Politecnico di Torino

DIPARTIMENTO DI INGEGNERIA MECCANICA E AEROSPAZIALE - DIMEAS
Corso di Laurea in Ingegneria Aerospaziale

Tesi di laurea Magistrale

Numerical evaluation of the fluid dynamic forces acting over an Aortic aneurysm

Candidato
Marco Altomare
Matricola 226519

Relatore
Prof. Domenic D' Ambrosio

Correlatore
Ing. Florian Bernard

Dicembre 2018

Abstract

The aim of the project is to develop a software that can be used to solve the cardiovascular flow inside the Aorta, this can be very useful in the medical investigation on the aneurism after the diagnosis. Visualize how the aneurism affect the blood flow inside the Aorta permit to plan with extreme accuracy the actions that has to be taken. This work will present a method to improve the accuracy and make more efficient the numerical evaluation of the force distribution on the surface of an Aorta affected by an aneurism. Fluid Dynamic forces will be evaluate over a complex geometry using immersed boundary method and octree grids. The advantages in terms of time and memory used, using an octree grid instead of a uniform cartesian grid are first investigate. After that we present and test two different methods to evaluate the force over an immersed body, The first consist in integrating the forces over the surface of the body, in the second case a control volume approach is used. The result will be compared and the method to be used will be chosen to evaluate the force distribution over an aneurism of the aorta. Preliminary simulation are then performed and force distribution is evaluate using a simple non realistic boundary condition model.

Lo scopo del progetto é quello di sviluppare un software che possa essere utilizzato per risolvere il flusso cardiovascolare all'interno dell'aorta, ció puó essere molto utile nell'indagine medica sull'aneurisma dopo la diagnosi. Visualizzare come l'aneurisma influisce sul flusso di sangue all'interno dell'aorta permette di pianificare con estrema precisione le azioni che devono essere intraprese. In questo lavoro si presenterá un metodo per migliorare l'accuratezza e rendere piú efficiente la valutazione numerica della distribuzione delle forze sulla superficie di un'Aorta affetta da un aneurisma. Le forze fluidodinamiche saranno valutate su una geometria complessa utilizzando un metodo *immersed boundary* e una griglia di tipo octree. I vantaggi in termini di tempo e memoria utilizzati, utilizzando una griglia octree anziché una griglia cartesiana uniforme, verranno investigati inizialmente. Dopodiché sono stati analizzati due diversi metodi per valutare la forza su un corpo immerso, il primo consiste nell'integrare le forze sulla superficie del corpo, nel secondo caso viene utilizzato un approccio che si basa sulla variazione di quantitá di moto all'interno di un volume di controllo. I risultati saranno poi confrontati e si sceglierá quale metodo da utilizzare per valutare la distribuzione delle forze su un aneurisma dell'aorta. Quindi verranno eseguite delle simulazioni preliminari del calcolo delle forze sulla superficie dell'aorta e si verificherá la presenza di una concentrazione delle forze all'interno dell'aneurisma.

Contents

Introduction

Computational fluid dynamics is already been used to investigate the characteristics of aortic flow in a detailed way clarifying structures that are otherwise invisible to experimental measurements. To analyze these conditions, CAD models of the cardiovascular system are build using state of the art imaging techniques as the computed axial tomography (TAC). A 3D model is reconstructed from several TAC images and the flow can be computed over or inside the structure. Blood properties like Non-Newtonian behavior and realistic boundary conditions (e.g. systemic pressure or Windkessel model) have to be taken into consideration in order to have a realistic simulation of the problem. Therefore, it is possible to analyze and optimize the flow in the cardiovascular system for different applications [1]. This project has been developed within the MEMPHIS team of the research center INRIA, Bordeaux.The aim of the team is to develop a software that can be used to solve the cardiovascular flow inside the Aorta, this can be very useful in the medical investigation on the aneurism after the diagnosis. An aortic aneurysm is an enlargement (dilation) of the aorta up to 1.5 times normal size. They are most commonly located in the abdominal aorta, but can also be located in the thoracic aorta. If untreated, aneurysms tend to become progressively larger, although the rate of enlargement is individual and unpredictable. Aortic aneurysms cause weakness in the walls of the aorta and an aortic rupture can occur. Ruptures can result in massive internal bleeding and, unless treated immediately, shock and death can occur. Rarely, clotted blood which lines most aortic aneurysms can break off and result in an embolus.The risk of rupture of an Aortic abdominal aneurysm is related to its diameter and walls thickness; once the aneurysm reaches about 5 cm, the risk of rupture may exceed the risks of surgical repair for an average patient. Rupture risk is also related to shape; so-called "fusiform" (long) aneurysms are considered less rupture prone than "saccular" (shorter, bulbous) aneurysms[1], the latter having more wall tension in a particular location in the aneurysm wall. This force concentration is what we want to investigate in this work. Aortic aneurysms resulted in about 152,000 deaths worldwide in 2013, up from 100,000 in 1990 [14]. The main challenge is to make a simple, user friendly, software that can be used from surgeons or other staff members inside any medical facility. It would be then easy to visualize how the aneurism affect the blood flow inside the Aorta and plan with extreme accuracy the action that has to be taken. The software has been written by the Team and can solve the pressure and velocity fields inside the aneurysm. In this work is presented a method to evaluate fluid dynamic forces and how it is implemented in the code. Different action are taken in order to improve and make more efficient the numerical evaluation of the force dis-

