becomes:

$$\nabla \cdot \tau = \mu \nabla^2 \mathbf{v} \tag{25}$$

Substituting Eqs. 25 and 23 in Eq. 21 we get to the usual form of the Navier-Stokes equation:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{g} \tag{26}$$

where $\nu = \mu/\rho$ is called the kinematic viscosity.

2.3 **Stream function-Vorticity Formulation**

This formulation is a way to express the Navier-Stokes equation in terms of the stream function ψ and the vorticity ω. This is possible because the flow velocity field is directly correlated with the stream function, presented in Eq. 27, which is a scalar function that can be used to plot the streamlines, i.e. the trajectory of the flow particles. Vorticity is a vector that describes the spinning motion of a continuum region of space. For a two dimensional flow the vorticity is always perpendicular to the flow plane and can be considered as a scalar function. It is defined as the curl of the velocity field (see Eq. 28 for the two dimensional cartesian description).

$$\frac{\partial \psi}{\partial y} = v_x \,, \quad \frac{\partial \psi}{\partial x} = -v_y \tag{27}$$

$$\omega = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \tag{28}$$

From Eqs. 27 and 28 we obtain the equation that relates the stream function and that vorticity as:

$$-\omega = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} \qquad \texttt{or} \qquad -\omega = \nabla^2 \psi \tag{29}$$

The other equation in this formulation is the vorticity transport. It is derived by applying the curl operator to the Navier-Stokes equation. In a two dimensional case this is done by differentiating the v_y component of the equation with respect to the x spatial variable and the v_x component to the y spatial variable, then subtracting both results, thus:

$$\frac{\partial}{\partial y}\left[\frac{Dv_x}{Dt}\right] - \frac{\partial}{\partial x}\left[\frac{Dv_y}{Dt}\right] = \frac{\partial}{\partial y}\left[-\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 v_x + \mathbf{g}\right] - \frac{\partial}{\partial x}\left[-\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\nabla^2 v_y + \mathbf{g}\right] \tag{30}$$

Opening the material derivatives on the left side of the Eq. 30 we get:

$$\frac{\partial}{\partial y}\left[\frac{Dv_x}{Dt}\right]-\frac{\partial}{\partial x}\left[\frac{Dv_y}{Dt}\right]=\frac{\partial}{\partial y}\left[\frac{\partial v_x}{\partial t}+v_x\frac{\partial v_x}{\partial x}+v_y\frac{\partial v_x}{\partial y}\right]-\frac{\partial}{\partial x}\left[\frac{\partial v_y}{\partial t}+v_x\frac{\partial v_y}{\partial x}+v_y\frac{\partial v_y}{\partial y}\right] \tag{31}$$

And by using the property in Eq. 33 together with the vorticity definition in Eq. 28, Eq. 31 becomes:

$$\frac{\partial}{\partial t}\left[\frac{\partial v_x}{\partial y}-\frac{\partial v_y}{\partial x}\right]+v_x\frac{\partial}{\partial x}\left[\frac{\partial v_x}{\partial y}-\frac{\partial v_y}{\partial x}\right]+v_y\frac{\partial}{\partial y}\left[\frac{\partial v_x}{\partial y}-\frac{\partial v_y}{\partial x}\right]= \frac{\partial\omega}{\partial t}+v_x\frac{\partial\omega}{\partial x}+v_y\frac{\partial\omega}{\partial y}=\frac{D\omega}{Dt} \tag{32}$$

$$\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_j}\right)=\frac{\partial}{\partial x_j}\left(\frac{\partial}{\partial x_i}\right) \tag{33}$$

Considering the gravity and viscosity to be constant and by also using the property in Eq. 33 with Eq. 28, the right hand side of Eq. 30 is written as:

$$\frac{1}{\rho}\left[\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial x}\right)-\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial y}\right)\right]+\nu\left[\frac{\partial(\nabla^2 v_x)}{\partial y}-\frac{\partial(\nabla^2 v_y)}{\partial y}\right]= \nu\left[\frac{\partial^2}{\partial x^2}\left(\frac{\partial v_x}{\partial y}-\frac{\partial v_y}{\partial x}\right)+\frac{\partial^2}{\partial y^2}\left(\frac{\partial v_x}{\partial y}-\frac{\partial v_y}{\partial x}\right)\right]=\nu\left[\frac{\partial^2\omega}{\partial x^2}+\frac{\partial^2\omega}{\partial y^2}\right] \tag{34}$$

Placing together the right hand side (see Eq. 34) and the left hand side (see Eq. 32), the vorticity transport equation is:

$$\frac{D\omega}{Dt}=\nu\nabla^2\omega \tag{35}$$

2.4 **Non-Dimensional Form**

In many situations the non-dimensional form of the equations in a mathematical model are useful to understand which are the more relevant factors in the dynamic of the system being investigated. Moreover, the non-dimensional numbers can be used to study experimental models in different scales because two different systems that are governed by the same mathematical model are said to be similar if they share the same product between non-dimensional numbers.

For the vorticity transport equation, the non-dimensional variables (with superscript *) in Eq. 36 are defined by some reference parameters, that are the characteristic length L and characteristic velocity U_0.

$$t^* = \frac{U_0 t}{L}\,, \quad x^* = \frac{x}{L}\,, \quad y^* = \frac{y}{L}\,, \quad \mathbf{v}^* = \frac{\mathbf{v}}{U_0} \tag{36}$$

From the definition of vorticity in Eq. 28 the non-dimensional vorticity is:

$$\omega = \frac{\partial v_y^* U_0}{\partial (x^* L)} - \frac{\partial v_x^* U_0}{\partial (y^* L)} = \frac{U_0}{L}\left(\frac{\partial v_y^*}{\partial x^*} - \frac{\partial v_x^*}{\partial y^*}\right) = \frac{U_0}{L}\omega^* \tag{37}$$

Substituting the non-dimensional variables in the vorticity transport equation (see Eq. 35):

$$\frac{\partial(\omega^*(U_0/L))}{\partial(t^*(L/U))} + v_x^* U_0 \frac{\partial(\omega^*(U_0/L))}{\partial(x^* L)} + v_y^* U_0 \frac{\partial(\omega^*(U_0/L))}{\partial(y^* L)} = \nu\left[\frac{\partial^2(\omega^*(U_0/L))}{\partial(x^* L)^2} + \frac{\partial^2 \omega^*(U_0/L))}{\partial(y^* L)^2}\right] \tag{38}$$

Because the parameters L and U_0 are referential constant values, they can move out of the derivatives and the equation becomes:

$$\frac{U_0^2}{L^2}\left(\frac{\partial \omega^*}{\partial t^*} + v_x^* \frac{\partial \omega^*}{\partial x^*} + v_y^* \frac{\partial \omega^*}{\partial y^*}\right) = \frac{U_0 \nu}{L^3}\left(\frac{\partial^2 \omega^*}{\partial x^{*2}} + \frac{\partial^2 \omega^*}{\partial y^{*2}}\right) \tag{39}$$

By writing the material derivative the equation is reduced to:

$$\frac{D\omega^*}{Dt^*} = \frac{\nu}{U_0 L}\nabla^2\omega^* \tag{40}$$

The non-dimensional stream function is obtained by substituting ω^* in Eq. 27, so:

$$-\omega^*\frac{U_0}{L} = \frac{\partial^2\psi}{\partial(x^*L)^2} + \frac{\partial\psi}{\partial(y^*L)^2} = \frac{1}{L^2}\left(\frac{\partial^2\psi}{\partial x^{*^2}} + \frac{\partial^2\psi}{\partial y^{*^2}}\right) \tag{41}$$

Then the constants U_0 and L move to the derivatives as:

$$-\omega^* = \frac{\partial^2(\psi/(U_0L))}{\partial x^{*^2}} + \frac{\partial^2(\psi/(U_0L))}{\partial y^{*^2}} \tag{42}$$

and now the non-dimensional stream function is defined as:

$$\psi^* = \frac{\psi}{U_0 L} \tag{43}$$

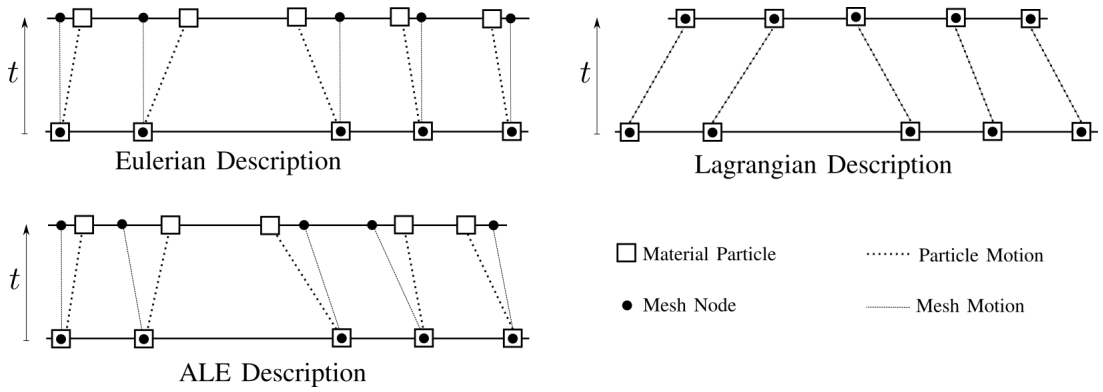
The non-dimensional parameter found in Eq. 40 is known as the Reynolds number. This is the main non-dimensional number used to characterize different types of flow. An important property of this number is the ability to distinguish between laminar and turbulent flow, which happens if Re is greater than a critical value Re_t for a given system ($Re_t \approx 2500$ for the flow in a pipe).

$$Re = \frac{U_0 L}{\nu} \tag{44}$$

2.5 **Arbitrary Lagrangian Eulerian Description**

There are two commonly used reference frames when dealing with discretizing the equations that govern fluid motion. One such frame is the Lagrangian viewpoint, in which the computational grid points are moving together with the fluid particles. The Eulerian

Figure 3 - Example of Lagrangian, Eulerian, and ALE descriptions for an one-dimensional mesh.



frame of reference is another largely used description for fluid motion and it works by setting a computational mesh fixed in space and the fluid moves through the grid points.

The arbitrary Lagrangian-Eulerian description is a third way in discretizing the fluid region and it aims in generalizing a frame of reference from the other two descriptions. The ALE description considers the computational mesh as a reference that may be moving with an arbitrary velocity **c** and it also doesn't depend on the fluid motion. Figure 3 shows an example of the Lagrangian, Eulerian and ALE descriptions for an one dimensional mesh between two time steps.

The Lagrangian and Eulerian descriptions come from two domains used in continuum mechanics. The material domain $\Omega_X \subset \mathbb{R}^n$, made up of material particles $\mathbf{X}$ and the spatial domain $\Omega_x \subset \mathbb{R}^n$ that consists of spatial points $\mathbf{x}$.

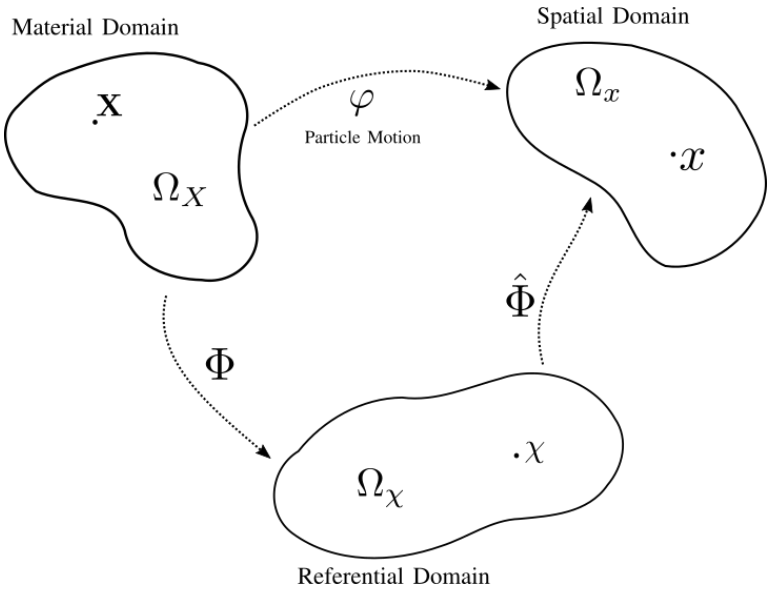
The Lagrangian descriptions assumes Ω_X as the reference configuration and it follows the motion of the material particles. This motion can be described by the mapping φ which relates the material coordinates to the spatial ones as $(\mathbf{X}, t) \rightarrow \varphi(\mathbf{X}, t) = (\mathbf{x}, t)$, which gives the law of motion:

$$\mathbf{x} = (\mathbf{X}, t) \,, \quad t = t \tag{45}$$

From the equation of motion we see that the material velocity is:

$$\mathbf{v}(\mathbf{X}, t) = \left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\mathbf{X}} \tag{46}$$

Figure 4 - Representation of the mappings between the material, referential and spatial domains.



where $\big|_{\mathbf{X}}$ means for a fixed material particle $\mathbf{X}$.

In both Eqs.45 and 46 the velocity and position depend upon the material particle $\mathbf{X}$ that is being observed. The Eulerian description takes the spatial domain $\Omega_x$as reference and so the velocity is examined in a fixed spatial point as $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$

For the ALE description, a third domain Ω_χ is introduced as the referential configuration containing the reference coordinates $\boldsymbol{\chi}$. The material domain is mapped into the referential domain by Φ and the referential domain is mapped into the spatial domain by $\hat{\Phi}$. Figure 4 show a schematic representation of the mappings.

The mapping $\hat{\Phi}$ represents the motion of the computational mesh in relation to the spatial coordinates and its velocity $\hat{\mathbf{v}}$ is given by:

$$\hat{\mathbf{v}}(\boldsymbol{\chi}, t) = \left.\frac{\partial \mathbf{x}}{\partial t}\right|_{\chi} \tag{47}$$

From the mapping Φ we define the velocity $\tilde{\mathbf{v}}$ of the material particles in relation to the referential domains as:

$$\tilde{\mathbf{v}}(\mathbf{X}, t) = \left.\frac{\partial \boldsymbol{\chi}}{\partial t}\right|_{\mathbf{X}} \tag{48}$$

According to Donea et al. (2004) these mappings are related by $\varphi = \hat{\Phi} \circ \Phi$ and it is

possible to obtain the following relationship between the velocities:

$$\mathbf{v} - \hat{\mathbf{v}} = \frac{\partial \mathbf{x}}{\partial \chi} \cdot \tilde{\mathbf{v}} \tag{49}$$

By using the relationship in Eq. 49, the relative velocity between the material particles and the computational mesh is defined as $\mathbf{c} = \mathbf{v} - \hat{\mathbf{v}}$.

In the equations presented in the previous sections the effect of using the ALE description of motion is captured by the material derivative in the vorticity transport equation (see Eq.35). According to Hughes, Liu and Zimmermann (1981) this derivative can be expressed in terms of the convective velocity and for a scalar function f it becomes:

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + (\mathbf{v} - \hat{\mathbf{v}}) \cdot \nabla f = \frac{\partial f}{\partial t} + \mathbf{c} \cdot \nabla f \tag{50}$$

If the mesh is to be considered fixed, i.e. $\hat{\mathbf{v}} = 0$, the purely Eulerian description is recovered, and by setting the mesh velocity to follow the fluid motion with $\hat{\mathbf{v}} = \mathbf{v}$ the equation is reduced to the purely Lagrangian description.

2.6 **Governing Equations**

The complete set of differential equations that constitute the mathematical model applied to a domain together with its boundary conditions and initial conditions is presented in this section.

Let $\Omega \subset \mathbb{R}^2$ be a domain with boundary $\Gamma = \Gamma_D \cup \Gamma_N$. The Strong statement of the model derived in the previous chapter is: *Find* ω, ψ, $\mathbf{v}$ *such that*

$$\begin{cases} \frac{D\omega}{Dt} = \frac{1}{Re}\nabla^2\omega \\ \nabla^2\psi = -\omega \qquad\qquad \texttt{in } \Omega \times t \in]0,T[, \\ \mathbf{v} = \left(\frac{\partial\psi}{\partial y}\,,\ -\frac{\partial\psi}{\partial x}\right) \end{cases} \tag{51}$$

subject to the boundary conditions

$$\begin{cases} \omega = \omega_d(\mathbf{x}, t) \\ \psi = \psi_d(\mathbf{x}, t) \qquad \texttt{on } \Gamma_D, \\ \mathbf{v} = \mathbf{v}_d(\mathbf{x}, t) \end{cases} \tag{52}$$

$$\begin{cases} \frac{\partial \omega}{\partial n} = 0 \\ \frac{\partial \psi}{\partial n} = 0 \qquad \texttt{on } \Gamma_N \\ \frac{\partial \mathbf{v}}{\partial n} = 0 \end{cases} \tag{53}$$

and initial conditions:

$$\begin{cases} \omega = \omega_0(\mathbf{x}) \\ \psi = \psi_0(\mathbf{x}) \qquad \texttt{in } \Omega, t = 0. \\ \mathbf{v} = \mathbf{v}_0(\mathbf{x}) \end{cases} \tag{54}$$

The boundary conditions defined over Γ_D are known as Dirichlet conditions, and they are used when we know or want to impose the value of the function on the boundary. Over Γ_N we have what are called Neumann conditions, where it is specified the value of the first derivative of the function in a direction normal to the boundary.

3 FINITE ELEMENTS METHOD

3.1 Weak Formulation

The weak formulation, also known as variational formulation, of a problem is the heart of the Finite Elements method. To write the equations in their weak form we make use of two classes of functions, the trial solutions and the weight functions. Both these classes are defined from the Sobolev space

$$H^1(\Omega) = \left\{ u \in L^2(\Omega),\ \frac{\partial u}{\partial x_i} \in L^2(\Omega), i = 1,2,\cdots,n \right\}, \tag{55}$$

where $L^2(\Omega)$ is the space of square-integrable functions

$$L^2(\Omega) = \left\{ u : \Omega \to \mathbb{R}, \int_\Omega u^2 d\Omega < \infty \right\}. \tag{56}$$

The trial solutions are a set of functions in the Sobolev space that are also required to satisfy the Dirichlet conditions, so, for each variable of the problem we define the spaces

$$\begin{aligned}
\mathbb{S}_\omega(\Omega) &= \left\{ \omega | \omega \in H^1(\Omega), \omega = \omega_d \ \texttt{on}\ \Gamma_D \right\}, \\
\mathbb{S}_\psi(\Omega) &= \left\{ \psi | \psi \in H^1(\Omega), \psi = \psi_d \ \texttt{on}\ \Gamma_D \right\}, \\
\mathbb{S}_\nu(\Omega) &= \left\{ \nu | \nu \in H^1(\Omega), \nu = \nu_d \ \texttt{on}\ \Gamma_D \right\}.
\end{aligned}$$

Similarly, the space of the weight functions is a subset of the Sobolev space and we require these functions to vanish at the Dirichlet boundary

$$\mathbb{W}(\Omega) = \left\{ \phi | \phi \in H^1(\Omega), \phi = 0 \ \texttt{on}\ \Gamma_D \right\}. \tag{57}$$

To obtain the final weak formulation, we need to multiply the governing equations by

the weight function and integrate over the domain:

$$\int_{\Omega} \phi \left[\frac{D\omega}{Dt} - \frac{1}{Re} \nabla^2 \omega \right] d\Omega = 0, \tag{58}$$

$$\int_{\Omega} \phi \left[\nabla^2 \psi + \omega \right] d\Omega = 0, \tag{59}$$

$$\int_{\Omega} \phi \left[\mathbf{v} - \left(\frac{\partial \psi}{\partial y} \, , \; -\frac{\partial \psi}{\partial x} \right) \right] d\Omega = 0. \tag{60}$$

Rewriting the integrals and separating the velocity components:

$$\int_{\Omega} \phi \frac{D\omega}{Dt} d\Omega - \int_{\Omega} \phi \frac{1}{Re} \nabla^2 \omega d\Omega = 0, \tag{61}$$

$$\int_{\Omega} \phi \nabla^2 \psi d\Omega + \int_{\Omega} \phi \omega d\Omega = 0, \tag{62}$$

$$\int_{\Omega} \phi v_x d\Omega - \int_{\Omega} \phi \frac{\partial \psi}{\partial y} d\Omega = 0, \tag{63}$$

$$\int_{\Omega} \phi v_y d\Omega + \int_{\Omega} \phi \frac{\partial \psi}{\partial x} d\Omega = 0. \tag{64}$$

Using Green's theorem we can reduce the order of the derivatives on the terms that have the Laplacian operator ∇^2, those integrals became:

$$\int_{\Omega} \phi \frac{1}{Re} \nabla^2 \omega d\Omega = -\int_{\Omega} \nabla \phi \cdot \frac{1}{Re} \nabla \omega d\Omega + \int_{\Gamma} \phi (\frac{1}{Re} \nabla \omega \cdot \mathbf{n}) d\Gamma, \tag{65}$$

$$\int_\Omega \phi \nabla^2 \psi = -\int_\Omega \nabla\phi \cdot \nabla\psi \, d\Omega + \int_\Gamma \phi(\nabla\psi \cdot \mathbf{n}) \, d\Gamma. \tag{66}$$

The integrals over Γ represent the contributions from the boundary conditions and $\mathbf{n}$ is a normal vector of the boundary. Because of the Neumann conditions imposed in our formulation the part of those integrals that range over Γ_N vanish, i. e. $\nabla\psi \cdot \mathbf{n} = 0$ and $\nabla\omega \cdot \mathbf{n} = 0$. Furthermore, the space in which we defined the weight function requires its value on the Dirichlet boundary Γ_D to be zero, therefore the integrals over the whole boundary is zero. Substituting Eqs. 65, 66 in Eqs. 61 , 62:

$$\int_\Omega \phi \frac{D\omega}{Dt} d\Omega + \frac{1}{Re} \int_\Omega \nabla\phi \cdot \nabla\omega \, d\Omega = 0, \tag{67}$$

$$\int_\Omega \phi\omega \, d\Omega - \int_\Omega \nabla\phi \cdot \nabla\psi \, d\Omega = 0. \tag{68}$$

The Weak statement of our formulation is:

Find $\omega \in \mathbb{S}_\omega$, $\psi \in \mathbb{S}_\psi$ *and* $v_x, v_y \in \mathbb{S}_v$ *such that*

$$\int_\Omega \phi \cdot \frac{D\omega}{Dt} d\Omega + \int_\Omega \frac{1}{Re} \nabla\phi \cdot \nabla\omega \, d\Omega = 0 \tag{69}$$

$$\int_\Omega \phi \cdot \omega \, d\Omega - \int_\Omega \nabla\phi \cdot \nabla\psi \, d\Omega = 0 \tag{70}$$

$$\int_\Omega \phi \cdot v_x d\Omega - \int_\Omega \phi \cdot \frac{\partial\psi}{\partial y} d\Omega = 0 \tag{71}$$

$$\int_\Omega \phi \cdot v_y d\Omega + \int_\Omega \phi \cdot \frac{\partial\psi}{\partial x} d\Omega = 0 \tag{72}$$

for all $\phi \in \mathbb{W}$.

3.2 **Galerkin Method**

So far we have described our problem in its variational statement by requiring our solutions to be represented by functions in sub-spaces of $H^1(\Omega)$. However, these sub-spaces still contain infinitely many functions. The Galerkin method is an approximation method in which we construct finite-dimensional spaces that are associated with a discretization

$\bar{\Omega}$ of the domain Ω. Hughes (2000) shows that it is sufficient to consider only the finite-dimensional space for the weight functions, denoted by $\mathbb{W}^{(n)}$. The trial solutions can be expressed as a sum between a function in $\mathbb{W}^{(n)}$ and a known function that satisfies the boundary conditions.

Now we need write the variables of our problem in terms of the n basis functions N_i(also known as shape functions) of the constructed finite-dimensional space, and by the basis function N_{n+1} that is defined for the boundary condition.

$$\psi \approx \sum_{i}^{n+1} \psi_i N_i \tag{73}$$

$$\omega \approx \sum_{i}^{n+1} \omega_i N_i \tag{74}$$

$$v_x \approx \sum_{i}^{n+1} u_i N_i \tag{75}$$

$$v_y \approx \sum_{i}^{n+1} v_i N_i \tag{76}$$

$$\phi \approx \sum_{i}^{n} \phi_i N_i \tag{77}$$

where $\omega_{n+1} = \omega_d$, $\psi_{n+1} = \psi_d$, $u_{n+1} = u_d$ and $v_{n+1} = v_d$ are the Dirichlet boundary conditions.

Substituting the approximations in the weak formulation equations:

$$\int_\Omega \sum_{j}^{n} \phi_j N_j \frac{D}{Dt} \left(\sum_{i}^{n+1} \omega_i N_i \right) d\Omega + \frac{1}{Re} \int_\Omega \nabla \left(\sum_{j}^{n} \phi_j N_j \right) \cdot \nabla \left(\sum_{i}^{n+1} \omega_i N_i \right) d\Omega = 0 \tag{78}$$

$$\int_\Omega \nabla \left(\sum_{j}^{n} \phi_j N_j \right) \cdot \nabla \left(\sum_{i}^{n+1} \psi_i N_i \right) d\Omega - \int_\Omega \sum_{j}^{n} \phi_j N_j \sum_{i}^{n+1} \omega_i N_i d\Omega = 0 \tag{79}$$

$$\int_\Omega \sum_{j}^{n} \phi_j N_j \sum_{i}^{n+1} u_i N_i d\Omega - \int_\Omega \sum_{j}^{n} \phi_j N_j \frac{\partial}{\partial y} \left(\sum_{i}^{n+1} \psi_i N_i \right) d\Omega = 0 \tag{80}$$

$$\int_\Omega \sum_j^n \phi_j N_j \sum_i^{n+1} v_i N_i d\Omega + \int_\Omega \sum_j^n \phi_j N_j \frac{\partial}{\partial x}\left(\sum_i^{n+1} \psi_i N_i\right) d\Omega = 0 \tag{81}$$

The equations can be further developed by the fact that only the shape functions depend on the domain spatial variables, thus

$$\sum_j^n \phi_j \left[\sum_i^{n+1}\left[\frac{D\omega_i}{Dt}\int_\Omega N_j N_i d\Omega + \frac{1}{Re}\omega_i \int_\Omega \nabla N_j \cdot \nabla N_i d\Omega\right]\right] = 0 \tag{82}$$

$$\sum_j^n \phi_j \left[\sum_i^{n+1}\left[\psi_i \int_\Omega \nabla N_j \cdot \nabla N_i d\Omega - \omega_i \int_\Omega N_j N_i d\Omega\right]\right] = 0 \tag{83}$$

$$\sum_j^n \phi_j \left[\sum_i^{n+1}\left[u_i \int_\Omega N_j N_i d\Omega - \psi_i \int_\Omega N_j \frac{\partial N_i}{\partial y} d\Omega\right]\right] = 0 \tag{84}$$

$$\sum_j^n \phi_j \left[\sum_i^{n+1}\left[v_i \int_\Omega N_j N_i d\Omega + \psi_i \int_\Omega N_j \frac{\partial N_i}{\partial x} d\Omega\right]\right] = 0 \tag{85}$$

The Eqs. 82, 83, 84 and 84 have to hold for all weight functions in $\mathbb{W}^{(n)}$ so that the coefficients can vary arbitrarily, therefore

$$\sum_i^{n+1}\left[\frac{D\omega_i}{Dt}\int_\Omega N_j N_i d\Omega + \frac{1}{Re}\omega_i \int_\Omega \nabla N_j \cdot \nabla N_i d\Omega\right] = 0 \tag{86}$$

$$\sum_i^{n+1}\left[\psi_i \int_\Omega \nabla N_j \cdot \nabla N_i d\Omega - \omega_i \int_\Omega N_j N_i d\Omega\right] = 0 \tag{87}$$

$$\sum_i^{n+1}\left[u_i \int_\Omega N_j N_i d\Omega - \psi_i \int_\Omega N_j \frac{\partial N_i}{\partial y} d\Omega\right] = 0 \tag{88}$$

$$\sum_{i}^{n+1}\left[\upsilon_i\int_\Omega N_j N_i d\Omega+\psi_i\int_\Omega N_j\frac{\partial N_i}{\partial x}d\Omega\right]=0 \tag{89}$$

Each of the above equations represents a $n\times n$ linear system in which the n unknown variables are: ω_i in Eq.86; ψ_i in Eq.87; u_i in Eq.88; υ_i in Eq.89.

3.2.1 **Matricial Equations**

From the Galerkin approximation done in the previous section, we can write the linear systems of equations in a more concise manner. Let

$$\mathbf{K}=\int_\Omega \nabla N_j\cdot\nabla N_i d\Omega \tag{90}$$

$$\mathbf{M}=\int_\Omega N_j N_i d\Omega \tag{91}$$

$$\mathbf{G}_x=\int_\Omega N_j\frac{\partial N_i}{\partial x}d\Omega \tag{92}$$

$$\mathbf{G}_y=\int_\Omega N_j\frac{\partial N_i}{\partial t}d\Omega \tag{93}$$

where **K**, **M**, are known as stiffness and mass matrices, and $\mathbf{G}_x$, $\mathbf{G}_y$ are matrices associated with derivatives in the x and y directions.

Also, let the unknown variables be: $\omega_i=\boldsymbol{\omega}$; $\psi_i=\psi$; $u_i=\mathbf{v_x}$; $\upsilon_i=\mathbf{v_y}$. The matrix equations are:

$$\mathbf{M}\frac{D\boldsymbol{\omega}}{Dt}+\frac{1}{Re}\mathbf{K}\boldsymbol{\omega}=0 \tag{94}$$

$$\mathbf{K}\psi = \mathbf{M}\omega \tag{95}$$

$$\mathbf{M}\mathbf{v}_\mathbf{x} = \mathbf{G}_\mathbf{y}\psi \tag{96}$$

$$\mathbf{M}\mathbf{v}_\mathbf{y} = -\mathbf{G}_\mathbf{x}\psi \tag{97}$$

3.3 **Boundary Conditions**

Having well defined boundary conditions are an important part in in solving any differential equation. For the simulations presented in this work, four classes of boundaries were defined, each with an associated boundary condition for the velocity, stream function and vorticity fields. The classes are:

- **Inflow:** This boundary is a region of space where the fluid moves into the simulation domain. Only Dirichlet boundary conditions are used here. First a velocity field is chosen according to the problem being solved, then the stream function and vorticity fields are analytically obtained from the velocity and imposed on the boundary.
- **Outflow:** This is the region of space where the fluid moves out of the simulation domain. Differently from the inflow, in here only Neumann natural conditions are applied so that the derivatives of all variables are null in the boundary normal direction. Much care is needed in choosing the outflow geometry as numerical errors can occur and propagate back to spoil the solution in the rest of the domain.
- **Wall:** Represents a physical wall that restricts the fluid flow. It can also be used to represent an obstacle as a hole boundary in the domain. The no-slip condition $\mathbf{v} = \mathbf{v}_{wall}$ is imposed for the velocity field, meaning that a fluid particle has to move with the boundary wall.

  By definition, the stream function has to be constant along a stream line and walls can be considered stream lines, therefore a constant value is imposed. Furthermore, the difference between the values on two wall boundaries represent the flow rate between them.

The vorticity has an imposed Dirichlet condition that has to be calculated. From the literature, the most common way in estimating ω values on the boundary is by using a finite difference approximation from the stream function, as done by **REF**. However, a FEM approximation from the velocity field is proposed. The vorticity on the boundary is calculated by solving:

$$\mathbf{M}\omega = \mathbf{G_x}v_y - \mathbf{G_y}v_x \tag{98}$$

- **Slip/Symmetry:** This is a condition that allows the fluid to move along the boundary as a stream line, so the stream function is imposed as constant. The velocity in the normal direction is set as zero and in the tangential direction has a null Neumann condition. The vorticity can be obtained from the velocity and is set as zero.

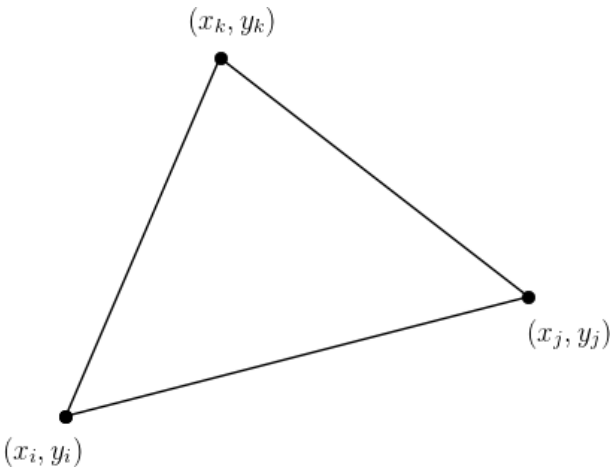
3.4 **Mesh Elements**

In the previous sections it was shown that a finite elements solution depends on a discretization $\bar{\Omega}$ of Ω. More generally, $\bar{\Omega}$ is a collection of a finite number n_e of partitions Ω^e such that the intersection between all elements, is an empty set. This means that there is no overlapping between each element.

There are many ways and classes of elements that can be used to represent a discrete domain and that is one of the biggest FEM advantages. In this work, only the triangular class of elements with linear polynomial basis functions was implemented for the simulations. Polynomials are usually a common choice of basis functions because they require less computational cost in computing their integrals, and the result precision can be improved by increasing its degree.

The functions for this type of element are widely described in the literature, such as in (LEWIS; NITHIARASU; SEETHARAMU, 2004), and presented in Eq. 99 with its coefficients in Eq. 100. A schematic representation of a single triangular element with three nodes is shown in Fig. 5.

Figure 5 - Representation of a linear triangular element with three nodes.



$$
\begin{aligned}
&\mathbf{N}(x,y) = [N_i \;\; N_j \;\; N_k] \\
&N_i = \frac{1}{2A}(a_i + b_i x + c_i y) \\
&N_j = \frac{1}{2A}(a_j + b_j x + c_j y) \\
&N_k = \frac{1}{2A}(a_k + b_k x + c_k y)
\end{aligned}
\tag{99}
$$

$$
\begin{aligned}
&a_i = x_j y_k - x_k y_j; \;\; b_i = y_j - y_k; \;\; c_i = x_k - x_j \\
&a_j = x_k y_i - x_i y_k; \;\; b_j = y_k - y_i; \;\; c_j = x_i - x_k \\
&a_k = x_i y_j - x_j y_i; \;\; b_k = y_i - y_j; \;\; c_k = x_j - x_i
\end{aligned}
\tag{100}
$$

In the FEM context, the integrals defined in Eqs. 94, 95, 96 and 97 are first evaluated for a single element and then their contributions are assembled into the respective global matrices, a procedure described in Chapter 4. For the linear triangular element, the local matrices depend on the element area, calculated by $2A = a_i + a_j + a_k$, and are:

$$
k^e = \frac{1}{4A}\begin{bmatrix} b_i^2 + c_i^2 & b_i b_j + c_i c_j & b_i b_k + c_i c_k \\ b_i b_j + c_i c_j & b_j^2 + c_j^2 & b_k b_j + c_k c_j \\ b_i b_k + c_i c_k & b_k b_j + c_k c_j & b_k^2 + c_k^2 \end{bmatrix}
\tag{101}
$$

$$m^e = \frac{A}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{102}$$

$$g_x^e = \frac{1}{6}\begin{bmatrix} b_i & b_j & b_k \\ b_i & b_j & b_k \\ b_i & b_j & b_k \end{bmatrix} \tag{103}$$

$$g_y^e = \frac{1}{6}\begin{bmatrix} c_i & c_j & c_k \\ c_i & c_j & c_k \\ c_i & c_j & c_k \end{bmatrix} \tag{104}$$

3.5 **Semi-Lagrangian Method**

So far we were only concerned with the spatial discretization of our equations and for that we used the Galerkin method. Now we have to deal with the time dependency of our problem that is presented by the material derivative in Eq. 94. The numerical treatment of the time discretization needs to be carefully chosen to assure that the solutions are numerically stable. This characteristic is essential for successful fluid simulations, especially in dealing with flows that have a high Reynolds number.

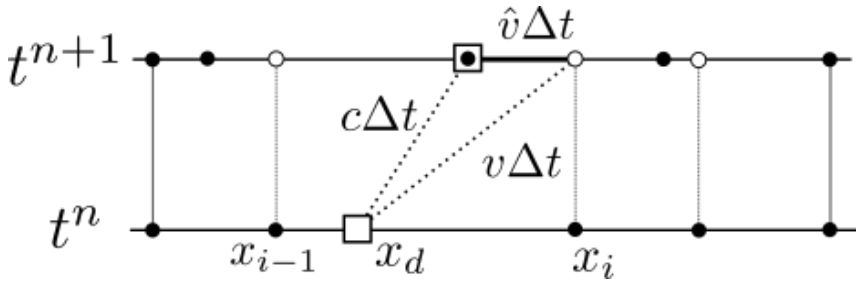
A semi-Lagrangian scheme is a method that gives an approximation of the material derivative, integrating it over the characteristic trajectory of a fluid particle during a discrete time step Δt. According to Pironneau (1982) the semi-Lagrangian is shown to be unconditionally stable, meaning it has no restriction in the choice of time step used.

Let $\omega_i^{n+1} = \omega(\mathbf{x}_i, t^{n+1})$ where x_i is a point in the computational mesh and t^{n+1} is the time step in which ω needs to be calculated. A semi-Lagrangian first order scheme can be written as:

$$\frac{D\omega}{Dt} \approx \frac{\omega_i^{n+1} - \omega_d^n}{\Delta t} \tag{105}$$

with $\omega_d^n = \omega(\mathbf{x}_d, t^n)$ being the value of vorticity from the last known time step t and at a point

Figure 6 - Example of the semi Lagrangian trajectory integration for a 1D mesh in an ALE context. Squares represent the material point, black dots are the current mesh nodes position and the white dots are the mesh nodes position in the previous time step.



x_d, called departure point. Applying the material derivative approximation in Eq. 105 to the Eq. 94, using an implicit scheme, the final matricial equation for the vorticity transport is then:

$$\left(\frac{\mathbf{M}}{\Delta t}+\frac{1}{Re}\mathbf{K}\right)\boldsymbol{\omega}^{n+1}=\frac{\mathbf{M}}{\Delta t}\boldsymbol{\omega}_d^n \tag{106}$$

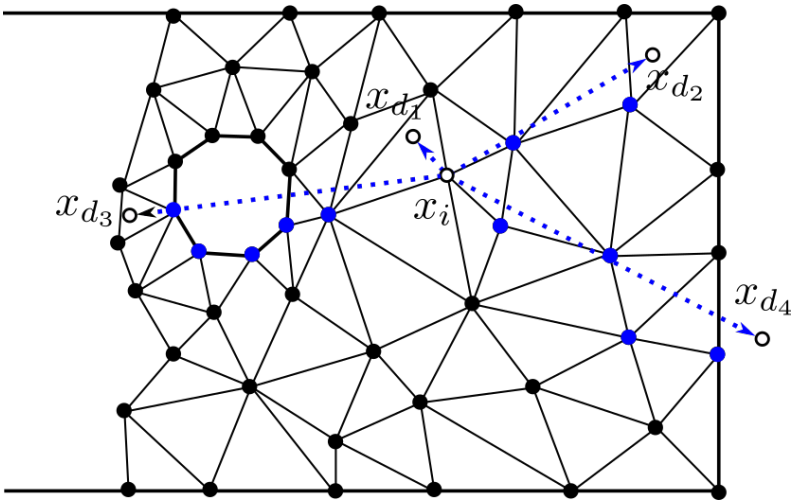
To find the coordinates of the departure points we linearly approximate the fluid trajectory between the steps t^n and t^{n+1} and for the moving mesh context of the ALE reference frame, the position is obtained by using Eq. 107. A schematic representation of this is shown in Fig.6 for a one dimensional mesh.

$$\mathbf{x}_d=\mathbf{x}_i-(\mathbf{c}+\hat{\mathbf{v}})\Delta t \tag{107}$$

$\mathbf{c}$ is the ALE velocity and $\hat{\mathbf{v}}$ the mesh velocity.

After calculating the departure coordinates, a search algorithm is necessary to find in which element the point x_d is. The developed search procedure made use of a data structure that maps each node to its surrounding elements. By starting at the initial position x_i (position of the i-th node) we check the nearby elements and, if x_d is not found, the nearest node in relation to x_d is chosen for a next step in the procedure. This repeats until the departure position is found and then an interpolation is done to assign the values of ω_d^n. This interpolation is done by using the same basis functions associated with the mesh nodes in the element where the departure point is found.

Figure 7 - Schematic representation of 4 possible trajectories of the departure node search algorithm in a triangular mesh.



The schematic representation in Fig. 7 shows four possible situations that may happen during a search step, beginning from the mesh node x_i:

- The departure node x_{d_1} is located in a neighbor element. In this case the search algorithm finishes after the first iteration over the mapping data structure;
- The departure node x_{d_2} is located in a far element. In this case the search takes three iterations moving from the node x_i to the other two blue dots shown in the figure. Generally, longer trajectories approximated by a first order semi-Lagrangian method may have an impact in the accuracy of the solution.
- The departure node x_{d_3} has a trajectory that goes through a hole boundary in the mesh. In this case the algorithm is capable of find the node my moving through the closest nodes along the boundary, indicated in blue.
- The departure node x_{d_4} is outside of the mesh. In this case the algorithm tries to search the neighbor elements of the closest node for a second time and when it happens it assumes the departure node is out of bounds and assign its value as the boundary value.

4 IMPLEMENTATION

One of the objectives of this work was the implementation of an in-house code capable of simulation fluid flow with the stream function-vorticity formulation. The Python programming language was chosen for that task as it is a widely used language with a great community support and well established mathematical packages for linear algebra calculations. Two open source softwares were also used in this work, Gmsh for mesh generation and Paraview for results visualization and post-processing.

4.1 Mesh Generation

To begin a numerical simulation we first need to define what is the domain geometry and then generate the computational mesh that will be used to solve the discretized linear system of equations. This generation is done by the software *Gmsh*, developed by Geuzaine and Remacle (2009) and released under GNU General Public License.

The software is divided in modules and the ones used were:

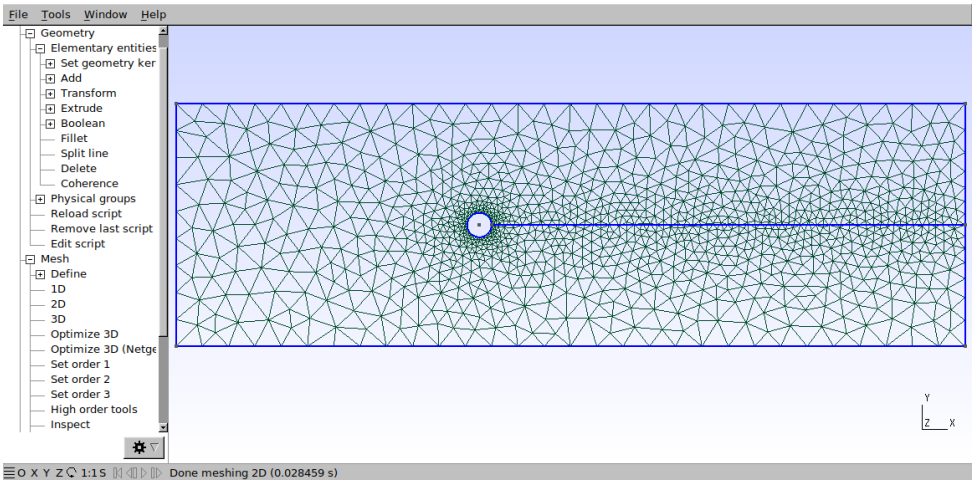
- **Geometric:** Used for the construction of the geometric objects (points, lines, surfaces, volumes) that define the domain to be discretized.
- **Mesh:** It is responsible to interpret the geometric objects and apply a discretization method to generate the computational mesh. This module has a wide selection of element types and is able to export standard FEM mesh files.

To obtain a mesh with linear triangular elements, the standard mesh generation method was used, which discretizes the geometric domain by applying a type of Delaunay triangulation process. The mesh is exported in a *.msh* file that is read by the python main code with the package *meshio*. An example of the Gmsh GUI with a triangular mesh is shown in Fig. 8.

4.2 Mesh Movement

In the ALE context, the computational mesh is allowed to move with an arbitrary velocity. However, much attention is needed when dislocating the nodes to assure that the

Figure 8 - Example of a triangular mesh with a hole boundary and different element sizes generated by Gmsh with its GUI.



mesh keeps its quality. When the elements undergo big distortions or overlap, the solution accuracy may be spoiled.

A few rules were implemented to guarantee a better control in the mesh movement. The mesh velocity $\hat{\mathbf{v}}$ can be calculated by:

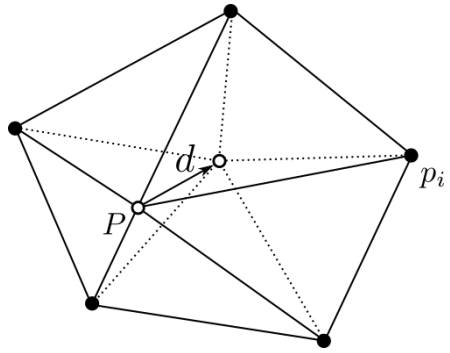
$$\hat{\mathbf{v}}_i = \begin{cases} \rho_1 \mathbf{v}_i + \rho_2 \mathbf{v}_i^s + \rho_3 \mathbf{v}_i^e & \text{if } i \text{ is an interior node} \\ \mathbf{v}_i^b & \text{if } i \text{ is a node in a moving boudary} \\ 0 & \text{if } i \text{ is a node in a fixed boundary} \end{cases} \tag{108}$$

In Eq. 108: $\mathbf{v}$ is the fluid velocity, $\mathbf{v}^b$ is a defined boundary velocity, $\mathbf{v}^s$ is a velocity calculated from a Laplacian Smoothing technique. The computational parameters ρ_1, ρ_2 and ρ_3 are used to fine tune the contribution from each defined velocity to the overall mesh velocity. The velocity $\mathbf{v}^e$ is defined as function of the distance from a moving boundary and the boundary velocity to assure the interior nodes move closer to it.