tribution on the surface of an aneurism. In order to discretize and solve the Navier-Stokes equations a finite volume approach will be used. The velocity and pressure fields are solved using a fractional step method that consist in a prediction step in which the velocity is evaluated using a guess pressure field, and then a Correction step to correct the velocity and satisfy the continuity equation. The body is taken into account using the "penalization method" that consist in adding a term to the momentum equation to define the rigid body. The grid used to discretize the domain is an octree grid, a consequential subdivision of the cells only where needed. This can guarantee a more efficient use of the memory and a faster computation.To evaluate forces on the surface of the body, the original method was not very accurate, so the first step of this work is to find a more accurate method to evaluate the force. The Lagrangian Marker [9] and Noca method [11] will be introduced and tested on the simple test case of the flow past a 2D cylinder. The best choice will be then Validated through the use of two Benchmark test case, In [12] is presented a proposal to validate 2D CFD software from S.Turek, and then the code will be validated in 3D using the flow past a still sphere, presented by R.Campregher et al. in 2009 [13]. Preliminary simulation on the aorta has been then performed to verify the presence of a concentration of the forces inside the aneurysm.

Chapter 1

Navier-Stokes equations and problem definition

1.1 Introduction to Navier-Stokes

To describe the behavior of a flow and it's macroscopic and microscopic characteristics, using *computational fluid dynamics* techniques, it is necessary to numerically solve, fluid mechanics equation. Fluid statics or hydrostatics is the branch of fluid mechanics that studies fluids at rest, while fluid dynamic is the study of fluids in motion. Fluid mechanics has a wide range of applications in several engineering field with the aim to solve complex flow field around bodies or structures. Even field like meteorology or oceanography are regulated by fluid mechanics equation. Hydrostatics is fundamental to hydraulics, the engineering of equipment for storing, transporting and using fluids. It is also relevant to some aspect of medicine (in the context of blood pressure) and it will be the principal objective of this thesis work. We will treat, for a firs general overview on the problem, the fluid as continuum (a continuous distribution of mass in space).The continuum assumption is an idealization of continuum mechanics under which fluids can be treated as continuous, even though, on a microscopic scale, they are composed of molecules. In so doing, the atomic or molecular nature of the fluid is neglected and this implies that any small volume element (small in comparison to the characteristic length scale of the system) is always supposed to be sufficiently big to contains a huge number of molecules. Speaking about of an infinitesimal volume element we mean that it's very small compared with the volume domain but still large to contain vary many molecules. There are two common descriptions of continuum motion were both first presented by Leonhard Euler (1707-83).

1. **Lagrangian Method**: is a way of looking the fluid motion where the observer follows an fluid parcel during it's motion in space and time. In summary, using the Lagrangian method, we follow the fluid parcel to determine it's properties

2. **Eulerian Method**: focuses on specific locations in the space through which the fluid flows as time passes. It means to fix the coordinate and evaluate fluid characteristics in that point.