4.2.1 Laplacian Smoothing

A Laplacian Smoothing technique is used in this work to help maintain the mesh quality and avoid big distortions in the elements. The idea of this method is to move the mesh nodes by trying to equally redistribute locally from regions of a higher concentration

Figure 9 - A Laplacian Smooth method representation for a node P with five neighbors p_i. The vector **d** represents the movement of the interior node.



of points to regions with a lower concentration, and therefore achieving a better distribution. When well adjusted, the elements of a given triangular mesh tend to became more equilateral after some iterations of this procedure.

To calculate the mesh smoothing velocity, we take a weighted average of the displacement vectors from the node P to its neighbors p_i as it is presented by Eq. 109, where N_N is the number of neighbor nodes and then dividing it by the simulation time step Δt. The weight is the distance $l_i = ||\mathbf{p}_i - \mathbf{P}||$ between the nodes and $L = \sum_i^{N_N} l_i$. In this work, this procedure is called in every iteration when a moving mesh simulation is done and its influence can be controlled by the parameter ρ_2 in Eq. 108.

A representation of the scheme is shown in Fig. 9 for a node with five neighbors and an application of this method on a full mesh is presented on Fig. 10 by comparing the original to the smoothed version.

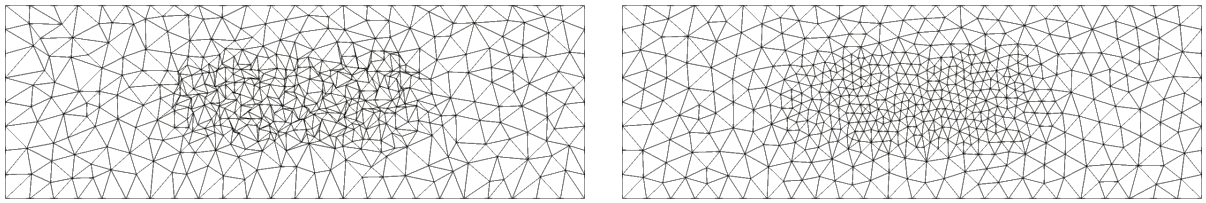
$$\begin{cases} \mathbf{d} = \sum_i^{N_N} \dfrac{l_i}{L}(\mathbf{p}_i - \mathbf{P}) \\ \mathbf{v}^s = \dfrac{\mathbf{d}}{\Delta t} \end{cases} \tag{109}$$

4.3 **Global Assembly**

The assembly of global matrices is an essential step in any FEM implementation. In simulations that change the mesh configuration over time multiple assemblies are needed to adequate the global matrices to the new elements form.

In this process, the contributions from each basis function, integrated over the local

Figure 10 - Application of the Laplacian smoothing technique in a poor quality mesh.



(a) Initial Mesh (b) Smoothed Mesh

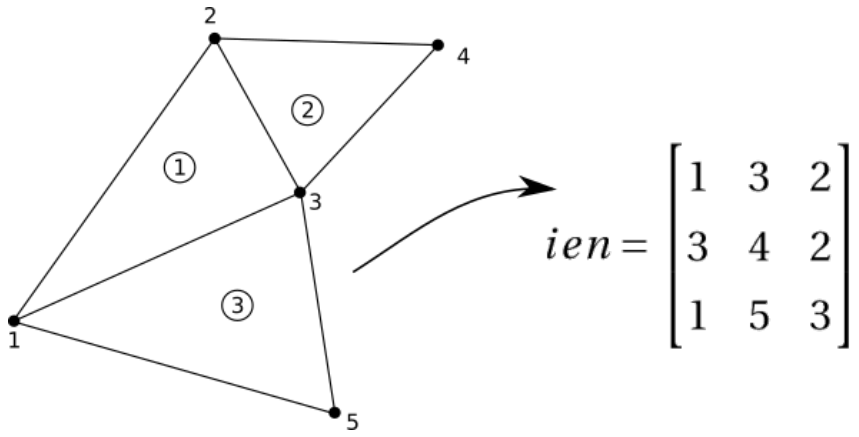
element, is added according to the following Python script:

```
for elem in range(num_ele):  # Looping over all elements

   k,m,gx,gy = local_trilin()   # Defining local matrices

   for i_local in range(3):             #
      i_global = ien[elem, i_local]     # Recover global index
                                        # from connectivity
      for j_local in range(3):          # matrix IEN
         j_global = ien[elem, j_local]  #

         K[i_global, j_global] += k[i_local, j_local]    # Assembling
         M[i_global, j_global] += m[i_local, j_local]    # from Local
         Gx[i_global, j_global] += gx[i_local, j_local]  # to Global
         Gy[i_global, j_global] += gy[i_local, j_local]  #
```

The connectivity matrix is a data structure that maps each element to the nodes it contains and how those nodes are connected. For a linear triangular element, this structure is an array with three columns where the nodes indexes are stored and organized in a clock-wise manner, and each line represents an element, Fig. 11 shows the connectivity matrix for a mesh with 3 triangular elements.

Inside the main elements loop there is also the necessary step of calculating the local matrices. In a more general FEM software this task is done by numeric integration of the shape functions according to the defined class of elements for the simulation. A usual choice is the Gaussian Quadrature method which works by a change of variables from the current element to a standard element where its integral is evaluated over a set number of quadrature

Figure 11 - Example of connectivity matrix for a mesh with three linear triangular elements.



points and then the result is obtained by summation of all values, with a respective weight for each point.

However, by implementing only linear triangles the shape functions can be analytically integrated by calculating the coefficients a, b, c as shown in Sec.3.4 form the previous chapter. With this the local $\mathbf{k}$, $\mathbf{m}$, $\mathbf{g}_x$ and $\mathbf{g}_y$ can be directly constructed.

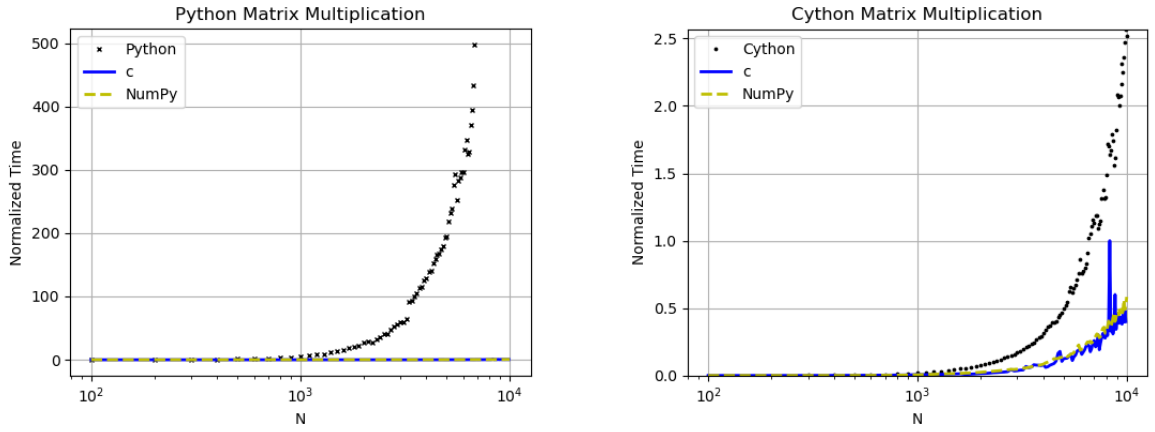
4.4 **Cython**

The main issue with the choice of using python is the slow performance it has when compared to lower level languages such as C and fortran. This problem is partially solved by using packages like Numpy and Scipy which implement mathematical subroutines in those lower level languages and make them callable by the python interpreter, however there are still efficiency problems with our defined functions.

A possible solution to improve the speed of our python functions is to rewrite them using the Cython language which is a super-set of Python, i.e. every Python code is Cython valid. It incorporates some C level elements like declaration of variables, pointers and data typing without sacrificing the python syntax, it also offers support to work with np.arrays from Numpy. Cython code works by first being compiled to C code and then compiled to a shared object callable from a Python script.

To illustrate the capabilities of this language, a first study was done to compare the efficiency of a standard matrix multiplication algorithm between Python, Cython, a C implementation and the Numpy function. The graphs in Fig.12 were plotted with the time nor-

Figure 12 - Comparison between Python, Cython, Numpy and C functions for multiplying matrices. N is the number of elements and the time is normalized by the maximum C time value.



(a) Python function

(b) Cython function

malized by the maximum C time value and they show a great difference in speed between Cython and Python. It is also important to mention the expected Numpy performance close to the C implementation.

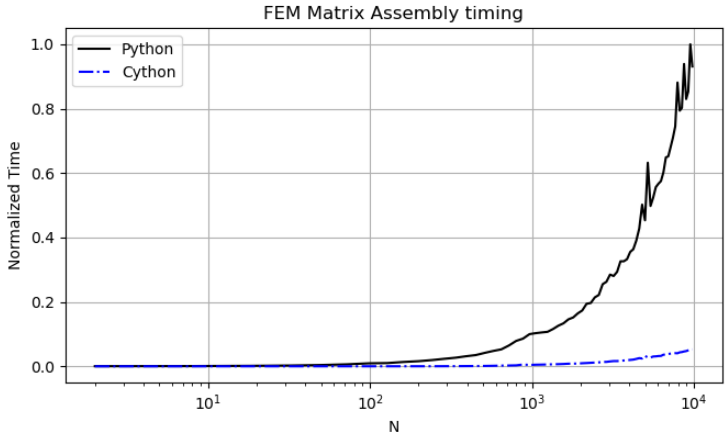
Considering the FEM algorithm developed in this work, we have evaluated the difference in speed between a Cython implementation of the global assembly function and the Python implementation. The results are presented in Fig. 13 with the time being normalized by the maximum Python value, and N being the number of elements. It shows a considerable improvement in time performance between the assembly functions.

4.5 **Algorithm**

This section presents an overview of the algorithm developed for simulating fluid flow with the stream function-vorticity formulation in the ALE reference frame. The solution of linear systems of equations was done by using the linear algebra modules from the package Scipy, arrays and matrix operations were handled by using the package Numpy. The code is represented in the flowchart in Fig. 14 and it works in the following order:

1. Read the mesh file from gmsh by using the python package *meshio* and storing the nodes coordinates, connectivity matrix and boundary nodes and elements in a mesh object implemented in this work. The mesh class also has the smoothing technique implemented as a method;

Figure 13 - Speed comparison between Cython assembly function and Python assembly function. N is the number of elements and the time is normalized by the maximum Python time value.



2. Define an initial velocity field according to the problem being solved and making sure that it satisfies the necessary boundary conditions;

3. Calculate the initial vorticity field from the initial velocity by solving the equation:

$$\mathbf{M}\omega = \mathbf{G_x}v_y - \mathbf{G_y}v_x; \tag{110}$$

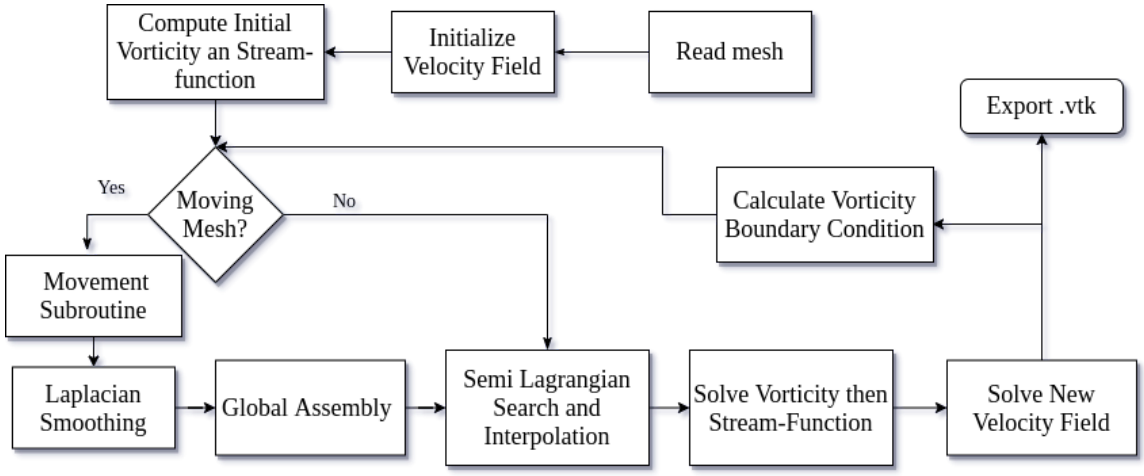
4. The initial stream function is then obtained from the initial vorticity by solving

$$\mathbf{K}\psi = \mathbf{M}\omega; \tag{111}$$

5. Calculate the mesh velocities from a defined movement function in case of a moving boundary simulation and from calling the Laplacian smoothing method. After that, the mesh nodes are displaced accordingly and a new global assembly of the FEM matrices is necessary;

6. Call the semi Lagrangian function by giving the mesh, the known vorticity field and the ALE velocity as arguments and with that find the vorticity values in the departure position;

Figure 14 - Flowchart of the algorithm developed to solve the stream function-vorticity formulation in the moving mesh context.



7. Solve the vorticity transport equation:

$$\left(\frac{\mathbf{M}}{\Delta t}+\frac{\mathbf{K}}{Re}\right)\omega^{n+1}=\mathbf{M}\frac{\omega_d^n}{\Delta t}; \tag{112}$$

8. Solve the stream function equation with the new vorticity with the equation:

$$\mathbf{K}\psi=\mathbf{M}\omega; \tag{113}$$

9. Get the new velocity field from the new stream function by solving

$$\mathbf{M}v_x=\mathbf{G_y}\psi\,,\quad \mathbf{M}v_y=-\mathbf{G_x}\psi; \tag{114}$$

10. Export the results in a vtk file that is used for visualization in the open source software Paraview and end simulation if necessary;
11. Calculate the vorticity boundary conditions with Eq.110;
12. Repeat from step 5.

5 NUMERICAL RESULTS

This chapter presents the different results obtained with the new implemented FEM solver for the stream function-vorticity formulation using the semi Lagrangian method in an ALE reference frame.

5.1 Code Verification

For every implemented numerical code it is necessary to verify that results obtained are in accordance with the literature and with the expected method behavior. This section brings three well known cases, the Zalesak disk test, the lid driven cavity flow and the flow between parallel plates.

5.1.1 Zalesak Disk Test

This test is a standard benchmark found in the literature (see Zhang and Fogelson (2014) for example) and it aims in evaluating numerical errors that may occur in highly advective problems.

It consists of placing a solid disk, with the dimensions shown in Fig. 15, in a purely advective flow, i.e. as $Re \to \infty$. The flow is given by the stream function in Eq.115 and the expected solution is the rotation of the disk around the center of the domain.

Our main goal is to study the semi Lagrangian method's implementation and behavior so the mesh was kept fixed for all the cases, and the parameters used are presented in Tab. 1. The simulation starts by defining a scalar function that is equal to 1 for the nodes inside the disk geometry and 0 outside, then this function is transported by the semi Lagrangian method.

$$\psi(x,y) = -\frac{\omega_z}{2}\left[(x-x_o)^2 + (y-y_o)^2\right] \tag{115}$$

where (x_o, y_o) is the center of rotation.

Figure 15 - Geometry of the Zalesak disk and the finest mesh used in the simulations

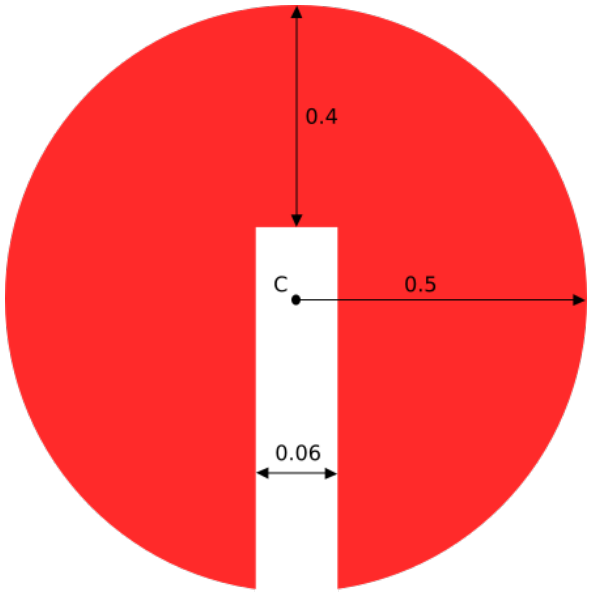


Table 1 - Parameters used for all the Zalesak disk simulations.

Parameters	
Computational Domain	[0, 4] x [0, 4]
Center of Disk	(2.0, 2.75)
Center of Rotation	(2.0, 2.0)
Angular velocity ω_z	0.5

Table 2 - Error between initial and final position of the Zalesak disk after a full revolution. The values were calculated by the L^2 norm. The number of elements in each mesh are: Coarse - 1476; Intermediate - 4926; Fine - 10214.

Mesh	Iterations		
	200	500	1000
Coarse	$7.478x10^{-3}$	$7.559x10^{-3}$	$7.584x10^{-3}$
Intermediate	$3.339x10^{-3}$	$3.440x10^{-3}$	$3.469x10^{-3}$
Fine	$2.074x10^{-3}$	$2.186x10^{-3}$	$2.216x10^{-3}$

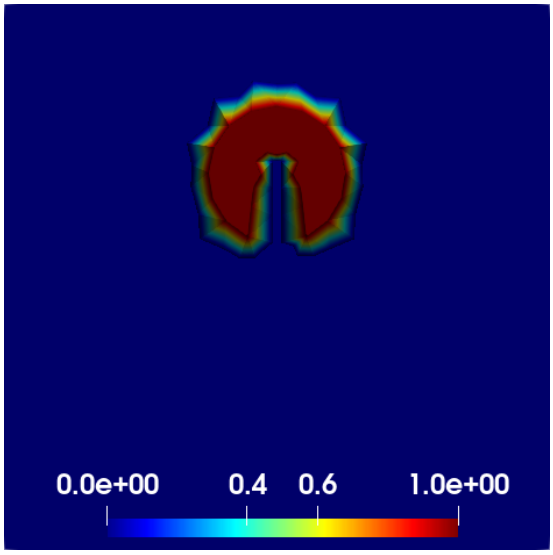
Three different mesh sizes and Δt were used for a total of 9 simulations. The meshes had: 1476 , 4926 , 10214 elements. The time step was calculated based on one full revolution divided in 200, 500 and 1000 iterations.

Fig.16, Fig.17, Fig.18 shows the solution for the coarser, intermediate and finer mesh, respectively, in 5 different instants: initial, 1/4, 1/2, 3/4 and a full revolution.

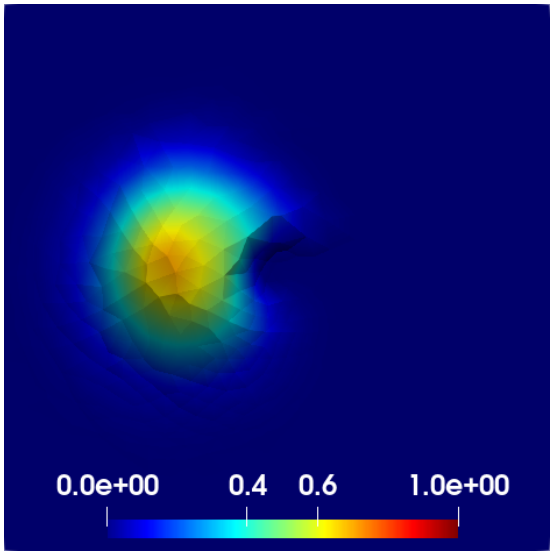
To compare the results, the L^2 norm presented in Eq. 116 was used to measure the error between the solution in the initial and final positions. This comparison is shown in Tab. 2 where it is noticeable that the error decreases when increasing the the number of elements. However, by reducing Δt the error increases and this happens because neither smaller nor greater time steps are associated with the accuracy of the method as explained by Mortezazadeh and Wan (2019). The interpolation truncation error depends on the time step, local velocity and local grid and can grow differently for each direction.

$$\epsilon = \sqrt{\frac{1}{N}\sum_{i}^{N}(x_i - \bar{x_i})^2} \tag{116}$$

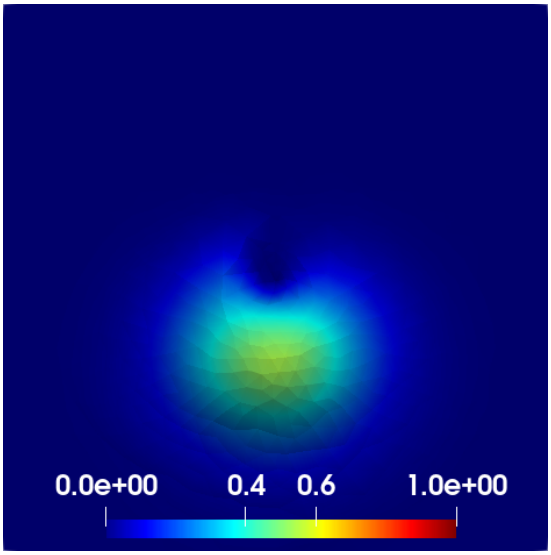
Figure 16 - A Full Revolution of the Zalesak disk for a coarse mesh with 1476 elements. The simulation was done in 200 iterations.



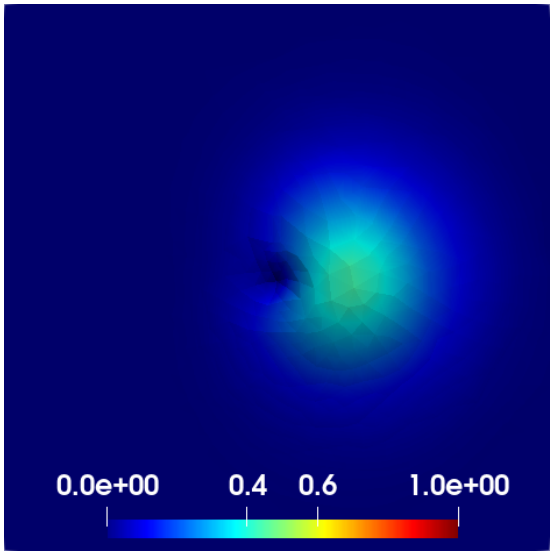
(a) Initial Position



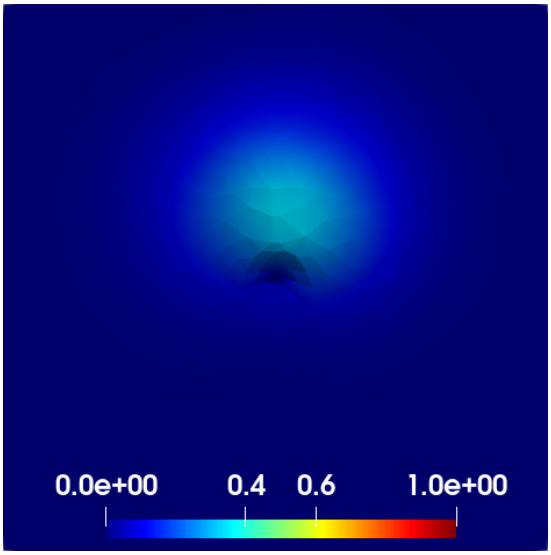
(b) 1/4 Revolution



(c) 1/2 Revolution

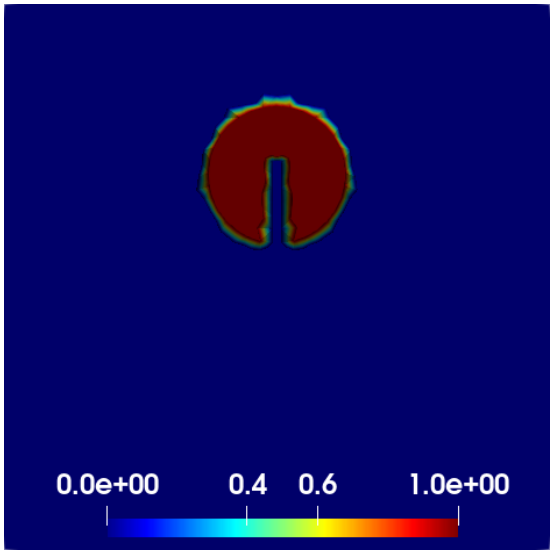


(d) 3/4 Revolution

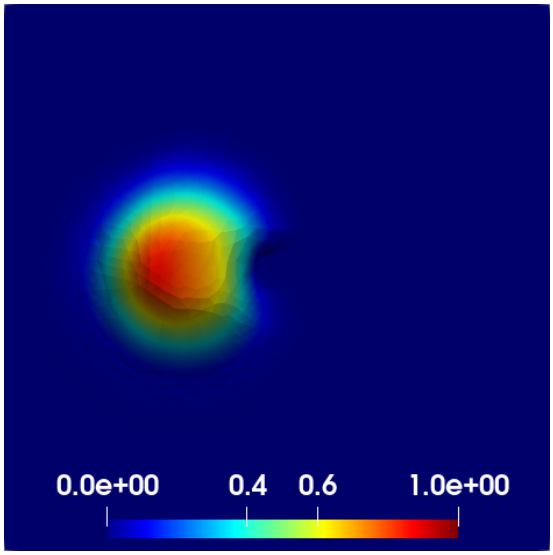


(e) Full Revolution

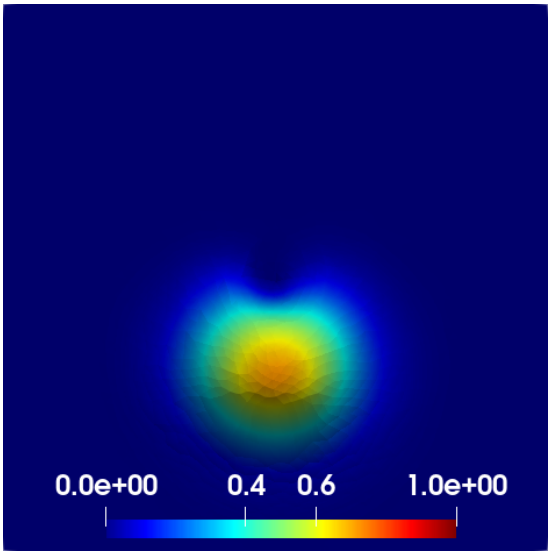
Figure 17 - A Full Revolution of the Zalesak disk for an intermediate mesh with 4926 elements. The simulation was done in 200 iterations.



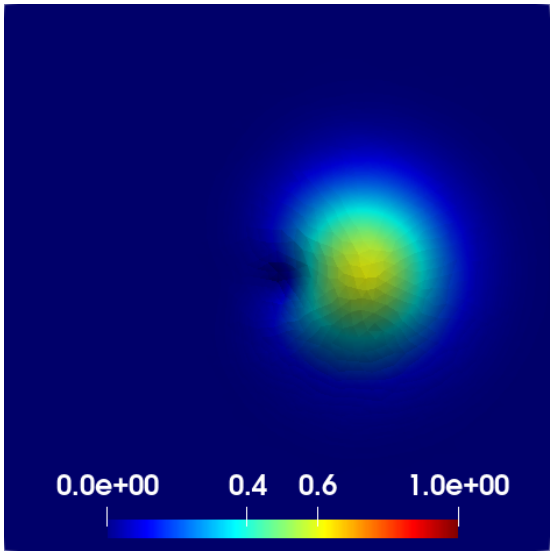
(a) Initial Position



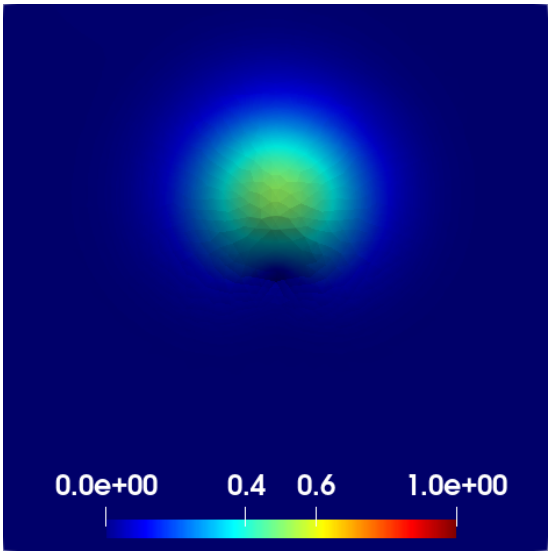
(b) 1/4 Revolution



(c) 1/2 Revolution

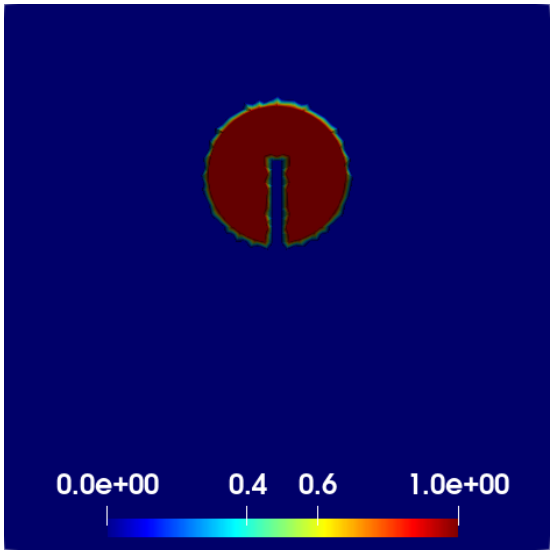


(d) 3/4 Revolution

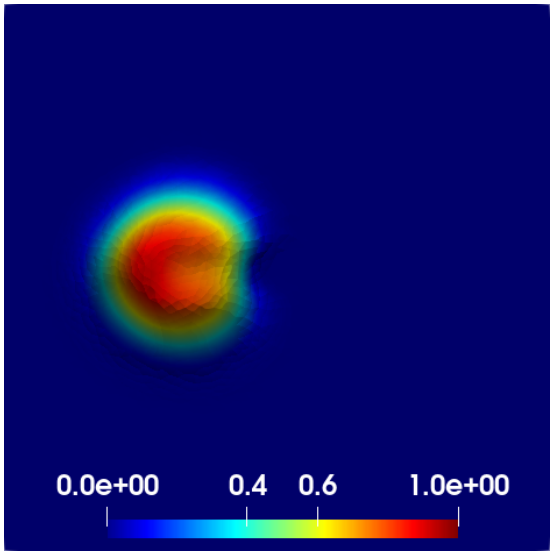


(e) Full Revolution

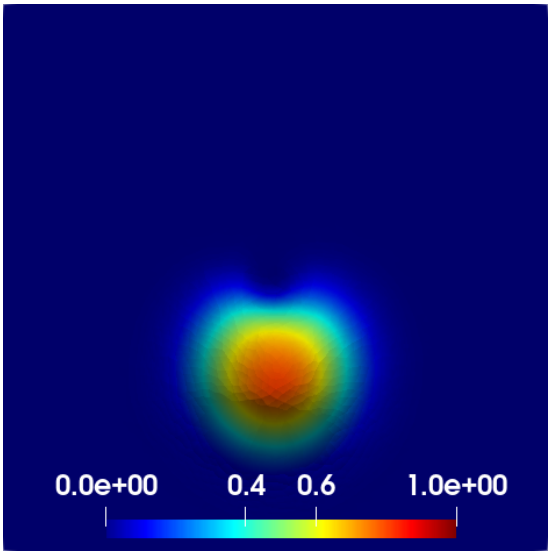
Figure 18 - A Full Revolution of the Zalesak disk for a fine mesh with 10214 elements. The simulation was done in 200 iterations.



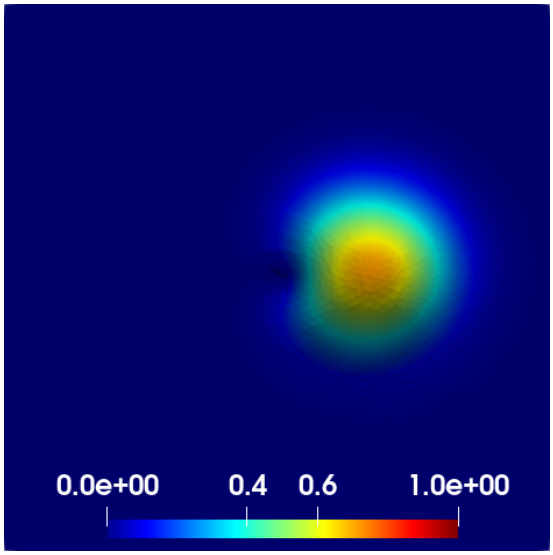
(a) Initial Position



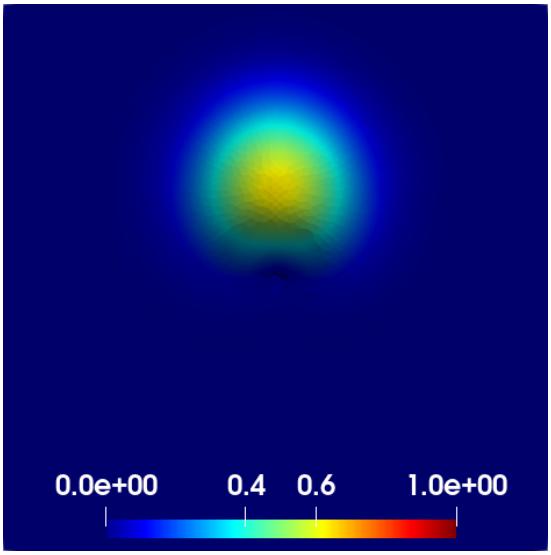
(b) 1/4 Revolution



(c) 1/2 Revolution

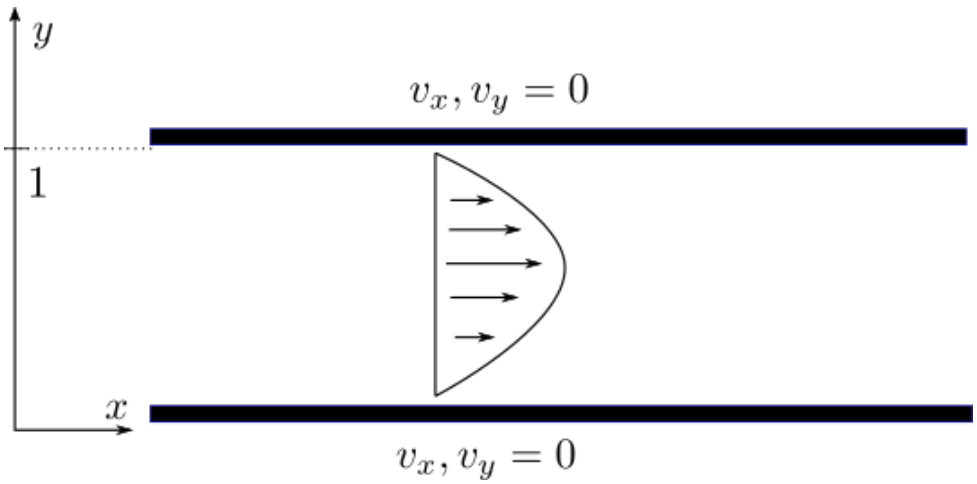


(d) 3/4 Revolution



(e) Full Revolution

Figure 19 - Schematic representation of the flow between parallel plates.



5.1.2 **Flow Between Parallel Plates**

The flow between parallel plates is a well known result in fluid mechanics. This was chosen as a verification test because it is possible to compare the numerical results obtained with the analytical solutions of the velocity, vorticity and stream function fields. A schematic representation of the problem with the boundary conditions is presented in Fig. 19 and the expected solutions in Eqs. 117, 119 and 118.

$$\begin{cases} v_y = 0 \\ v_x = 6\left(y - y^2\right) \end{cases} \tag{117}$$

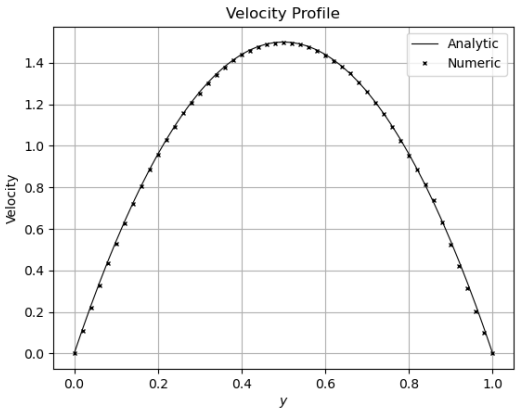
$$\psi = 3y^2 - 2y^3 \tag{118}$$

$$\omega = 12y - 6 \tag{119}$$

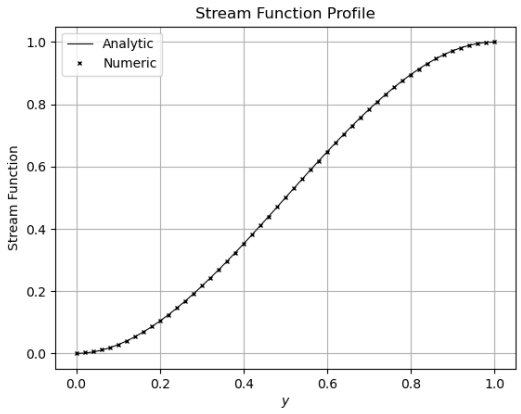
The velocity was initialized as analytical for the inflow condition and equal to zero for the rest of the domain. The stream function was set $\psi_0 = 0$ on the bottom wall and $\psi_0 = 1$ on the top wall.

This problem doesn't have moving boundaries and using the Lagrangian velocity to move the mesh causes a large deformation on the elements near the inflow region. To test the ALE algorithm, it was attributed an artificial velocity to the nodes making them oscillate with a random amplitude smaller than the element size.

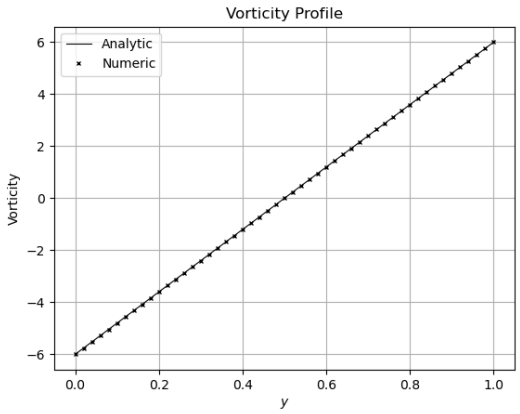
Figure 20 - Comparison between analytical and numerical solution obtained from the ALE/FE method for the flow between parallel plates. The mesh used for these results consisted of 3892 elements.



(a) Python function



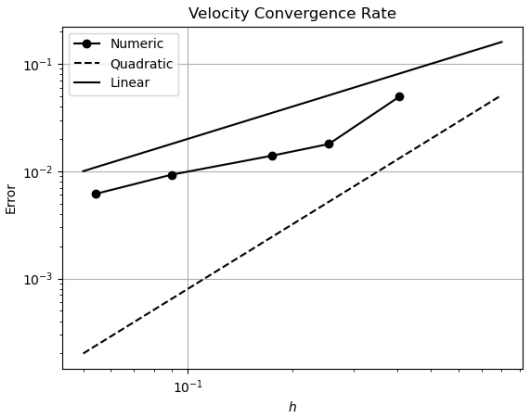
(b) Cython function



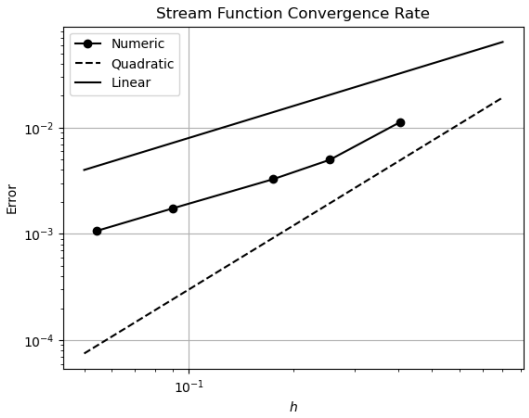
(c) Cython function

With the aim of testing the convergence of the method, 5 simulations were done with different mesh refinement. The number of elements in each case were 70, 178, 378, 1428 and 3892. The comparison between numerical and analytical solution is shown in Fig. 20 for the finest mesh case. Convergence for the stream function, vorticity and horizontal velocity is presented in Fig. 21 and it shows a linear behavior for all quantities even though the FEM with a linear element should have a quadratic convergence rate.

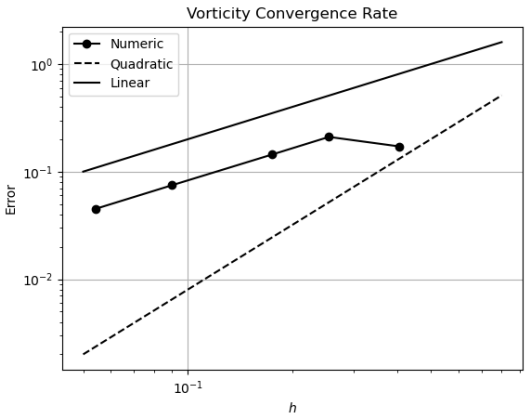
Figure 21 - Convergence rate of the velocity, stream function and vorticity solutions for the complete method simulation using five different meshes.



(a) Horizontal Velocity



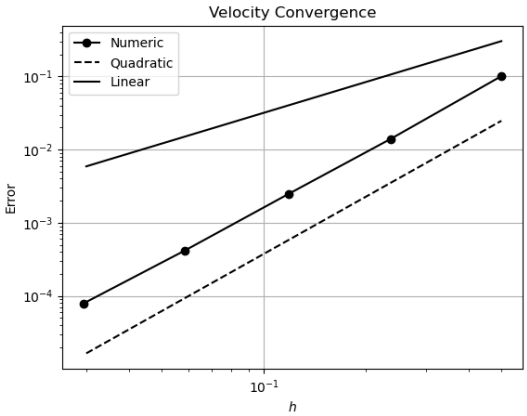
(b) Stream Function



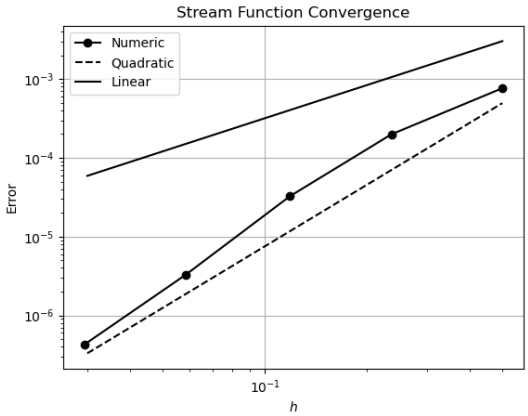
(c) Vorticity

To understand this linear convergence, we tested the solution of each equation separately by letting the other variables have their analytical value (to test the stream function, both vorticity and the velocity were set as analytical, and so on). By doing this, the quadratic convergence rate was achieved by the stream function solution and the velocity solution, however, the vorticity kept the linear rate when testing its boundary calculation and that is most likely what limits the overall convergence of the method. The convergence plots for this test are found in Fig. 22.

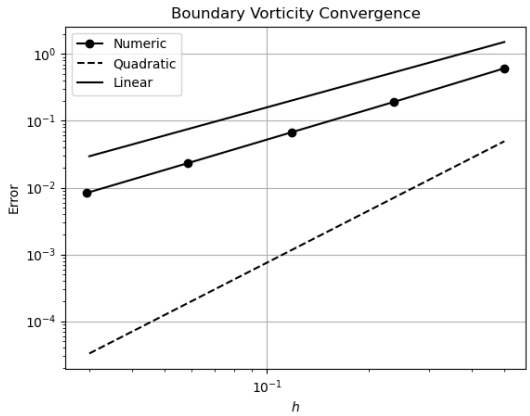
Figure 22 - Test for understanding the convergence rate. Each variable was tested separately by imposing the analytical value to the others.



(a) Horizontal Velocity

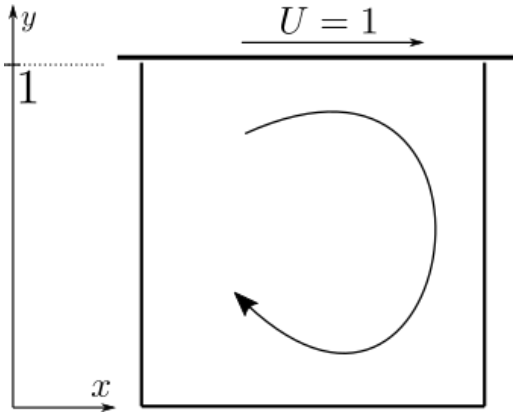


(b) Stream Function



(c) Boundary Vorticity

Figure 23 - Schematic representation of the lid driven cavity flow.