All the assumption inherent to a Newtonian fluid can be expressed in term of equation:

1. Conservation of mass
2. Conservation of energy
3. Conservation of momentum

It's necessary chose a control volume, which is an imaginary surface enclosing a volume of interest. Different control volume could be employed during the Navier Stokes writing, that choice affect the form of the governing equations but the physical concept is always the same. As graphically showed in 1.1 from [2] the first difference that can be observed is between a *Finite volume*, and an*Infinitesimal volume*. That difference conducts us, respectively, to an integral form and a differential form of the Navier-Stokes equation. Starting

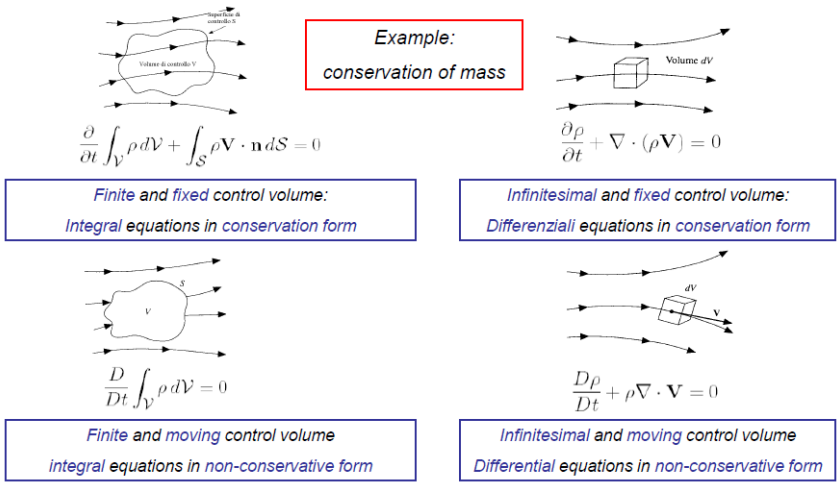


Figure 1.1: Different control volumes result in different approaches

from the example of the mass conservation law showed in 1.1, it's possible to write the first equation of interest.

1.1.1 Mass balance equation

The general form quoted for a mass balance is The mass that enters a system must, by conservation of mass, either leave the system or accumulate within the system . General hypothesis are shown below, referring to 1.2

- Defining a Cartesian frame of reference x, y, z, where density and velocity are function of space and time t.
- Considering an infinitesimal volume element placed in a generic point x, y, z, those dimensions are dx, dy, dz.
- Considering the mass flux across the six surfaces of the volume, and evaluating the net flux difference along x axis, so the flux across the left surface and the right one, we obtain the gradient of the mass along x direction.

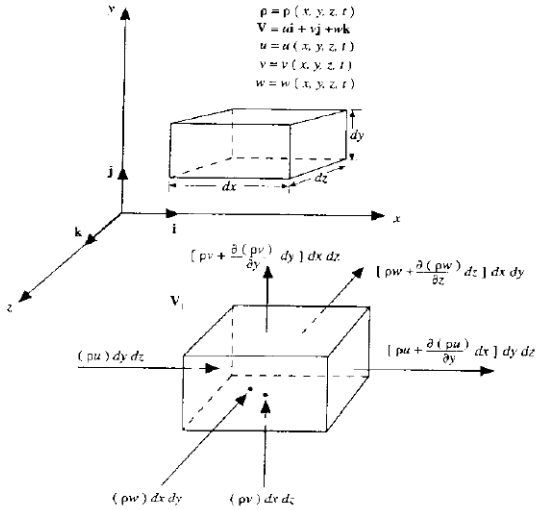


Figure 1.2: Volume forces acting on the control volume

$$\Big[(\rho u) + \frac{\partial(\rho u)}{\partial x}\Big]dydz - (\rho u)dydz = \frac{\partial(\rho u)}{\partial x}dxdydz \tag{1.1}$$

Using the same process it's possible to obtain the y and z directions the net flux coming out from the volume

$$\Big[(\rho v) + \frac{\partial(\rho v)}{\partial y}\Big]dydz - (\rho v)dydz = \frac{\partial(\rho v)}{\partial y}dxdydz \tag{1.2}$$

$$\Big[(\rho w) + \frac{\partial(\rho w)}{\partial z}\Big]dydz - (\rho w)dydz = \frac{\partial(\rho w)}{\partial z}dxdydz \tag{1.3}$$

Therefore, the global mass flux that pass through the volume chosen is

$$\Big[\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y}\Big]dxdydz \tag{1.4}$$

So the overall mass in the domain is

$$\rho dxdydz \tag{1.5}$$

So the time-rate of reduction of the mass fluid in the control volume is

$$\frac{\partial \rho}{\partial t}dxdydz \tag{1.6}$$

Obviously we can say that the *Time rate reduction of the mass* is equal to the *Net flux of mass coming out from the control volume*. Comparing the expression of the 1.4 with the 1.6, we obtain the mass balance equation written as

$$\Big[\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y}\Big]dxdydz = \frac{\partial \rho}{\partial t}dxdydz \tag{1.7}$$