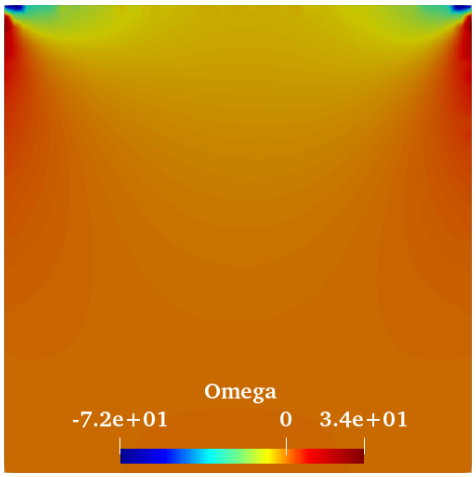


5.1.3 **Lid Driven Cavity Flow**

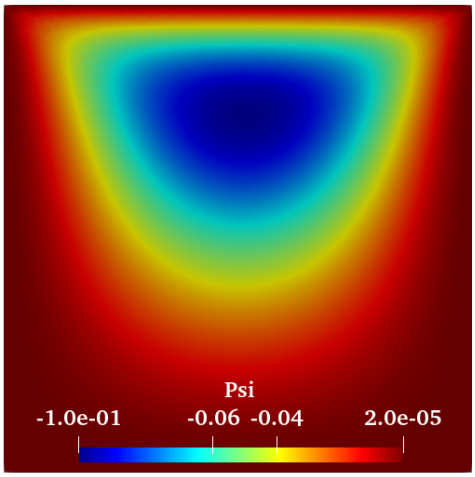
This test consists of a flow inside a square cavity (see Fig. 23) where the top wall moves with constant velocity $U = 1$ and by viscous effect makes the fluid move in a circular pattern. The stream function is set to zero along the entire boundary as there is no net flow rate between the walls.

To verify the implemented code two different Reynolds numbers were simulated, 10, 100 and the mesh used had 4416 elements. The results are shown in Figs. 24, 25 where the solutions for all the domain is presented. Figures 26 and 27 show a plot of the horizontal and vertical velocities solution over a line on the axis $y = 0.5$ and $x = 0.5$ respectively. The results show a good agreement when compared to the solutions found in the work of Marchi, Suero and Araki (2009) and Ghia, Ghia and Shin (1982).

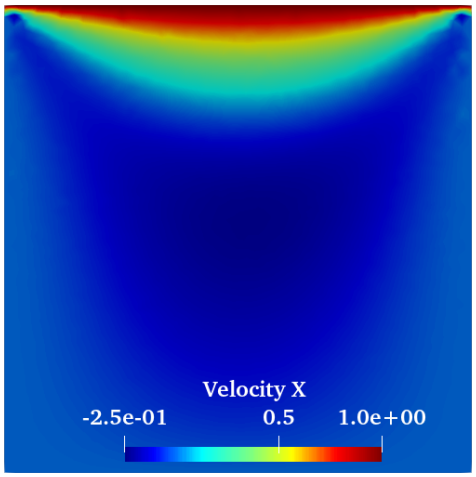
Figure 24 - Numerical solution of the lid driven cavity problem for a mesh with 4416 elements and $Re = 10$.



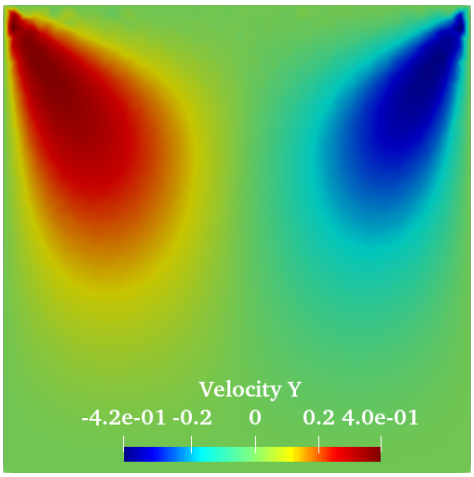
(a) Vorticity



(b) Stream Function

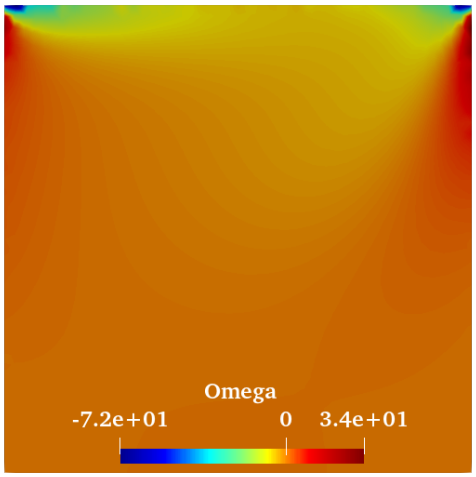


(c) Horizontal Velocity

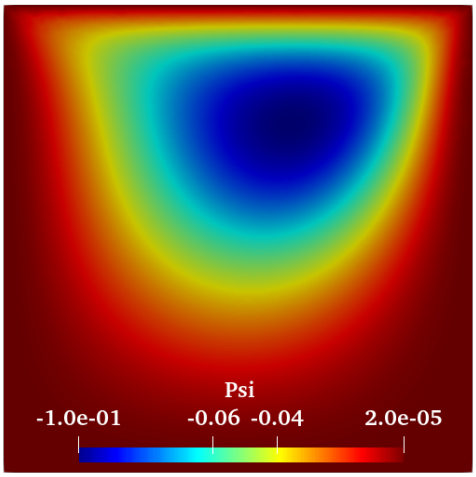


(d) Vertical Velocity

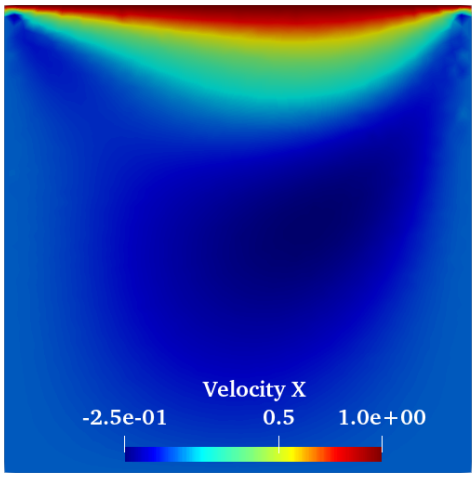
Figure 25 - Numerical solution of the lid driven cavity problem for a mesh with 4416 elements and $Re = 100$.



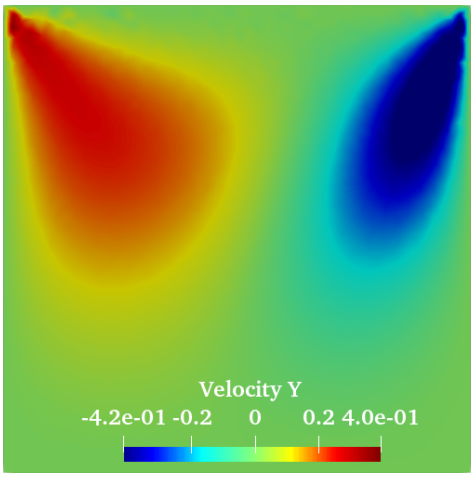
(a) Vorticity



(b) Stream Function

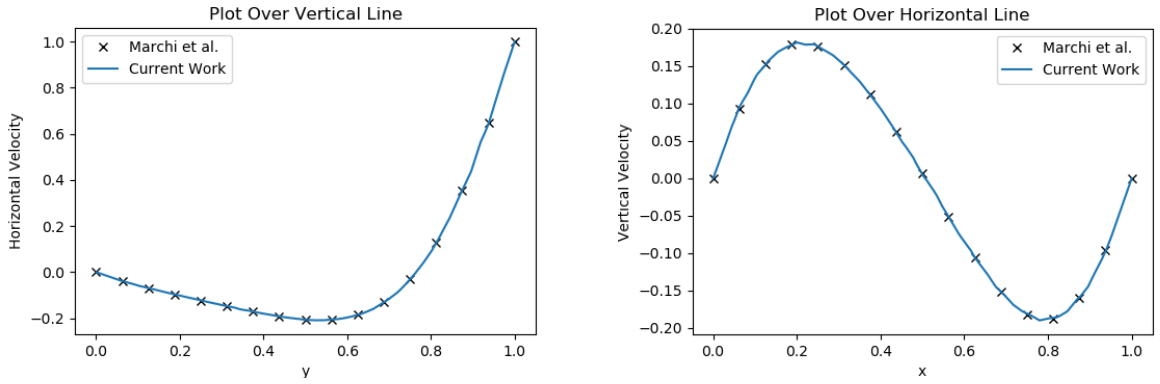


(c) Horizontal Velocity



(d) Vertical Velocity

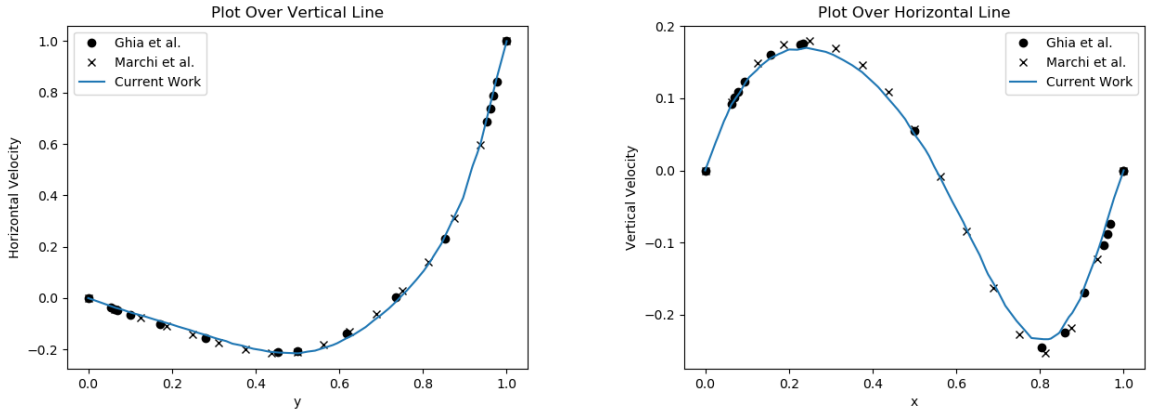
Figure 26 - Comparison between the ALE/FE solution with a solution found in the literature for the velocity field in the lid driven cavity problem with $Re = 10$. The plots were made along the axis $y = 0.5$ and $x = 0.5$.



(a) Horizontal Velocity

(b) Vertical Velocity

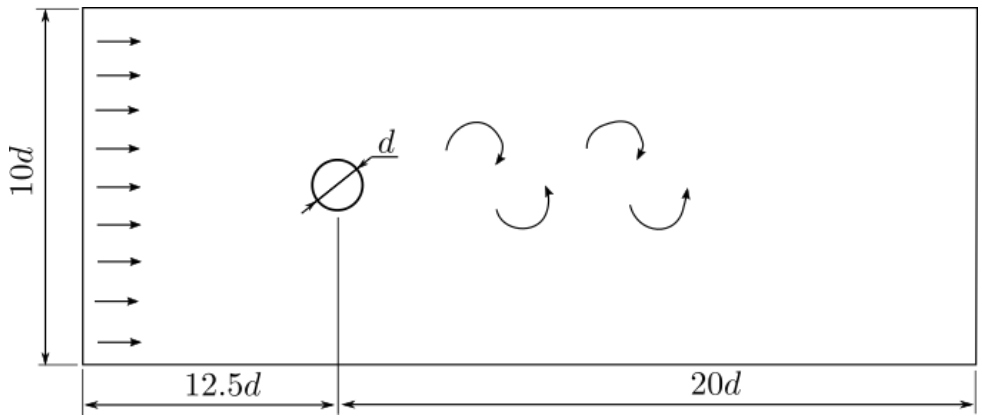
Figure 27 - Comparison between the ALE/FE solution with a solution found in the literature for the velocity field in the lid driven cavity problem with $Re = 100$. The plots were made along the axis $y = 0.5$ and $x = 0.5$.



(a) Horizontal Velocity

(b) Vertical Velocity

Figure 28 - Schematic representation of the flow around a cylinder and geometry used in the simulations.



5.2 **Flow Around Cylinder**

The flow around a cylinder is a well known problem of fluid-structure interaction where we can observe the formation of vortices along the flow wake.There are different flow regimes that can occur depending on the Reynolds number.

For $Re < 49$ the fluid achieves a stationary solution. Between $Re = 49$ and $Re = 180$ the flow is no longer stationary and starts to produce vortices behind the cylinder and the frequency of vortex formation increases with the Reynolds number. This frequency is measured by the non-dimensional Strouhal number, defined in Eq. 120. When $Re > 180$ the flow goes through another transition and the frequency tends to stay in the range $0.20 < St < 0.22$. With the intent of observing the different flow regimes, simulations were made with the following Reynolds numbers: 30, 40, 70, 120, 200.

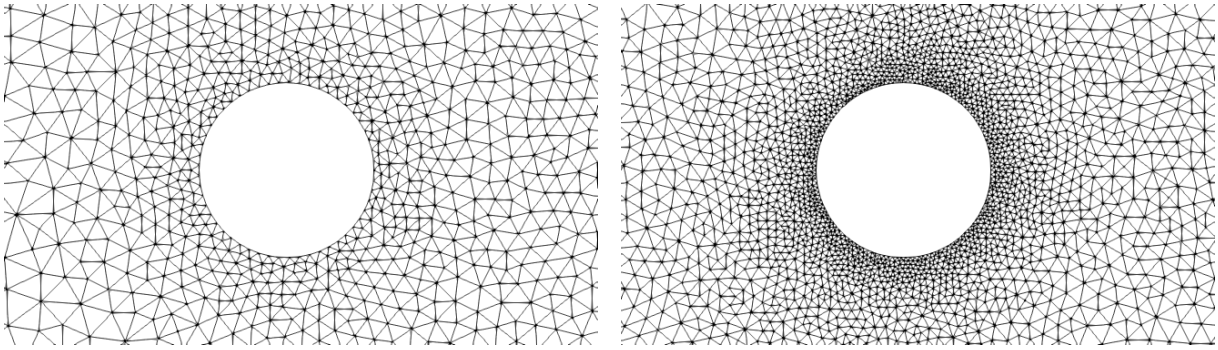
$$St = f_s \frac{d}{U_\infty} \tag{120}$$

where f_s is the Strouhal frequency and d is the cylinder diameter.

The solution of this problem is very sensitive to the computational domain and the Reynolds number, specially for low Re values and with the purpose of comparing our solution to that of Placzek, Sigrist and Hamdouni (2009) we chose the same geometry, represented in Fig. 28. The boundary conditions defined were:

- Symmetry for the top and bottom boundary;
- Inflow on the left side, with a defined constant velocity $U = 1$;

Figure 29 - Region closer to the cylinder wall of both meshes used in the simulations. (a) a coarse mesh that consisted of 53 nodes around the cylinder; (b) a fine mesh with 128 nodes around the cylinder.



(a) Coarse Mesh (b) Fine Mesh

- no-slip wall around the cylinder.

The stream function values were $\psi_0 = 0$ on the bottom, $\psi_0 = 10$ on the top and $\psi_0 = 5$ around the cylinder. To show the mesh influence on the solution, the simulations were done for a coarse mesh with 53 nodes on the cylinder and a finer mesh with 126 nodes around the cylinder and are partially shown in Fig. 29.

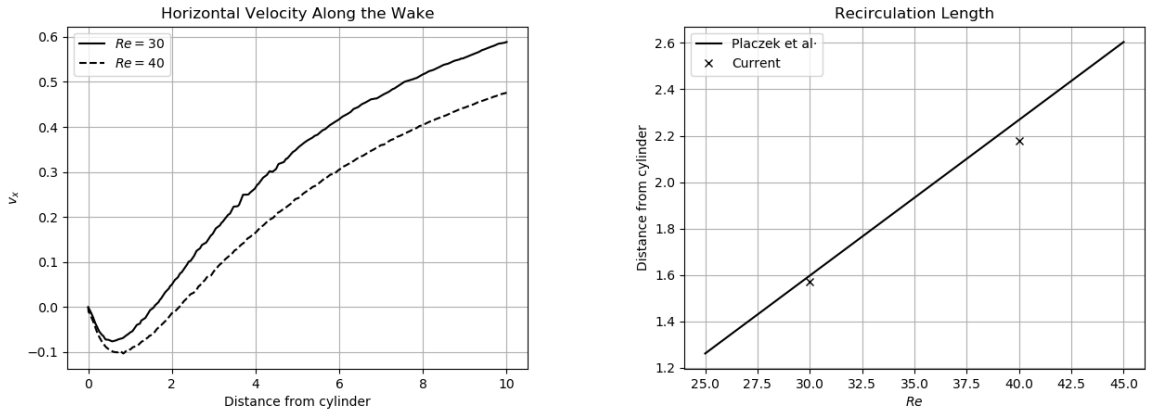
5.2.1 **Stationary Solution**

The two stationary simulations were done with $Re = 30$ and $Re = 40$ and only for the finer mesh. This flow regime is characterized by the formation of two opposing vortices behind the cylinder and they delimiter the recirculation zone, which depends on the Reynolds number. Figure 30a show that dependence by plotting the horizontal velocity v_x along a line behind the cylinder and Fig. 30b compares the recirculation length with the empirical expression $l_r = 0.0671Re - 0.4155$. Figure 31 shows the vorticity and stream function solutions for the simulation with $Re = 40$.

5.2.2 **Vortex Shedding**

The relationship between the Strouhal and Reynolds number was evaluated for both vortex shedding regime and compared to the empirical prediction given by Fey, König and

Figure 30 - Recirculation zones on the stationary solution of the flow around a cylinder and the dependence of its length with the Reynolds number. The line in (b) is an empirical relation found in the literature.



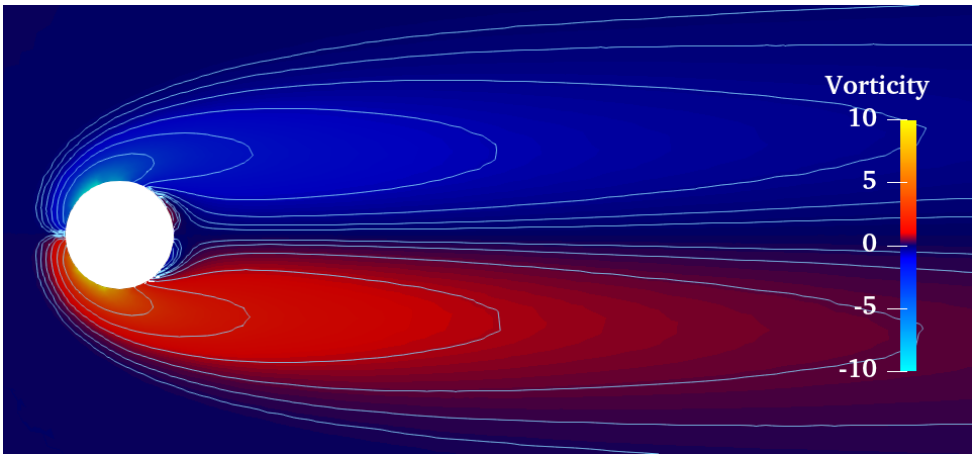
(a) (b)

Eckelmann (1998). Test with both meshes showed that the coarser mesh had a significant impact in the solution and produced frequencies lower than expected. This results are expressed in Fig. 33.

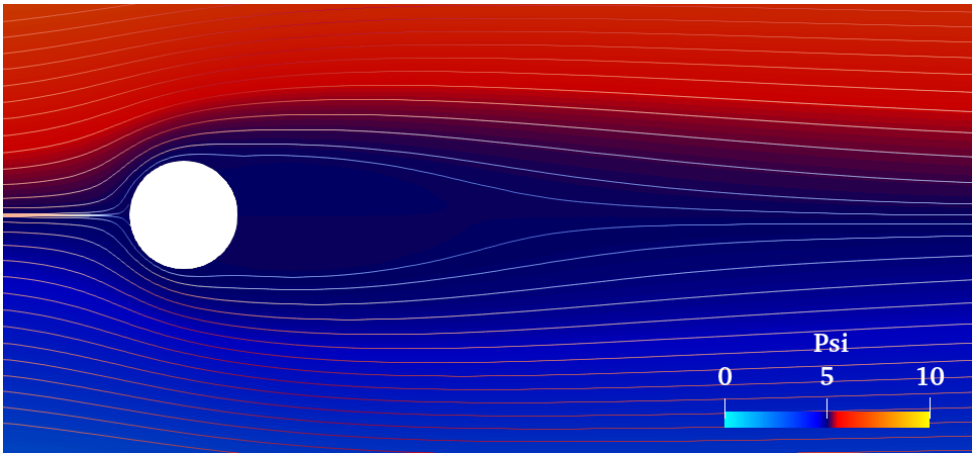
To measure the Strouhal number, a point in the near wake had its vorticity value plotted over time. From the oscillating solution, as shown in Fig. 32 for the case when $Re = 200$, the frequency is found by applying Fourier transformation and getting its dominant frequency value.

As the mechanism of vortex formation is similar for all the cases, only the vorticity and stream function solutions for $Re = 200$ are presented in Fig. 34 and Fig. 35 respectively. Both figures show the evolution over a full shedding period $T_s = 1/f_s$.

Figure 31 - Vorticity and stream function solutions on the wake of a cylinder for the stationary solution when $Re = 40$.



(a) Vorticity



(b) Stream Function

Figure 32 - Numerical solution in time of the vorticity oscillation at a point in the cylinder wake for $Re = 200$.

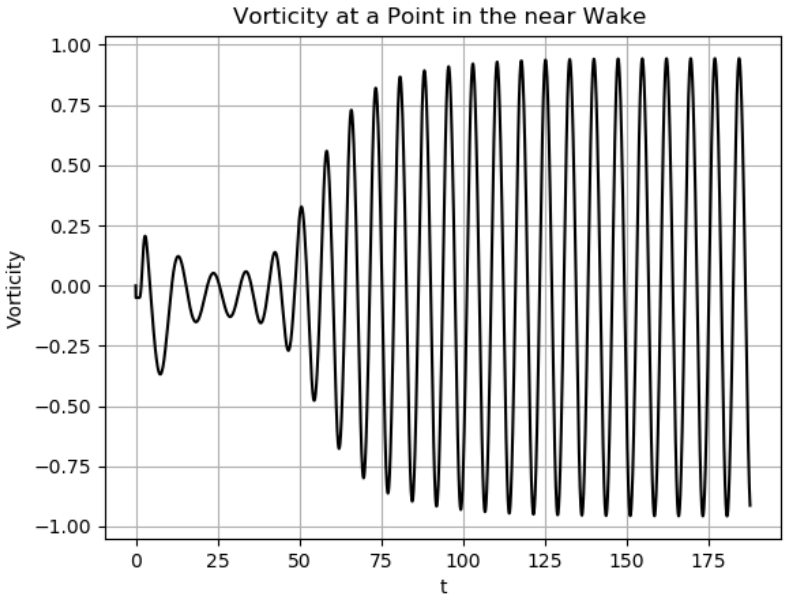


Figure 33 - Relationship between the Strouhal and Reynolds number for the vortex shedding in the cylinder wake. The mesh showed a large influence on the results obtained. The line shows an empirical relation found in the literature.

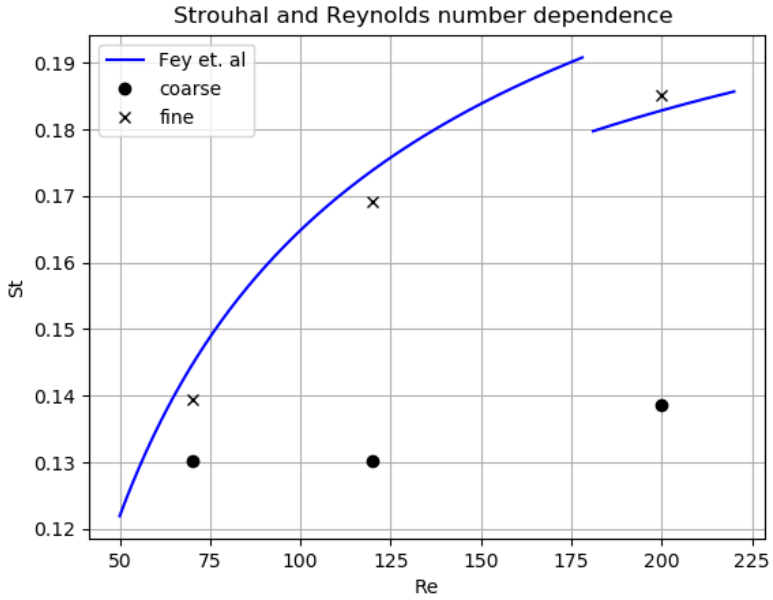
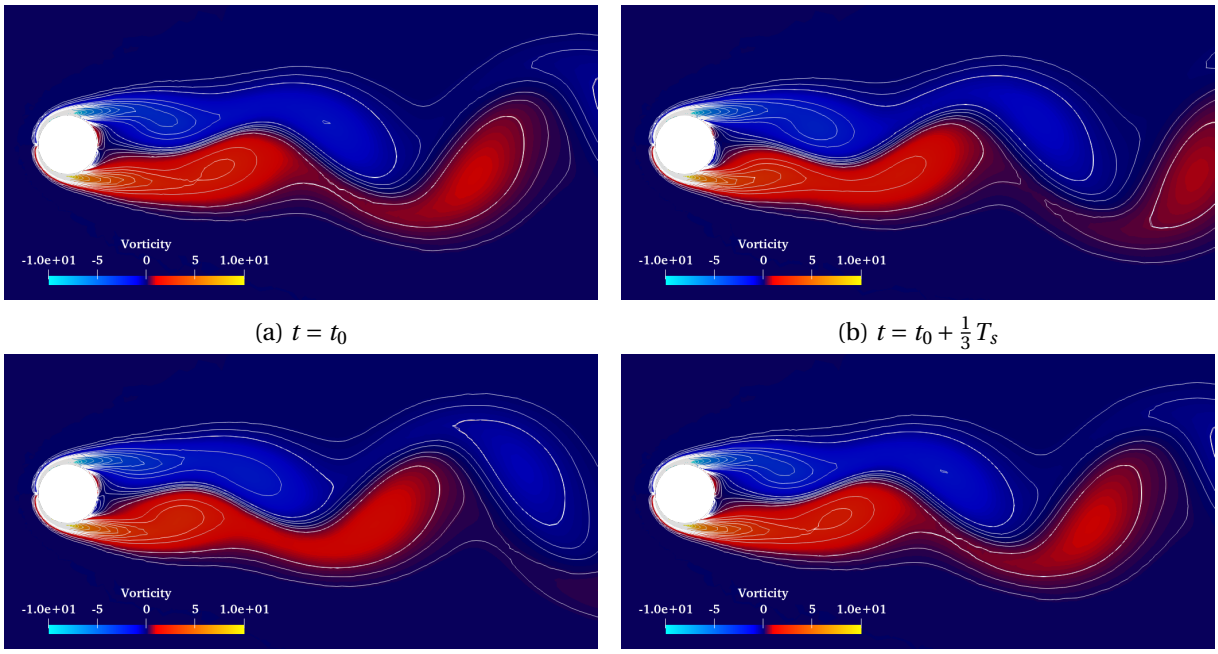


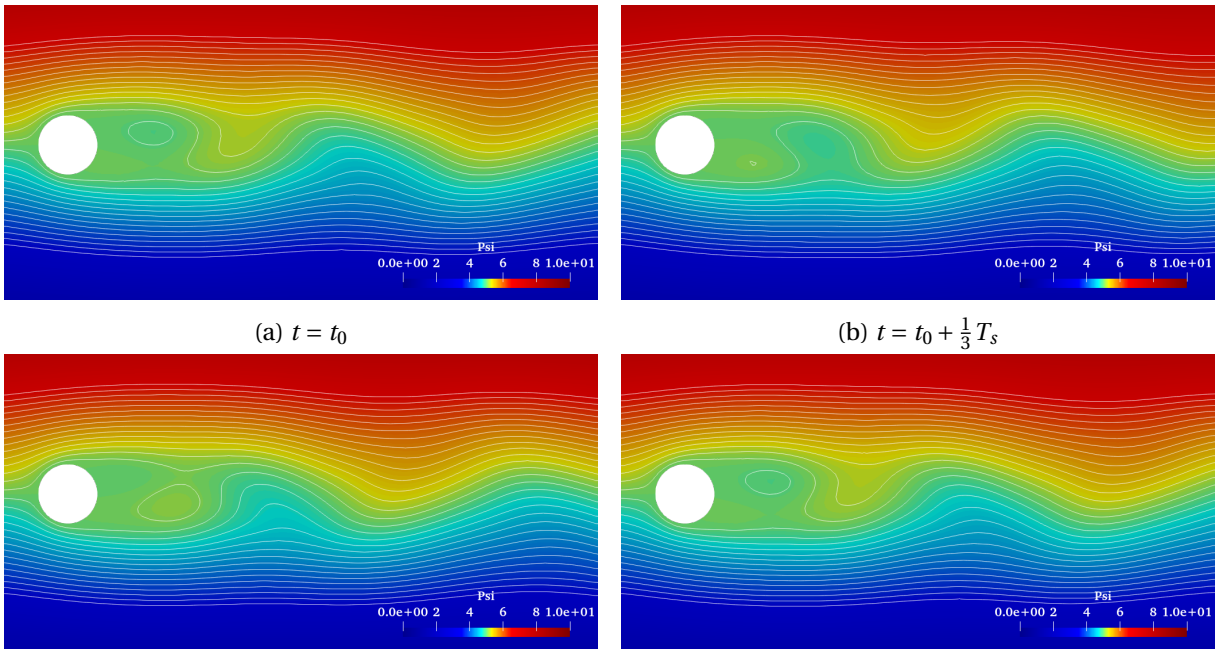
Figure 34 - ALE/FE solution of the vorticity field for the flow around a cylinder with $Re = 200$ during a full vortex shedding period.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_s$

(c) $t = t_0 + \frac{2}{3}T_s$ (d) $t = t_0 + T_s$

Figure 35 - ALE/FE solution of the stream function field for the flow around a cylinder with $Re = 200$ during a full vortex shedding period.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_s$

(c) $t = t_0 + \frac{2}{3}T_s$ (d) $t = t_0 + T_s$

5.3 **Flow Around a Moving Object**

This problem is similar to the fixed cylinder in the previous section, however the wall boundary is forced to oscillate vertically. Other geometries were also used in the simulations, and they were all done with $Re = 100$.

The boundary movement velocity is given by Eq. 121 and from that we define an internal mesh velocity based on the distance from the center of the hole geometry to guarantee that the interior nodes will keep close to the wall. This velocity is expressed in Eq. 122 and an example of the mesh movement is shown in Fig. 36 with the boundary on the start position and then on the top part of the oscillation.

$$v^b = 0.25 \sin(2\pi f_f t) \tag{121}$$

where f_f is the forced oscillation frequency.

$$v^e = v^b \left(e^d + e^{d-5} - e^{-5} \right) \tag{122}$$

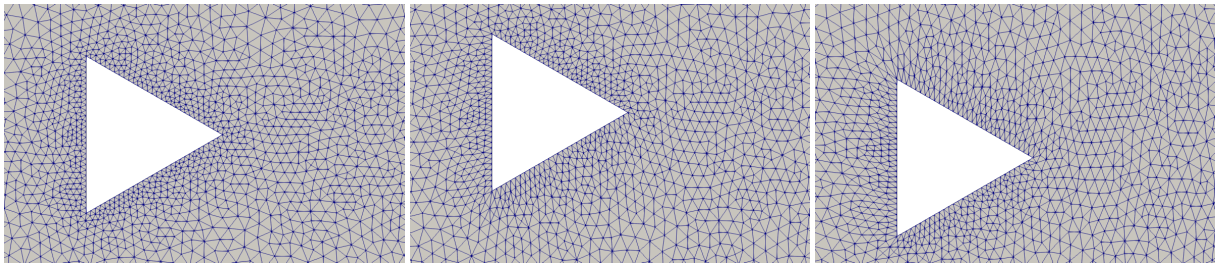
where d is the distance between the node and the geometric center of the hole boundary.

The mesh movement was done accordingly to what is described in Chapter 4 by first moving the hole boundary with the prescribed velocity v^b and also moving the interior nodes by the artificial velocity v^e, which is a function of v^b and the distance from the hole geometric center designed to have higher values near the boundary. Afterwards, the Laplacian smoothing method is called to alleviate possible large distortions and maintain mesh quality. For these simulations, the parameter used to tune the strength of the artificial velocity was $\rho_3 = 1.4$ and the smoothing velocity parameter was $\rho_2 = 0.6$.

To compare the results between the different geometries, the square and triangle were set to have the same hydraulic diameter as the cylinder. All boundary conditions were defined in the same way as the fixed cylinder case, except the stream function value around the hole wall. This value was set to change over time according to the vertical position of the boundary geometric center to keep the flow rate above and below proportional to the distance from the outer boundary. The stream function was calculated as $\psi_0 = y_c$, where y_c is the center vertical coordinate.

Table 3 shows the difference in shedding frequency between the cylinder, square and triangle for a forced oscillation frequency of 0.1. In this case, the triangle shedding frequency

Figure 36 - Region closer to the hole boundary wall of the triangle mesh being moved according to a vertical forced oscillation.



(a) Initial Position (b) Top Position (c) Bottom Position

Table 3 - Measured shedding frequency f_s for the flow around a moving object oscillating with a forced frequency $f_f = 0.1$ in a flow wit $Re = 100$

$f_f = 0.1$	Triangle	Cylinder	Square
f_s	0.1013	0.8603	0.9012

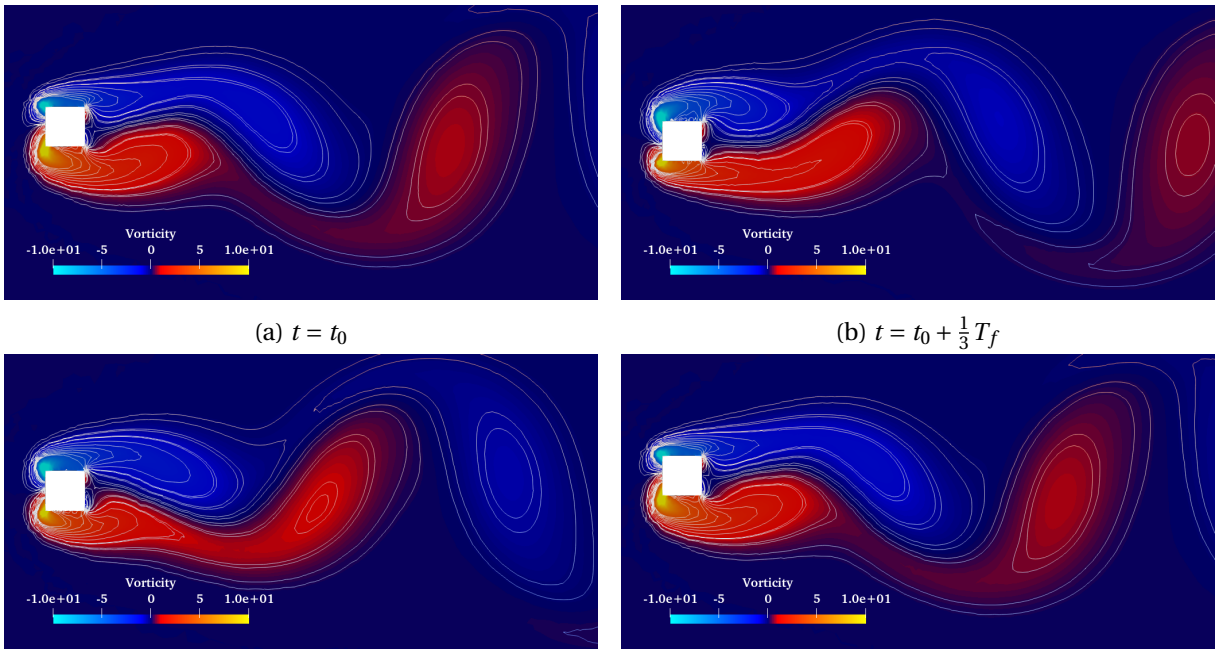
is close to the movement frequency. The cylinder and square geometries present a lower vortex formation frequency.

The results for the vorticity and stream function fields are presented in Figs. 37, 38 for the square, Figs. 39, 40 for the cylinder and Figs. 41, 42 for the triangle. They show a full oscillation period starting from the top position going down and stopping when the geometry reaches the top again.

From the results presented, it is possible to observe that the vortices formed in the square and cylinder geometries case appear to detach further away from the hole boundary when compared to the triangular geometry and possibly that is the cause for a slower shedding frequency measured. In all the cases the vortices formed an alternating pattern of clockwise and counter-clockwise rotation and always forming on the same layers respectively.

Another noticeable effect is the high vorticity concentrated at the sharp vertices of the triangle and square caused by the stagnation point that happens in those corners. Due to the shape and orientation of the triangle, the small eddies that stay right next to boundary in the near wake form over the top and bottom side and have a stronger effect on the shedding process.

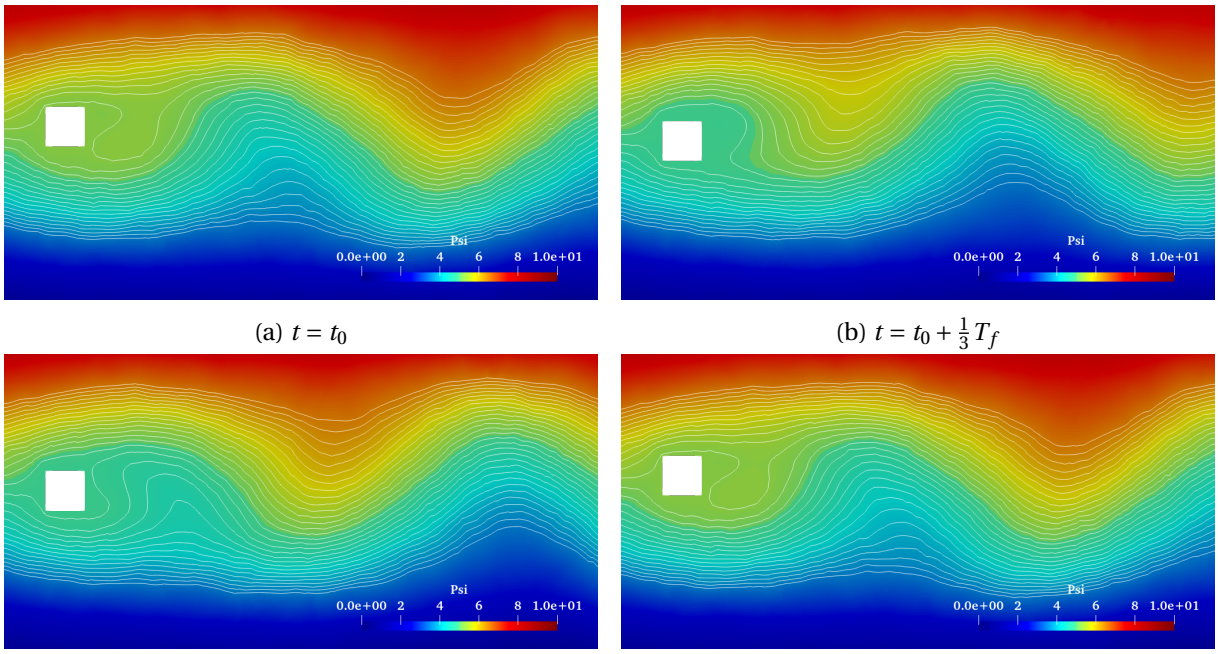
Figure 37 - ALE/FE solution of the vorticity field for the flow around a moving square with $Re = 100$ during a full oscillation period T_f.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_f$

(c) $t = t_0 + \frac{2}{3}T_f$ (d) $t = t_0 + T_f$

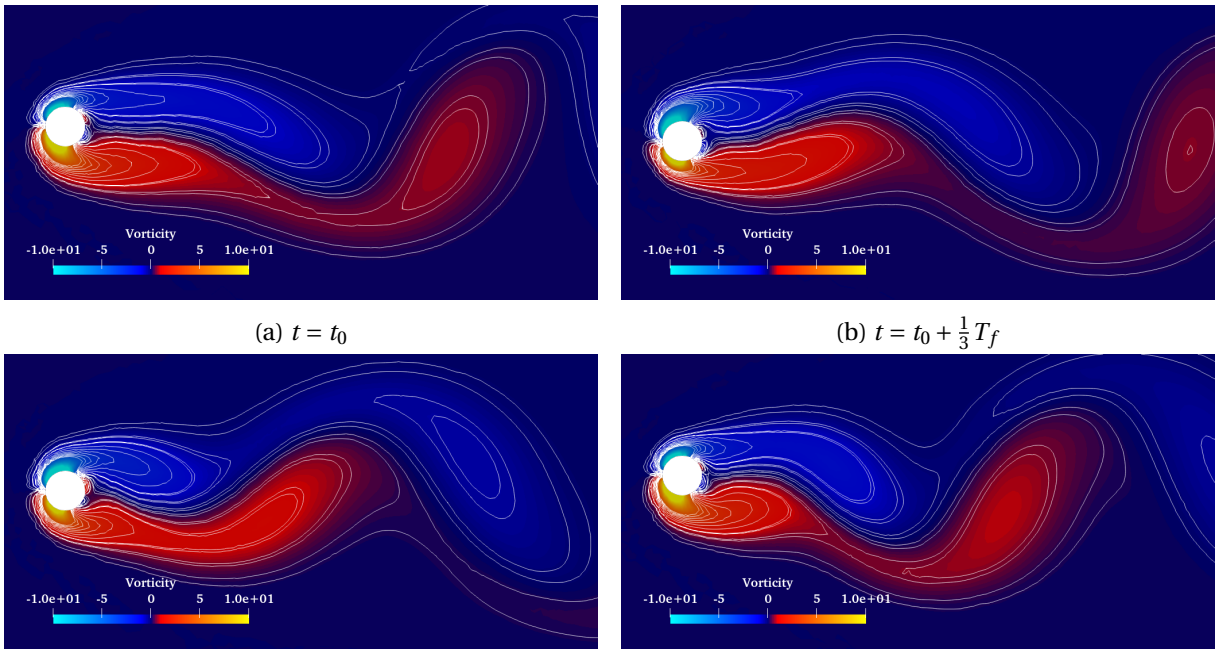
Figure 38 - ALE/FE solution of the stream function field for the flow around a moving square with $Re = 100$ during a full oscillation period T_f.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_f$

(c) $t = t_0 + \frac{2}{3}T_f$ (d) $t = t_0 + T_f$

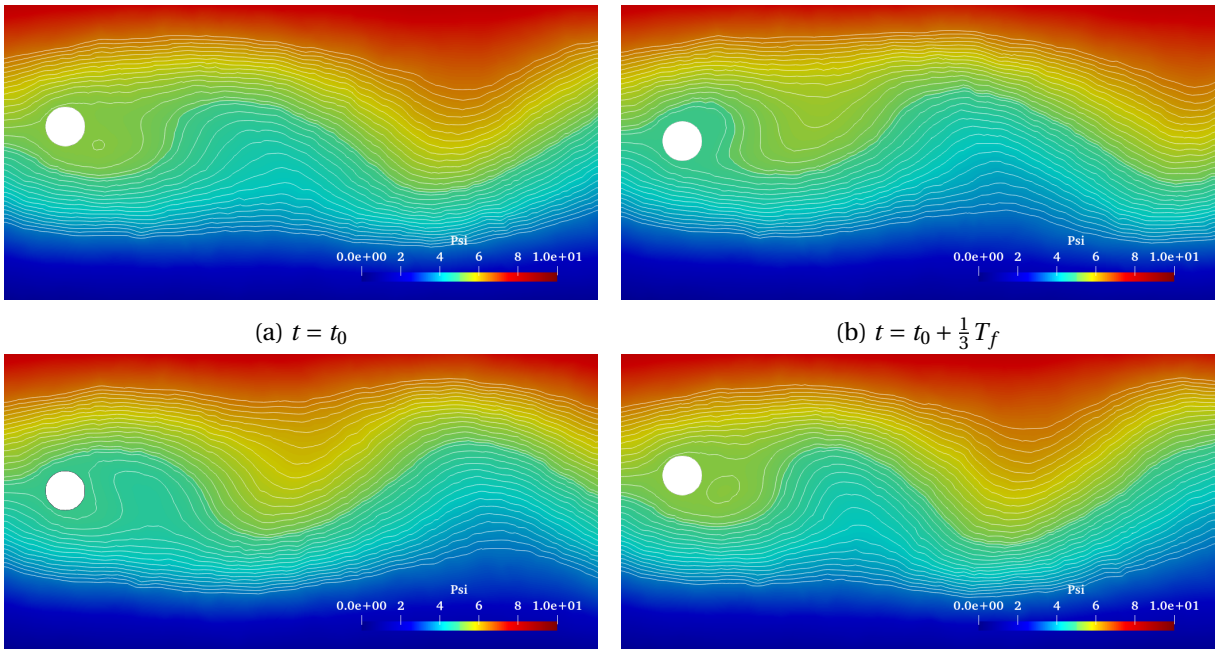
Figure 39 - ALE/FE solution of the vorticity field for the flow around a moving cylinder with $Re = 100$ during a full oscillation period T_f.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_f$