Writing the vectorial form of the above equation we obtain the *conservative form* of the mass balance equation

$$\frac{\partial(\rho)}{\partial t} + \nabla\dot{(}\rho\underline{V}) = 0 \tag{1.8}$$

Using the $\nabla \cdot (\rho \underline{V}) = \underline{V} \cdot \nabla \rho + \rho \nabla \cdot \underline{V}$ and the material derivative applied to the tensor field as

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \underline{V} \cdot \nabla \rho \tag{1.9}$$

we finally have been obtained the *Lagrangian* or the *Non-Conservative* form. Summarizing we can write:

- **finite** control volumes:equation in **integral form**.
- **infinitesimal** control volumes:equation in **differential form**.
- **fixed** control volumes:equation in **conservative form**.
- **moving** control volumes:equation in **non-conservative form**.

If the equation has been written using an integral approach, it's always possible to switch to the differential form and vice versa. To reconstruct one form to each other we use the Gauss theorem in order to transform a surface integral into a volume integral of the flux through the surface *ds* that surrounds the volume *dv*.

Gauss Theorem

The Gauss, or divergence, theorem states that, if V is a connected three-dimensional region in R^3 whose boundary is a closed, piece-wise connected surface S and $\underline{F}$ is a vector field with continuous first derivatives in a domain containing V then

$$\int_V \nabla \cdot \underline{F} dV = \int_S \underline{F} \cdot \mathbf{n} dS \tag{1.10}$$

where S is oriented with the normal pointing outward . S can be disconnected, if V has one or more inner boundaries the normal points inwards on the inner boundary. In other words, the normal always points away from V.

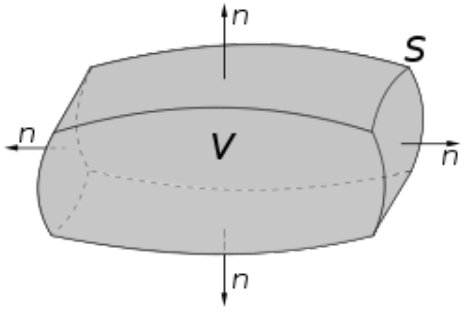


Figure 1.3: Volume example for Gauss theorem

1.1.2 Momentum balance equation

Using the second Newton's law on a infinitesimal control volume that contain a fixed mass δm and moves with the flow, the equation will be a vector equation.

The control volume always contains $\rho dxdydz$, and the acceleration can be written, dividing the three direction components as

$$\begin{aligned} a_x &= \frac{Du}{Dt} \\ a_y &= \frac{Dv}{Dt} \\ a_z &= \frac{Dw}{Dt} \end{aligned} \tag{1.11}$$

We can identify two kind of forces as the *Volumetric* and the *Surfaces* ones. The first one act directly on the mass of the control volume, while the second one can be described as the consequences of two different phenomena, referring only to the components along x-axis of the vectorial equation

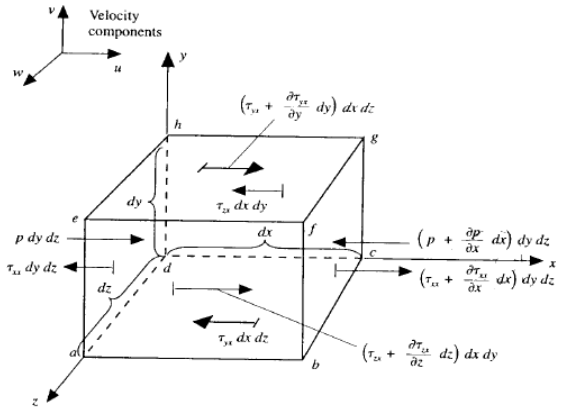


Figure 1.4: Forces and shear stress acting on the infinitesimal control volume

1. **Pressure** imposed from the external field on the surface that surrounds the cell volume. If defined as as a force per unit mass

$$\rho f_x dxdydz \tag{1.12}$$

2. **Shear and normal stress** imposed by the external field due friction on the surface.