(c) $t = t_0 + \frac{2}{3}T_f$ (d) $t = t_0 + T_f$

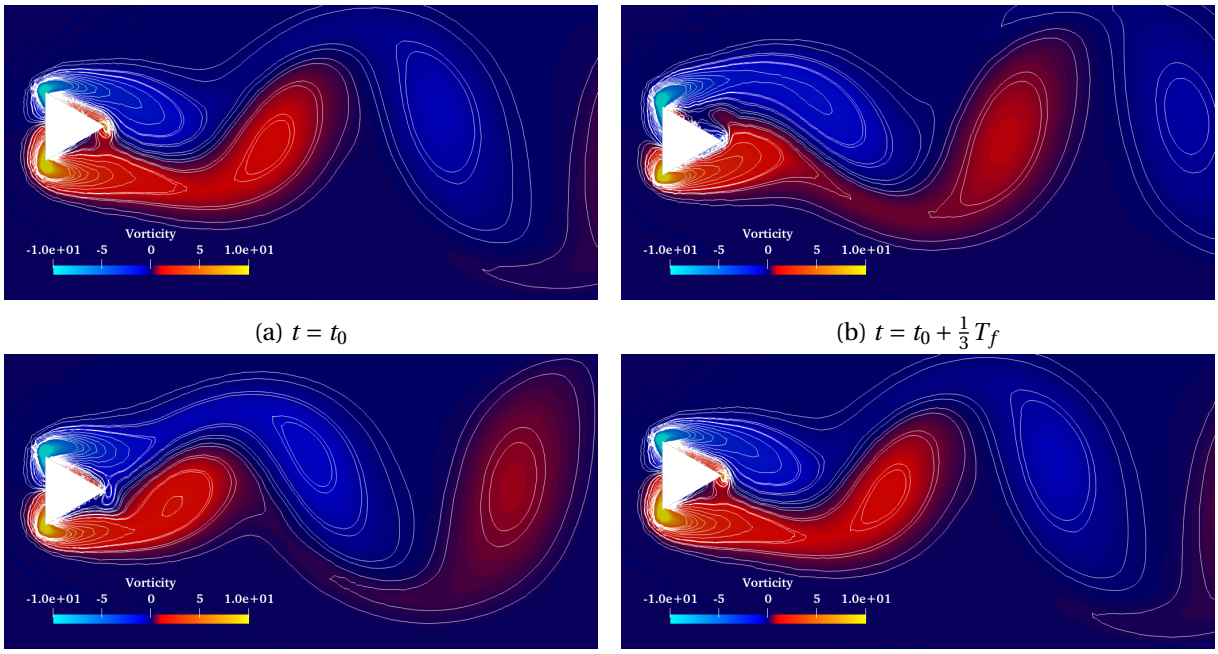
Figure 40 - ALE/FE solution of the stream function field for the flow around a moving cylinder with $Re = 100$ during a full oscillation period T_f.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_f$

(c) $t = t_0 + \frac{2}{3}T_f$ (d) $t = t_0 + T_f$

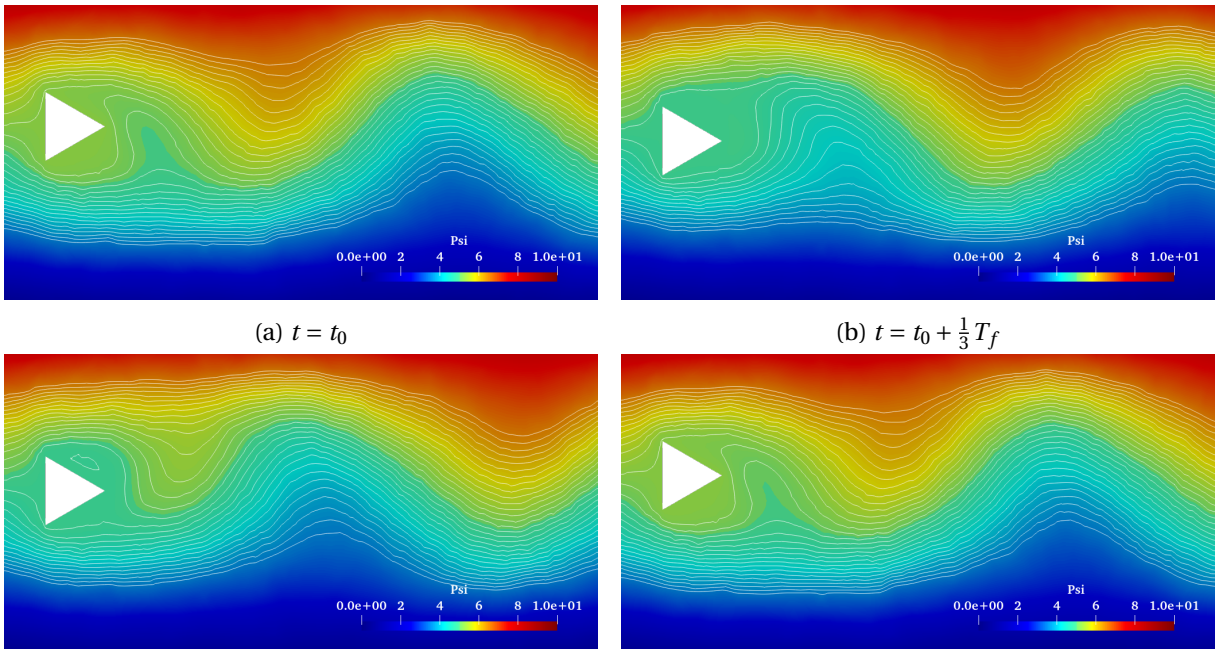
Figure 41 - ALE/FE solution of the vorticity field for the flow around a moving triangle with $Re = 100$ during a full oscillation period T_f.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_f$

(c) $t = t_0 + \frac{2}{3}T_f$ (d) $t = t_0 + T_f$

Figure 42 - ALE/FE solution of the stream function field for the flow around a moving triangle with $Re = 100$ during a full oscillation period T_f.



(a) $t = t_0$ (b) $t = t_0 + \frac{1}{3}T_f$

(c) $t = t_0 + \frac{2}{3}T_f$ (d) $t = t_0 + T_f$

CONCLUSION

This dissertation demonstrated an implementation of a Finite Elements algorithm to solve the two-dimensional stream function-vorticity formulation with the ALE frame of reference. The spatial discretization was done by the open software Gmsh using linear triangular elements. In the vorticity transport equation, the material derivative was approximated by a first order semi-Lagrangian scheme which is unconditionally stable and allows larger time steps during the simulations.

From the numerical verification tests presented, we observe that the method is only able to achieve a linear convergence rate and this happens, most likely, because of the proposed approximation of vorticity on the walls. So, to improve the overall convergence, more tests need to be done on how to estimate the values for the boundary condition condition.

A purely advective flow was evaluated in the Zalesak disk test to asses the capability of the semi-Lagrangian method. The results showed a high numerical diffusion of the solution and that this is alleviated when the mesh is refined. However, using smaller time steps do not guarantee a better accuracy of the simulation, and in fact for all the tests done, the results with a higher time step were more successful. This observation is backed by the literature and as explained by Mortezazadeh and Wan (2019) it happens because the diffusion error comes from the interpolation and this depends on the local velocities and mesh as well. To obtain a more robust implementation, the use of higher order semi-Lagrangian methods should be studied together with an adaptive time step strategy for better computational efficiency.

The main results used to showcase the method developed were the simulations of flow around different moving objects, including the classical case of flow around a cylinder. First, the fixed cylinder situation was studied for different Reynolds numbers and mesh refinements where once again, the semi-Lagrangian method proved to be too diffusive on the coarser mesh. Considering the flow around a moving object, it was noted that the triangle geometry was capable of keeping the vortex shedding frequency closer to the oscillating frequency. Both the square and cylinder presented a lower vortex shedding frequency.

Overall the ALE/FE and semi-Lagrangian method implemented in Python with optimizations done in Cython proved to be an useful alternative in simulating two-dimensional flows with the stream function-vorticity formulation especially due to its simplicity in implementation when compared to the Navier-Stokes with primitive variables. However, there are some important issues that need to be addressed with further research, namely: the need of a better estimation of the wall vorticity value to apply proper boundary conditions; need of a

higher order semi-Lagrangian scheme with an adaptive time step strategy to reduce the diffusion; explore more optimization possibilities with Cython on the current implementation.

REFERENCES

ABDELLATIF, N.; TOUIHRI, M.; AMIN, M. E. Spectral element discretization for the stream-function and vorticity formulation of the axisymmetric stokes problem. *Calcolo*, 2016.

ANJOS, G. R. et al. A 3d moving mesh finite element method for two-phase flows. *Journal of Computational Physics*, 2014.

CESINI, G. et al. Natural convection from a horizontal cylinder in a rectangular cavity. *International Journal of Heat and Mass Transfer*, 1998.

COMINI, G.; CORTELLA, G.; MANZAN, M. A streamfunction-vorticity-based finite-element formulation for laminar-convection problems. *Numerical Heat Transfer, Part B: Fundamentals*, Taylor & Francis, 1995.

COURANT, R. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, American Mathematical Society, vol. 49, 1943.

DONEA, J.; GIULIANI, S.; HALLEUX, J. P. An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 1982.

DONEA, J. et al. Arbitrary lagrangian–eulerian methods. In: ____. *Encyclopedia of Computational Mechanics*. [S.l.]: American Cancer Society, 2004. chap. 14. ISBN 9780470091357.

FEY, U.; KöNIG, M.; ECKELMANN, H. A new strouhal–reynolds-number relationship for the circular cylinder in the range $47 < re < 2 \times 10^5$. *Physics of Fluids*, 1998.

FISH, T. B. J. *A first course in finite elements*. [S.l.]: John Wiley & Sons Ltd, 2007.

FROMM, J. E.; HARLOW, F. H. Numerical solution of the problem of vortex street development. *Physics of Fluids*, 1963.

GEUZAINE, C.; REMACLE, J.-F. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 2009.

GHIA, U.; GHIA, K.; SHIN, C. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 1982.

HOFFMANN, K. A.; CHIANG, S. T. *Computational Fluid Dynamics*. Fourth. [S.l.]: Engineering Education Systems, 2000. vol. 1.

HUGHES, T. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. [S.l.]: Dover Publications, 2000. (Dover Civil and Mechanical Engineering). ISBN 9780486411811.

HUGHES, T.; LIU, W. K.; ZIMMERMANN, T. K. Lagrangian-eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 1981.

IDELSOHN, S.; OñATE, E. Finite volumes and finite elements:two 'good friends'. *International Journal for Numerical Methods in Engineering*, 1994.

LEWIS, R. W.; NITHIARASU, P.; SEETHARAMU, K. N. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. [S.l.]: John Wiley & Sons Ltd, 2004.

LO, D. C.; YOUNG, D. L. Arbitrary lagrangian–eulerian finite element analysis of free surface flow using a velocity–vorticity formulation. *Journal of Computational Physics*, 2014.

MARCHI, C. H.; SUERO, R.; ARAKI, L. K. The lid-driven square cavity flow: Numerical solution with a 1024 x 1024 grid. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 2009.

MORTEZAZADEH, M.; WAN, L. L. An adaptive time-stepping semi-lagrangian method for incompressible flows. *Numerical Heat Transfer, Part B: Fundamentals*, 2019.

NGUYEN, V.-T. An arbitrary lagrangian–eulerian discontinuous galerkin method for simulations of flows over variable geometries. *Journal of Fluids and Structures*, 2010.

PEETERS, M. F.; HABASHIT, W. G. .; DUECK, E. G. . Finite element stream function-vorticity solutions of the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 1987.

PIRONNEAU, O. On the transport-diffusion algorithm and its applications to the navier-stokes equations. *Numerische Mathematik*, 1982.

PLACZEK, A.; SIGRIST, J.-F. cois; HAMDOUNI, A. Numerical simulation of an oscillating cylinder in a cross-flow at low reynolds number: Forced and free oscillations. *Computers & Fluids*, 2009.

PONTES, J.; MANGIAVACCHI, N. *Fenômenos de Transferência*. 1. ed. [S.l.: s.l.], 2010. vol. 1.

TEZDUYAR, T. E.; GLOWINSKI, R.; LIOU, J. Petrov-galerkin methods on multiply connected domains for the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 1988.

TEZDUYAR, T. E.; LIOU, J. On the downstream boundary conditions for the vorticity-stream function formulation of two-dimensional incompressible flows. *International Journal for Numerical Methods in Fluids*, 1991.

TURNER, M. J. et al. Stiffness and deflection analysis of complex structures. *Journal of the Aeronautical Sciences*, 1956.

VERSTEEG, H.; MALALASEKRA, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. 2nd. ed. [S.l.]: Prentice Hall, 2007.

VYNNYCKY, M. et al. Forced convection heat transfer from a flat plate: the conjugate problem. *International Journal of Heat and Mass Transfer*, 1998.

WILLIAMSON, C. H. K.; ROSHKO, A. Vortex formation in the wake of an oscillating cylinder. *Journal of Fluids and Structures*, 1988.

ZHANG, Q.; FOGELSON, A. Fourth-order interface tracking in two dimensions via an improved polygonal area mapping method. *SIAM Journal on Scientific Computing*, 2014.

ANNEX A – Conference Publications

Annexed in the following pages of this dissertation is a paper published and presented in the 25^{th} ABCM International Congress of Mechanical Engineering that contains some early stages of this work development. In the paper, an axisymetrical form of the stream function-vorticity formulation is used to solve a conjugated heat transfer problem with the semi-Lagrangian method.

25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

SEMI-LAGRANGIAN METHOD APPLIED IN THE CONJUGATED HEAT PROBLEM USING AN AXISYMMETRICAL STREAM FUNCTION-VORTICITY FORMULATION

L. H. CARNEVALE [1]
N. MANGIAVACCHI [2]
Rio de Janeiro State University - UERJ - Rua Fonseca Teles, 121, 20940- 903, São Cristóvão, Rio de Janeiro, RJ, Brazil
[1] lh.carnevale@gmail.com
[2] norberto@uerj.br

G. R. ANJOS
COPPE/Universidade Federal do Rio de Janeiro, Departamento de Engenharia Mecânica, Centro de Tecnologia, Ilha do Fundão, Rio de Janeiro, Brasil
gustavo.rabello@mecanica.coppe.ufrj.br

Abstract. *Conjugated heat transfer problems are the ones where it is necessary to understand how heat convection in a fluid influences the temperature in solid regions. Instead of solving the Navier-Stokes equation with its primitive variables, we make use of the stream function-vorticity formulation. In this paper we propose using the Semi-Lagrangean method for solving the couple problem of the stream function-vorticity formulation. A possible application of the method is also demonstrated using the calculated velocity field from the formulation in the heat transport equation to study the temperature distribution in a incompressible single-phase fluid medium as a conjugate heat transfer problem.*

Keywords: *Conjugated Heat Transfer, Stream Function-Vorticity, Semi-Lagrangian Method*

1. INTRODUCTION

The main goal of this work is the implementation of an in-house code that enables us to study various applications of conjugate heat problems. The algorithm implemented is based on (Salih, 2013) for the stream function-vorticity formulation and the finite elements method (FEM) was chosen for the simulations. The FEM is implemented by discretizing the governing equations using the Galerkin method to all terms except the convective one which is discretized by the Semi-Lagrangean method bringing stability even for higher Reynolds flow as in Anjos (2012).

Conjugate heat problem describe how heat is transfered in a domain where there is an interaction between a solid body and and a fluid.Understanding how the flow of a fluid can influence heat transport in different applications is important for optimizing processes, such as the cooling of electronic components when using a system where a refrigerant flows through channels between the components (this type of heat exchange can be seen in Szczukiewicz (2012)).

The stream function-vorticity formulation is an alternative way for expressing the equation not with the primitive variables (velocity and pressure) but in terms of the stream function ψ and vorticity ω (Peeters *et al.*, 1987). The biggest advantage of this formulation is the removal of the pressure-velocity coupling problem. However, this approach is best suited for two dimensional models (Hoffmann and Chiang, 2000) since 3D simulations require the solution of six equations.

An example of the finite element method apllied to the stream function-vorticity formulation used in a conjugate heat problem is shown in Cesini *et al.* (1998). Boundary conditions for the vorticity are traditionally based on finite-difference schemes for computing its value which limits their use to regular domains as is shown in Vynnycky *et al.* (1998). In this paper we use the FEM to calculate the vorticity on the boundary by its relation to the velocity components, Comini *et al.* (1995) also uses the FEM to compute the vorticity values on the baundary, however they use its relation to the stream function. Abdellatif *et al.* (2016) shows an application of the axisymmetric stream function-vorticity formulation in a variational form.

2. METHODOLOGY

2.1 Governing Equations

The governing equations for the stream function-vorticity formulation in cylindrical coordinates for an axisymmetric flow (see Fig. 1), as found in Panton (1996), are:

$$\frac{D\omega}{Dt} = \frac{\omega v_r}{r} + \nu\left[\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial r\omega}{\partial r}\right) + \frac{\partial^2\omega}{\partial z^2}\right] \tag{1}$$

$$\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r} \tag{2}$$

$$\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \psi}{\partial r}\right) + \frac{\partial^2}{\partial z^2}\left(\frac{\psi}{r}\right) = -\omega \tag{3}$$

$$(v_z\ ,\ v_r) = \left(\frac{1}{r}\frac{\partial \psi}{\partial r}\ ,\ -\frac{1}{r}\frac{\partial \psi}{\partial z}\right) \tag{4}$$

Equation 1 is the transport of the vorticity where r is the radius, z is the coordinate along the symmetry axis, t is the time variable, ν is the kinematic viscosity, ω_θ is the vorticity component in the θ direction, and v_r de radial velocity. Equation 2 is the definition of vorticity as a function of the velocity components, v_z being the axial component.

Equation 3 is the relation between the stream function ψ and the vorticty, this equation is deduced by the definition of the stream function in Eq. 4 and Eq. 2.

The axisymmetric temperature equation for a medium with thermal diffusivity α and the material derivative operator are, respectively:

$$\frac{DT}{Dt} = \alpha\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{\partial^2 T}{\partial z^2}\right] \tag{5}$$

$$\frac{D(\cdot)}{Dt} = \frac{\partial(\cdot)}{\partial t} + v_r\cdot\frac{\partial(\cdot)}{\partial r} + v_z\frac{\partial(\cdot)}{\partial z} \tag{6}$$

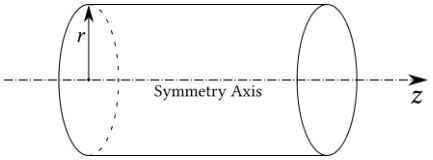


Figure 1: Schematic representation of axisymmetric coordinates. The numerical domain is a 2D plane discretized by a set of triangles.

2.2 Finite Element Formulation

Let $\Omega \in R^2$ be the problem domain with boundary Γ, and $\Omega_f \in \Omega$ with boundary Γ_f be the region in which flow occurs. From the Sobolev space $H^1(\Omega) = \left\{u \in L^2(\Omega),\ \ \frac{\partial u}{\partial x_i} \in L^2(\Omega), i = 1, 2, \cdots, n\right\}$ we define the sub-spaces:

$$\mathbb{S}_\psi(\Omega_f) = \left\{\psi \in H^1(\Omega), \psi = \psi_0 \ \texttt{on}\ \Gamma_f\right\} \tag{7}$$

$$\mathbb{S}_\omega(\Omega_f) = \left\{\omega \in H^1(\Omega), \omega = \omega_0 \ \texttt{on}\ \Gamma_f\right\} \tag{8}$$

$$\mathbb{S}_T(\Omega) = \left\{T \in H^1(\Omega), T = T_0 \ \texttt{on}\ \Gamma\right\} \tag{9}$$

$$\mathbb{V}(\Omega_f) = \left\{u | u \in H^1(\Omega), u = v_0 \ \texttt{on}\ \Gamma_f\right\} \tag{10}$$

$$\mathbb{H}^1_0(\Omega) = \left\{w | w \in H^1(\Omega), w = 0 \ \texttt{on}\ \Gamma\right\} \tag{11}$$

The weak formulation of the problem can be written as: find $\psi \in \mathbb{S}_\psi(\Omega_f)$, $\omega \in \mathbb{S}_\omega(\Omega_f)$, $T \in \mathbb{S}_T(\Omega)$ and $(v_r, v_z) \in \mathbb{V}(\Omega_f) \times \mathbb{V}(\Omega_f)$, such that

$$m(\phi, \frac{D\omega}{Dt}) - m_v(\phi, v_r, \omega) + \nu k(\phi, \omega) + \nu m_3(\phi, \omega) = 0 \tag{12}$$

$$k(\phi, \psi) + 2g_r(\phi, \psi) - m_2(\phi, \omega) = 0 \tag{13}$$

$$(m(\phi, v_r),\ m(\phi, v_z)) - (-g_z(\phi, \psi),\ g_r(\phi, \psi)) = 0 \tag{14}$$

$$m(\theta, \frac{DT}{Dt}) + \alpha k(\theta, T) = 0 \tag{15}$$

for all weight functions $\phi \in \mathbb{H}^1_0(\Omega_f)$ and $\theta \in \mathbb{H}^1_0(\Omega)$.

The discretization is done by the Galerkin method to all the terms except on the term containing the material derivative, which is discretized by the semi-Lagranian method. For an element $\Omega^e \in \Omega$ with shape functions $\mathbf{N}(r, z)$, the local operators from the weak formulation are:

$$\mathbf{m} = \int_{\Omega^e} \mathbf{N}_i \mathbf{N}_j r dr dz \tag{16}$$

$$\mathbf{k} = \int_{\Omega^e} \nabla \mathbf{N}_i \nabla \mathbf{N}_j r dr dz \tag{17}$$

$$\mathbf{m}_v = \int_{\Omega^e} \mathbf{N}_i \mathbf{N}_j v_r r dr dz \tag{18}$$

$$\mathbf{g}_r = \int_{\Omega^e} \mathbf{N}_i \frac{\partial \mathbf{N}_j}{\partial r} dr dz \tag{19}$$

$$\mathbf{m}_2 = \int_{\Omega^e} \mathbf{N}_i \mathbf{N}_j r^2 dr dz \tag{20}$$

$$\mathbf{m}_3 = \int_{\Omega^e} \mathbf{N}_i \mathbf{N}_j \frac{1}{r} dr dz \tag{21}$$

$$\mathbf{g}_z = \int_{\Omega^e} \mathbf{N}_i \frac{\partial \mathbf{N}_j}{\partial z} dr dz \tag{22}$$

The set of equation for a single element in matricial form can be written as:

$$\mathbf{m}\frac{D\omega}{Dt} + (\nu\mathbf{k} + \nu\mathbf{m}_3 - \mathbf{m}_v)\omega = 0 \tag{23}$$

$$(\mathbf{k} + 2\mathbf{g}_r)\psi = \mathbf{m}_2\omega \tag{24}$$

$$\mathbf{m}\frac{D\mathbf{T}}{Dt} + \alpha\mathbf{k}\mathbf{T} = 0 \tag{25}$$

In Eq. 23, the velocity v_r is considered constant inside the element, as the average of the nodal values, to calculate de operator $\mathbf{m}_v$. The complete field is calculated with the assembly of global $\mathbf{M}$, $\mathbf{G}_r$ and $\mathbf{G}_z$ matrices:

$$\mathbf{M}(\mathbf{v}_r\ , \mathbf{v}_z) = (-\mathbf{G}_z\psi\ , \mathbf{G}_r\psi) \tag{26}$$

The vorticity boundary condition is calculated from the definition of the vorticity as a function of the velocities using the Finite Element Method with:

$$\mathbf{M}\omega = \mathbf{G}_z\mathbf{v}_r\ - \mathbf{G}_r\mathbf{v}_z \tag{27}$$

2.3 Semi-Lagrangian Method

The semi-Lagrangian method is used in the discretization of terms containing the material derivative because it improves the overall stability and therefore making it possible to use larger time steps in relation to the Galerkin discretization of those terms. In computational fluid dynamics the method has shown great efficiency, specially for cases considering high Reynolds number.

The method consists in discretizing the material derivative for a function scalar f as:

$$\frac{Df}{Dt} = \frac{f_i^{n+1} - f_d^n}{\Delta t} \tag{28}$$

where c_d^n is the function at the time n in the called departure position and c_i^{n+1} is the function at the time $n+1$ in the current position.