   - Net pressure in the x direction acting on the $dydz$ surface, taking care of 1.4.

$$\left[p - \left(p + \frac{\partial p}{\partial x}dx\right)\right]dydz \tag{1.13}$$

   - Net friction force in the x-direction, according to

$$\begin{aligned} &\left[-\tau_{xx} + \left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x}dx\right)\right]dydx + \\ &\left[-\tau_{yx} + \left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y}dy\right)\right]dxdz + \\ &\left[-\tau_{zx} + \left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z}dz\right)\right]dydx \end{aligned} \tag{1.14}$$

By adding together the field and the surfaces forces, and writing on the left hand side the second Newton's law for each Cartesian axis direction we obtain the three momentum balance equation we were looking for. These equation are the three component of the one vectorial equation mentioned before. Using some mathematical manipulation as the material derivative, and introducing the 1.9 it's possible to remove a therm in the equation that represent exactly the 1.9, thus

$$\rho\frac{Du}{Dt} = \frac{\partial(\rho u)}{\partial t} + \nabla(\rho u\underline{V}) \tag{1.15}$$

that's the *non-conservative* form of the momentum balance equation. We can also obtain the differential conservative form.

$$\begin{aligned}
\frac{\partial(\rho u)}{\partial t} + \nabla(\rho u\underline{V}) &= -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \\
\frac{\partial(\rho v)}{\partial t} + \nabla(\rho v\underline{V}) &= -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \\
\frac{\partial(\rho w)}{\partial t} + \nabla(\rho w\underline{V}) &= -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z
\end{aligned} \tag{1.16}$$

1.1.3 Energy balance equation

For this subsection we should consider all the net heat flux through the surface containing volume showed in 1.5 and the volumetric heating. We have although

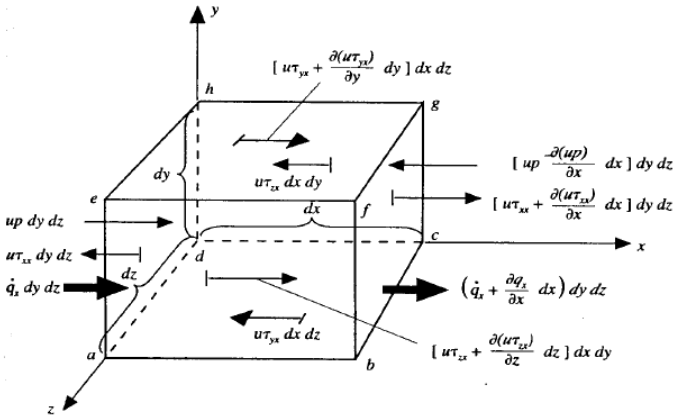


Figure 1.5: The net heat flux due to thermal conduction that flows along x axis

consider the work done on each surface of our control volume by the shear stress. Using the equation of the total energy in our infinitesimal volume and it's rate of change and the Fourier's law we define the non-conservative form of the energy balance equation. That equation can be manipulated using also the mass balance equation multiplied by total volume energy per unit mass to obtain the differential conservative form.

1.1.4 Newtonian and non-Newtonian fluids

A Newtonian fluid is defined as a fluid whose *shear stress* is proportional to the velocity gradient(measured in the direction perpendicular to the plane the

shear stress) of the flow. Water is a simple example of Newtonian fluid. A less rigorous definition of Newtonian fluid is that the drag of a generic body immersed in the flow is proportionally to the force applied to the body.
By contrast a Non-Newtonian fluid does not obey Newton's law viscosity. In general the viscosity for that kind of fluid depends on shear rate. Although fluid can even exhibit time-dependent viscosity. Therefore, it's impossible to define a constant viscosity term. A classic example of a non-Newtonian flows is the blood. It results important to specify that all the study about the blood through aorta vessel did in this thesis project have been considered as Newtonian flow in order to use the classical Navier-Stokes equation. It's necessary evaluate a viscous time dependent term to introduce blood characteristics on the test case that characterize the present work.

1.2 The incompressible Navier-Stokes equations

It's important to note that for incompressible flows, equation of state does not exist. In practice this means that the energy equation is decoupled from the other two equations. Therefore we can solve continuity and Navier-Stokes equations to find the unknown velocity and pressure distribution without knowing the temperature (we assume that fluid properties are taken to be constant, i.e. not functions of temperature. If fluid properties change with temperature all equations becomes coupled as in the case of compressible flows). Heat transfer and therefore the energy equation isn't always a primary objective in an incompressible flow. For isothermal incompressible flows energy equation can be dropped and only the mass and momentum equations are solved together to obtain the velocity and pressure fields in the whole domain. The main difficulty of solving these equation for an incompressible test case lies in the role of pressure. Pressure, under the incompressible hypothesis is no longer a thermodynamic quantity and it can not be related to density or temperature through an equation of state. It establishes it self with infinite velocity, so it's instantaneously, so that the velocity field always remains divergence free. In the continuity equation there is no pressure term and in the momentum equation there are only the derivatives of pressure, but not the pressure itself. This means that , in this case, under that hypothesis the value of the pressure is not important, only the changes of pressure in space are important. The equation system of the Navier-Stokes equation for an incompressible flow are shown below.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \underline{V}) = 0 \rightarrow \nabla \cdot \underline{V} = 0 \tag{1.17}$$

$$\frac{\partial (\rho \underline{V})}{\partial t} + \nabla p + \nabla \cdot (\rho \underline{V}\underline{V}) - \nabla \cdot \underline{\underline{\tau}} = 0 \rightarrow \rho \frac{\partial (\underline{V})}{\partial t} + \nabla p + \rho \nabla \cdot (\underline{V}\underline{V}) - \nabla \cdot \underline{\underline{\tau}} = 0 \tag{1.18}$$

where $\underline{\underline{\tau}}$ is the shear stress tensor defined as a $3by3$ matrix, and can be written in a compact form as.

$$\tau_{ij} = \delta_{ij} \lambda \nabla \cdot \underline{V} + \mu \Big[\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i} \Big] \qquad \text{with } \lambda = -\frac{2}{3}\mu \tag{1.19}$$

Thanks to the continuity equation ($\nabla \cdot \underline{V} = 0$) the components of the shear stress tensor are reduced to

$$\tau_{ij} = \mu\Big[\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\Big] \tag{1.20}$$

Expanding the vectorial notation in a two-dimension configuration is possible to observe, as done in [2], that we should now resolve a three equation system. The first one is the classical incompressible mass equation, the second and the third are the momentum equation in the two direction axis.Re- arranging the diffusive terms using the mass balance equation and using $\mu = const$ it is possible to obtain the following equation system.

$$\begin{aligned} &\nabla \cdot \underline{V} = 0 \\ &\rho\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} + \rho\nabla \cdot (u\underline{V}) - \mu\nabla \cdot (\nabla u) = 0 \\ &\rho\frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} + \rho\nabla \cdot (v\underline{V}) - \mu\nabla \cdot (\nabla v) = 0 \end{aligned} \tag{1.21}$$

Now, we need further equations that will allow us to solve the pressure field. First of all, let's derive the two equation from the momentum balance equation and summing them together applying the mass balance equation we arrive to the *Poisson equation* for the pressure field as

$$\nabla^2 p + \rho\frac{\partial(\underline{V} \cdot \nabla u)}{\partial x} + \rho\frac{\partial(\underline{V} \cdot \nabla v)}{\partial y} = 0 \tag{1.22}$$

Further manipulation leads us to the *Integral form* and it will be possible to transform the non time dependent term showed in the 1.22 in surface integrals using the Gauss theorem explained before.

1.3 Problem definition

Out task is to solve a 3D incompressible flow inside a cave body (the Aorta), but we start considering a 2D flow around a solid body first, we will subsequently extend the discussion to the three dimensional case. The whole domain is denoted as $\Omega = \Omega_f \cup \Omega_b$. The solid domain is named Ω_b and the boundary of the solid body is denoted as $\partial\Omega_b$. Let $\mathbf{u_b}$ be the velocity of each $x_b \in \Omega_b$. The problem is governed by the following equations:

$$\rho\big(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}\big) = -\nabla p + \mu\Delta\mathbf{u} \qquad \text{in } \Omega_f \tag{1.23}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega_f \tag{1.24}$$

$$\mathbf{u} = \mathbf{u_b} \qquad \text{on } \partial\Omega \tag{1.25}$$

With homogeneous initial condition in Ω and Boundary Conditions of Dirichelet for the pressure on $\partial\Omega$ and Neumann boundary condition for the velocity on the inlet side of $\partial\Omega$ and free flow condition on the outlet side of $\partial\Omega$.