The departure position (x_d) is obtained by $x_d = x_i - \mathbf{v}\Delta t$, x_i being the position of the nodal points in the mesh. Knowing the result c_i^n, c_d^n is calculated by interpolation and then we solve for c_i^{n+1}.

The final set of equations discretized in space and time for a single element in matricial form is:

$$\left(\nu\mathbf{K} + \nu\mathbf{M}_3 - v_r\mathbf{M} + \frac{\mathbf{M}}{\Delta t}\right)\omega_i^{n+1} = \frac{\mathbf{M}}{\Delta t}\omega_d^n \tag{29}$$

$$(\mathbf{K} + 2\mathbf{G}_r)\psi_i^{n+1} = \mathbf{M}_2\omega_i^{n+1} \tag{30}$$

$$\left(\frac{\mathbf{M}}{\Delta t} + \alpha\mathbf{K}\right)\mathbf{T}_i^{n+1} = \frac{\mathbf{M}}{\Delta t}\mathbf{T}_d^n \tag{31}$$

2.4 Algorithm

So far, our software is being developed to solve conjugate heat problems with an uncoupled simulation of the flow and the heat transport. It works by first generating and reading a computational mesh done with the open source software Gmsh (see Geuzaine and Remacle (2009)), in which the nodes and elements of the fluid region are specified.

The respective operators needed to solve the stream function-vorticity formulation are assembled only for the fluid elements in the mesh, and to solve the heat transport the operators are assembled for all the elements.

The algorithm used to solve the stream function-vorticity formulation is based on Salih (2013). The initial step is to define a starting velocity field respecting the boundary conditions and with Eq. 27 calculate a starting vorticity field. With that, Eq. 30 is solved for the initial stream function solution.

With all the starting variables defined, the time iterations begin by calculating the vorticity at the semi-Lagrangian departure position by interpolating vorticity solution in the nodes dislocated by the flow velocity and then Eq. 29 with the result from Eq. 27 as boundary condition.

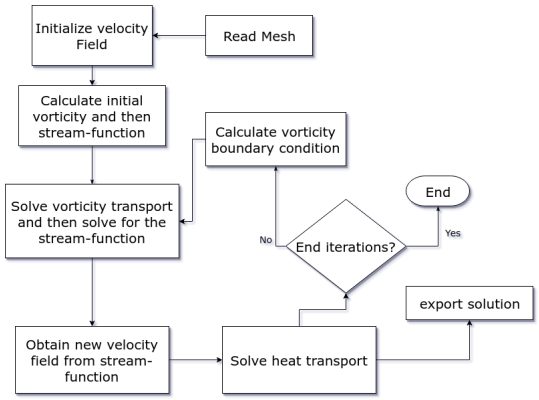


Figure 2: Summarized algorithm flow chart.

Now Eq. 30 can be solved for the new stream-function field. The heat transport is calculated by first interpolating the temperature solution at the departure nodes and then solving Eq. 31. Lastly, the results of the current time step are exported and a new velocity field is calculated with Eq. 26.

Python is th programming language being used in the development of our software and the systems of equations are solved by the linear algebra module found in the scipy library (Jones *et al.* (2001–)). The routine used in this work applies the LU decomposition to compute the solutions.

The mesh need generated needs to be composed of linear triangular elements and the integration of the local matrices is done through the Gaussian quadrtature method using weighting points. For post-processing and visualization we used the open source software Paraview (see Ahrens *et al.* (2005)). Figure 2 shows the summarized algorithm flow chart.

3. RESULTS

3.1 Zalesak Disk Test

The first proposed test cases evaluates the accuracy and stability of the semi-Lagrangian method in comparison to the standard Galerkin method for the advection in the transport equation. The Zalesak disk rotation is a standard benchmark found in the literature (see **?**, and **?**). The test consists in placing the disk in a purely rotational velocity field given by the stream function

$$\psi(x,y) = -\frac{\omega}{2}\left[(x-x_o)^2 + (y-y_o)^2\right] \tag{32}$$

where (x_o, y_o) is the center of rotation.

The simulation and geometric parameters are shown in Fig. 3 and in Tab. 1.

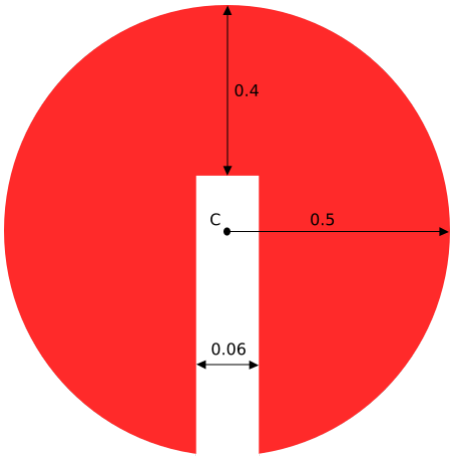


Figure 3: Initial position of Zalesak Disk rotation test

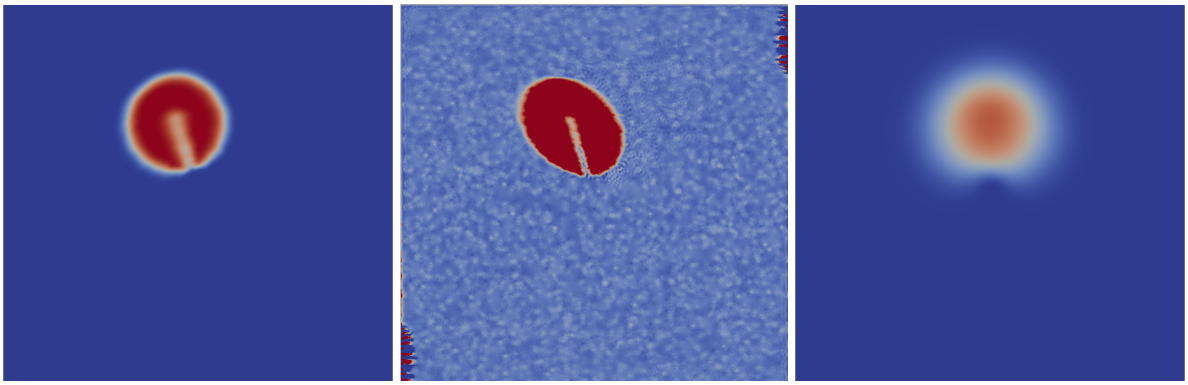
Parameters	
Computational Domain	[0, 4] x [0, 4]
Center of Disk	(2.0, 2.75)
Center of Rotation	(2.0, 2.0)
Angular velocity ω	0.5
Δt	$2\pi/1000\omega$
Number of nodes	33909
Number of elements	67816

Table 1: Parameters used for the numerical simulation

The rotation of the disk is viewed as the transport of a scalar function g given by the equation

$$\frac{Dg}{Dt} = 0 \tag{33}$$

The solutions after 50 iterations (5% of a revolution) are presented in Fig. 4 (a) and (b). The Galerkin discretization of the material derivative produces a solution with large error contaminating the results and after that the result becames highly unstable. On the other hand, the semi-Lagrangian solution is stable but its diffusive characterisc can already be observed. Figure 4 (c) shows the semi-Lagrangian solution after a full revolution.



(a) Semi-Lagrangian after 50 iterations

(b) Galerkin after 50 iterations

(c) Semi-Lagrangian after a full revolution

Figure 4: Solution of Zalesak disk rotation test for the semi-Lagrangian and the Galerkin method

3.2 Heat Conduction Validation

A heat conduction simulation was done to validate the axisymmetric formulation for the temperature equation. The test consists of a solid cylinder with radius R and length L subjected to an internal heat generation g_0 represented in Fig. 5 with the boundary conditions.

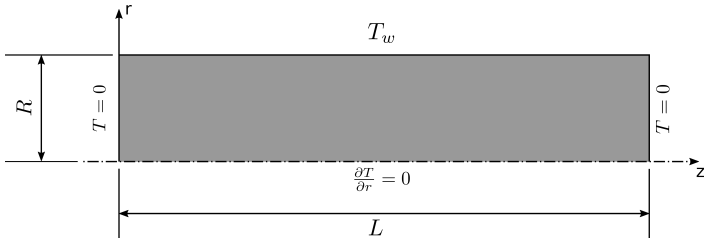


Figure 5: Geometry and boundary conditions for the heat conduction problem on a solid cylinder.

An analytic solution for this problem is presented in Hahn and Özisik (2012) and is given by:

$$T(r,z) = \sum_{n=1}^{\infty} C_n I_0(\lambda_n r)\sin(\lambda_n z) + \frac{g_0 L^2}{2k}\left[\frac{z}{L} - \left(\frac{z}{L}\right)^2\right] \tag{34}$$

with k being the thermal conductivity. The term C_n is

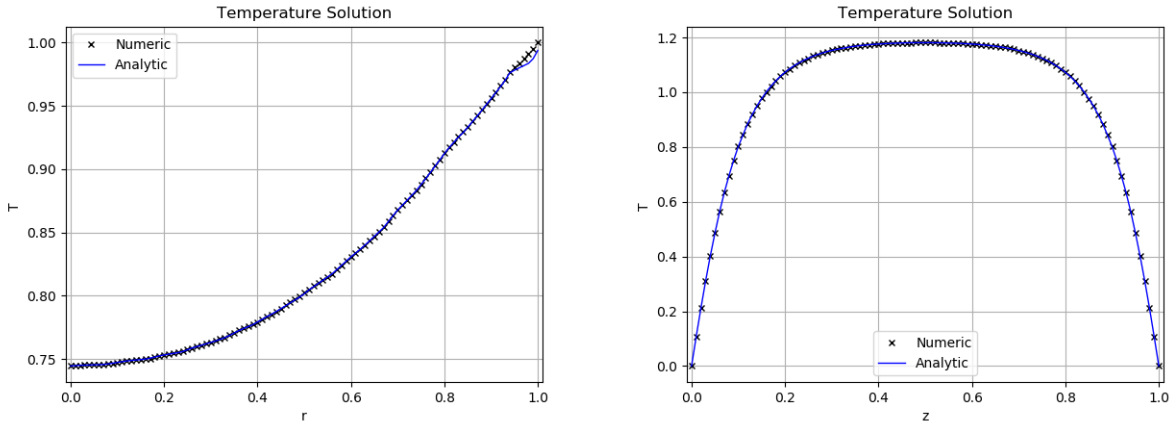
$$C_n = \frac{\int_0^L g(z)\sin(\lambda_n z)dz}{I_0(\lambda_n R)\int_0^L \sin^2(\lambda_n z)dz} \tag{35}$$

where $\lambda_n = n\pi/L$, I_0 is the modified Bessel function of the first kind with order zero and

$$g(z) = T_w - \frac{g_0 L^2}{2k}\left[\frac{z}{L} - \left(\frac{z}{L}\right)^2\right] \tag{36}$$

The parameters used for the numeric simulation were $L = 5$, $R = 1$, $g_0 = 1$, $k = 1$, and the analytic solution used to compare the results considered 100 terms of the infinite sum. The graphics shown in Fig. 6 were taken on: (a) the

radial direction with $z = 0.5$; (b) the axial direction with $r = 0.5$. It is possible to observe a good agreement between the simulation and the analytical result. Moreover, a complete solution of the temperature field is presented in Fig. 7



(a) Solution on $z = 0.5$ (b) Solution on $r = 0.5$

Figure 6: Temperature comparison between analytic and numeric solution on radius direction (a) and axis direction (b).

Figure 7: Numerical solution of the heat equation for a triangular mesh with 1357 nodes and 2712 elements. The color represents the temperature with blue being the lowest value and red the highest.

3.3 Conjugated Heat Problem

The conjugated heat problem shown in Fig. 8 represents a hollow cylinder with a thick wall (solid region) and internal temperature T_{wall} being cooled by the flow in an annular tube enclosing it. The fluid has temperature T_{in} on $z = 0$. Both left an right sides of the cylinder and the outside wall of the flow section are considered thermally isolated.

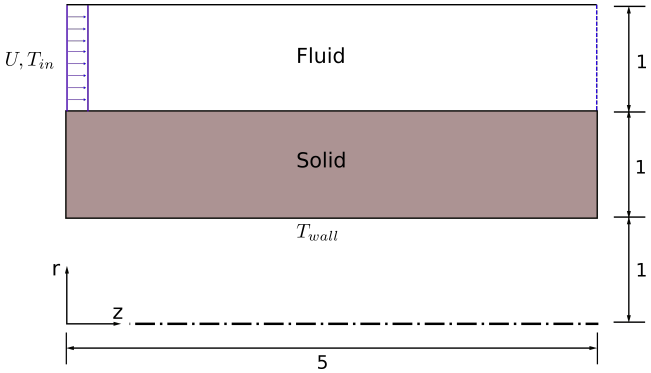


Figure 8: Proposed conjugate heat problem.

In this problem it is possible to compare the flow simulation to an analytic result given by the Hagen-Poiseuille flow in an annular section (see Batchelor (2000)). The analytical solution for the velocity field is:

$$v_z(r) = \frac{G_p}{4\mu}(R_{in}^2 - r^2) + \frac{G_p}{a\mu}(R_{out}^2 - R_{in}^2)\frac{\log(r/R_{in})}{\log(R_{out}/R_{in})} \quad , \quad v_r = 0 \tag{37}$$

where G_p is a constant pressure gradient in the axial direction, μ is the fluid viscosity, R_{in} and R_{out} are the inner and outer radii respectively. The stream-function and vorticity analytical solutions can be obtained by solving Eq. 2 and Eq. 4 with the given velocity field and a given boundary condition for ψ.

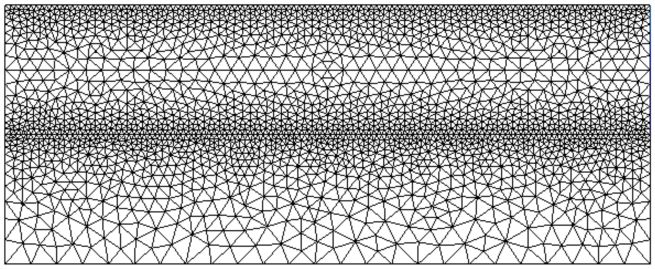
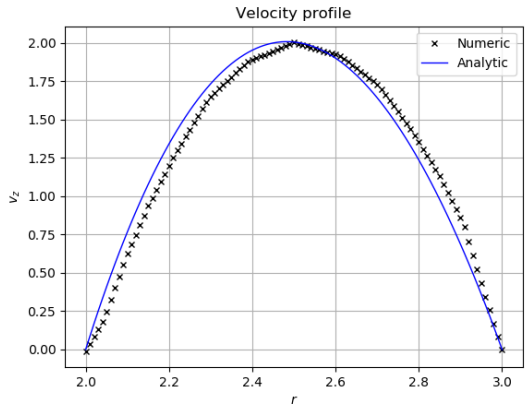


Figure 9: Mesh used in the simulation of proposed conjugate heat problem.

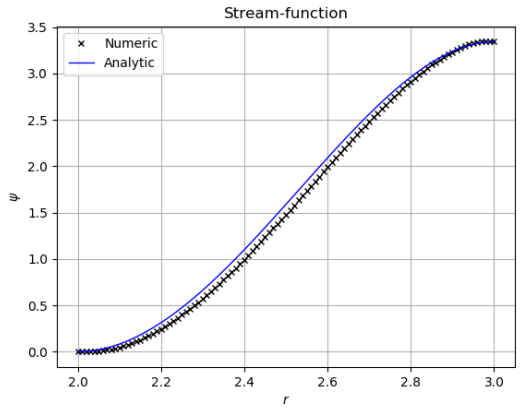
Viscosity	$\mu = 1$
Fluid density	$\rho = 1000$
Fluid thermal diffusivity	$\alpha_f = 0.001$
Solid thermal diffusivity	$\alpha_s = 0.05$
Δt	0.05
Number of nodes	2328
Number of elements	4779

Table 2: Parameters used for the numerical simulation

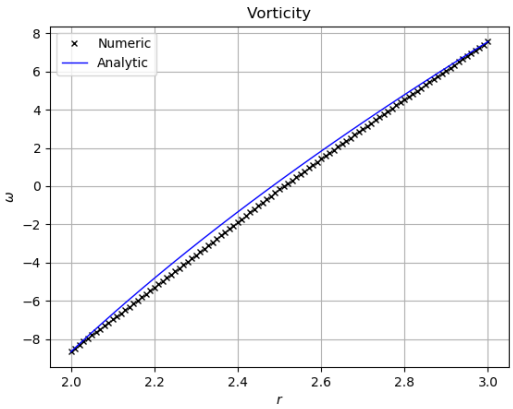
The parameters used in the simulation were arbitrarily defined and are found in Tab. 2, the complete domain mesh is shown in Fig. 9. The boundary conditions were: $T_{wall} = 10$; $T_{in} = 1$; $U = 2$; $\psi = 0$ on inner wall and $\psi = 3.34$ on outer wall. In Fig. 10 (a), (b) and (c), the numerical results for the Hagen-Poiseuille flow were compared to the analytical solution and there is a little difference in the velocity, stream-function and vorticity profiles but overall they show a similar behavior. The complete numerical solution of the temperature and its profile in the radial direction for different positions along the axial direction is presented in Fig. 11.



(a) Velocity Solution

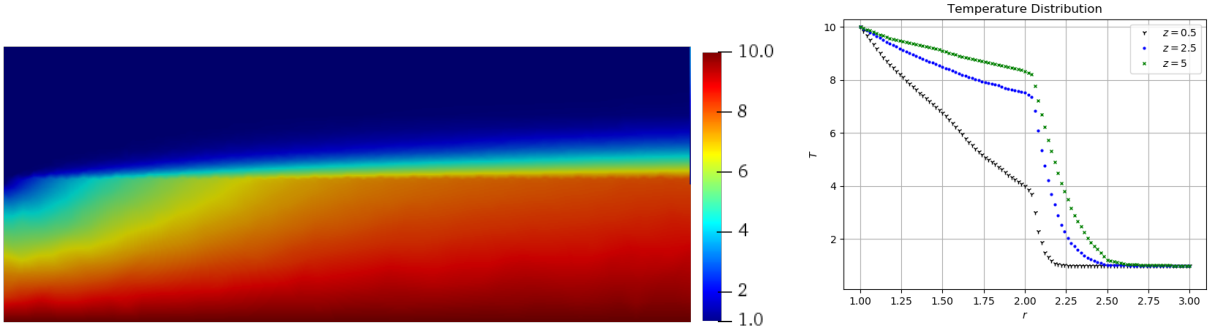


(b) Stream-function Solution



(c) Vorticity Solution

Figure 10: Numeric solution of the proposed conjugate heat problem. In (a), (b) and (c) it is shown a comparison between the simulation and analytical solution for the flow in an annular tube, the profiles were plotted from solution on $z = 5$.



(a) Numerical solution of the temperature on entire mesh

(b) Temperature in the radial direction

Figure 11: Numerical solution of the temperature distribution for a conjugate heat problem for a triangular mesh with 2328 nodes and 4779 elements. The lowest value, represented in blue is the boundary condition for the flow inlet temperature, and the highest value in red is the inner wall temperature of the cylinder. In (b) it is presented the temperature distribution on the radial direction on 3 different axial coordinates, $z = 0.5$, $z = 2.5$ and $z = 5$.

4. CONCLUSION

This work presents the development of a software capable of simulating conjugate heat transfer problems in axisymmetric applications. Both finite elements and semi-Lagrangian methods were validated with the Zalesak disk test and with the analytical solution for an axisymmetric heat conduction problem and for the Hagen-Poiseuille flow in an annular tube. The semi-Lagrangian improves stability of the numerical solutions and further tests need to be done in order to assess it's limits and the influence of its diffusivity in the current method.

Further steps in the development involve the generalization for higher order elements which further improves the numerical solution. Also, it is of interest to calculate the heat transfer coefficient between the solid and fluid phases and apply it to channels with different geometric configurations in order to simulate more complex heat exchanger devices, such as in the cooling of electronic components.

5. ACKNOWLEDGMENTS

The authors thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and FAPERJ (Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro) for their financial support.

6. REFERENCES

Abdellatif, N., Touihri, M. and El Amin, M., 2016. "Spectral element discretization for the stream-function and vorticity formulation of the axisymmetric stokes problem". *Calcolo*.

Ahrens, James, Geveci, Berk, Law and Charles, 2005. "Paraview: An end-user tool for large data visualization". URL `www.paraview.org`.

Anjos, G.R., 2012. *A 3D ALE Finite Element Method for Two-Phase Flows with Phase Change*. Ph.D. thesis, École Polytechnique Fédérale De Lausanne.

Batchelor, G.K., 2000. *An Introduction to Fluid Dynamics*. Cambridge University Press. ISBN 9780521663960,0521663962.

Carnevale, L.H., Anjos, G.R. and Mangiavacchi, N., 2018a. "Finite element analysis applied on the cooling of electronic components". *Congresso Nacional de Engenharia Mecânica*.

Carnevale, L.H., Anjos, G.R. and Mangiavacchi, N., 2018b. "Stream function-vorticity applied in the conjugated heat problem using the fem with unstructured mesh". *Brazilian Congress of Thermal Sciences and Engineering*.

Cesini, G., Paroncini, M., Cortella, G. and Manzan, M., 1998. "Natural convection from a horizontal cylinder in a rectangular cavity". *International Journal of Heat and Mass Transfer*.

Comini, G., Cortella, G. and Manzan, M., 1995. "A streamfunction-vorticity-based finite-element formulation for laminar-convection problems". *Numerical Heat Transfer, Part B: Fundamentals*.

Geuzaine, C. and Remacle, J.F., 2009. "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities". *International Journal for Numerical Methods in Engineering*.

Hahn, D. and Özisik, M., 2012. *Heat Conduction*. Wiley.

Hoffmann, K.A. and Chiang, S.T., 2000. *Computational Fluid Dynamics*, Vol. 1. Engineering Education Systems, 4th edition.

Jones, E., Oliphant, T., Peterson, P. *et al.*, 2001–. "SciPy: Open source scientific tools for Python". URL `http://www.scipy.org/`.

Panton, R., 1996. *Incompressible Flow*. Developmental clinical psychology and psychiatry. Wiley.

Peeters, M.F., Habashit, W.G.. and Dueck, E.G.., 1987. "Finite element stream function-vorticity solutions of the incompressible navier-stokes equations". *International Journal for Numerical Methods in Fluids*.

Salih, A., 2013. "Streamfunction-vorticity formulation". Indian Institute of Space Science and Technology, Department of Aerospace Engineering.

Szczukiewicz, S., 2012. *Thermal and Visual Operational Characteristics of Multi-Microchannel Evaporators using Refrigerants*. Ph.D. thesis, École Polytechnique Fédérale De Lausanne.

Vynnycky, M., Kimura, S., Kanev, K. and Pop, I., 1998. "Forced convection heat transfer from a flat plate: the conjugate problem". *International Journal of Heat and Mass Transfer*.