1.4 Penalization method

Having a body immersed in the fluid leads to the use of a body fitted mesh, which complicate the problem, to avoid the insurgence of this issue the Penalization Method [3] can be applied to our problem.
According to that to take into account the presence of the solid body inside the fluid domain the whole system is considered to be a fluid flow that has density ρ. The solid body is considered to be a porous item inside the fluid with a very small permeability $K << 1$. We now define a characteristic function χ that will be equal to 1 inside the solid body and null elsewhere.

$$\chi(\mathbf{x}, t) = 1 \qquad \text{if } \mathbf{x} \in \Omega_b \tag{1.26}$$
$$\chi(\mathbf{x}, t) = 0 \qquad \text{if } \mathbf{x} \notin \Omega_b \tag{1.27}$$

In [3] is shown how to take into account the penalization model inside the Navier-Stokes equation on the entire domain Ω through adding the so called penalization term to the momentum equation in the limit of $K \to 0$.

$$\rho\big(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}\big) = -\nabla p + \mu \Delta \mathbf{u} + \frac{\rho}{K}\chi(\mathbf{u_b} - \mathbf{u}) \qquad \text{in } \Omega \tag{1.28}$$
$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{1.29}$$

So the velocity on the boundary of the immersed body through the use of the penalization term $\frac{1}{K}\chi(\mathbf{u_b} - \mathbf{u})$ and the equations governing the problem can be discretized on the cartesian grid, so the use of a body fitted mesh is no longer needed. But the points on the boundary of the body do not correspond to the points of the grid, to solve this issue and to verify the condition 1.25 in the first approach of the penalization method, the penalized velocity is forced on all the grid points inside the rigid body, that gives a first order of accuracy in time.

1.4.1 Second order penalization method

In [4] is shown how the penalized velocity can be corrected using the *Image Point Correction*. Using this method is possible to correct the penalized velocity in all the ghost points, solid points which have at least one fluid neighbor. this is needed to evaluate a more accurate value of the gradient in the zone near the immersed boundary.
The level set function ϕ is now introduced, is defined as the signed distance from the solid-fluid interface $\partial\Omega_b$ to a given point of the domain [5], is positive if outside the solid and negative if inside ($\in \Omega_b$) figura1.6. The solid-liquid interface is defined by $\phi = 0$. The level set function has to satisfies the condition

$$[h]\frac{\partial \phi}{\partial t} + (\mathbf{u} \cdot \nabla)\phi\big) = 0 \text{ in } \Omega \tag{1.30}$$

and the normal vector to the surface is

$$\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|} \tag{1.31}$$

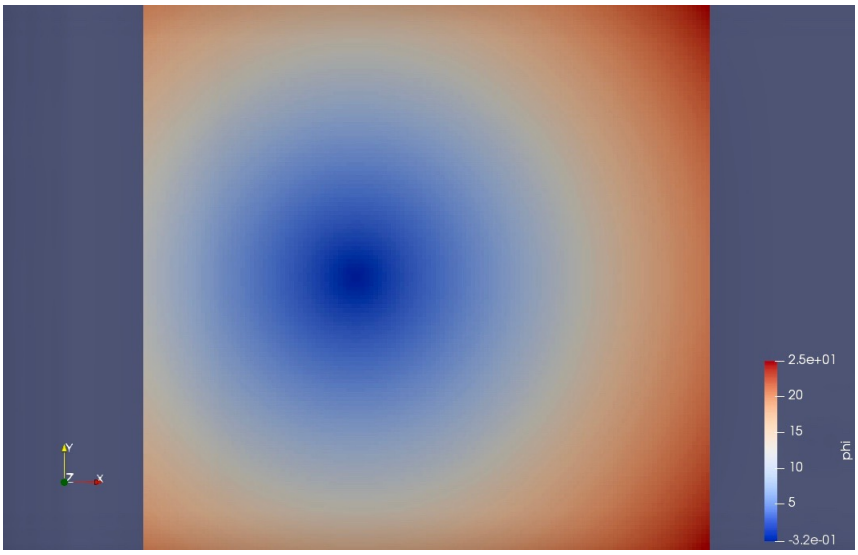


Figure 1.6: Level set function distribution for a solid cylinder immersed in the fluid domain.

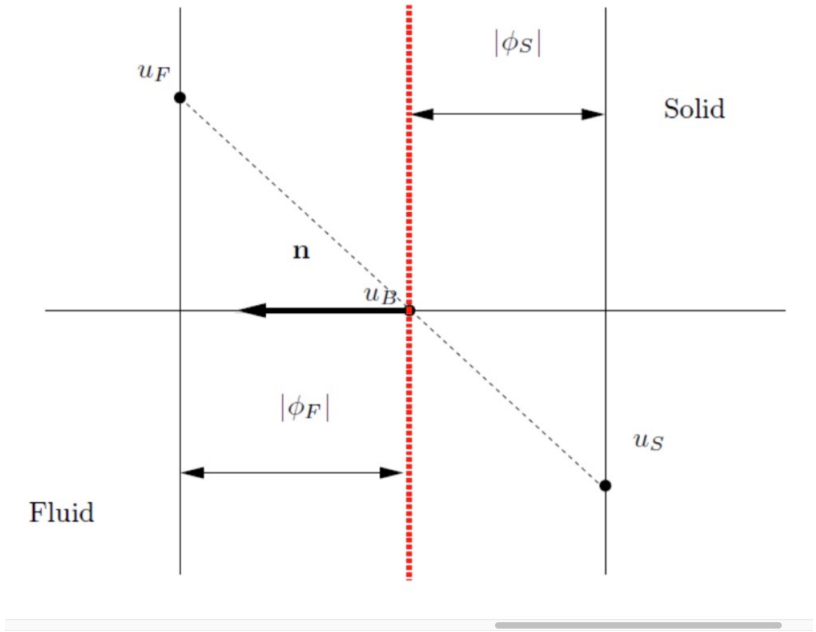


Figure 1.7: [4] one-dimensional scheme for second order penalization

To evaluate the correction of the velocity we use the fact that the velocity gradient through the interface in the normal direction with respect to the interface does not change. In relation to figure 1.7 we can express this statement as:

$$\frac{\mathbf{u_S} - \mathbf{u_B}}{|\phi_S|} = \frac{\mathbf{u_B} - \mathbf{u_F}}{|\phi_F|} \tag{1.32}$$

where$|\phi_S|$ and $|\phi_F|$ are the absolute value of the level set in the points S and F

The velocity on the ghost nodes $\mathbf{u_S}$ (figure 1.8) is evaluated knowing the velocity on the symmetric point with respect to the interface along the Normal to the interface itself $\mathbf{u_F}$, and imposing the desired velocity of the point on the interface $\mathbf{u_B}$.

$$\mathbf{u_S} = \mathbf{u_S} - \phi\left(\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right)\Big|_{\phi=0} \tag{1.33}$$

In the two-dimensional case (figure 1.8) we need to compute $\left(\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right)for\phi = 0$, we find all the normal points to the ghost points as the point on the interface B is located at a distance ϕ ant the symmetric point is at 2ϕ from the ghost point. The value of $\mathbf{u_F}$ can be interpolated using the velocity his fluid neighbors.

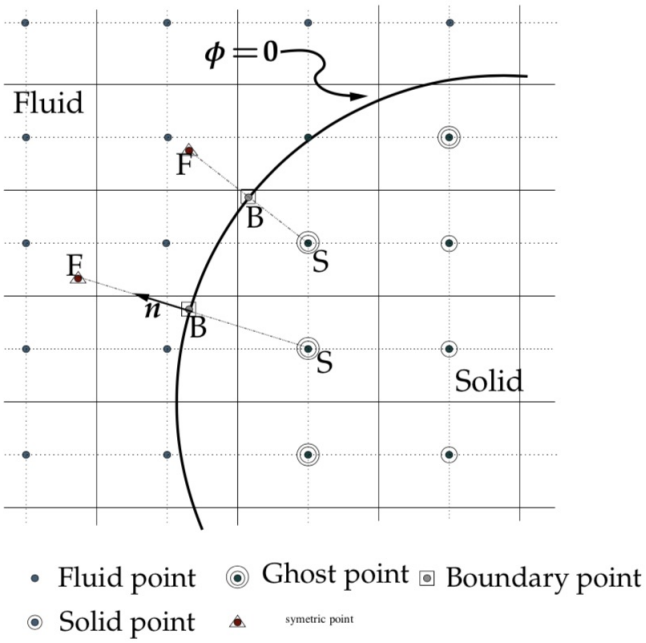


Figure 1.8: [4] sketch of the correction for the velocity in a two-dimensional case.

1.5 Prediction-Correction fractional step method

For the discretization in time of the problem a predictor-corrector fractional step method is used, This method consist in calculate a velocity $\mathbf{u}^*$ using an initial guess for the pressure field p^*. But in this way the condition $\nabla \cdot \mathbf{u}^{n+1} = 0$ is not verified so a correction on the velocity field has to be evaluated.

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{\nabla p}{\rho} + \nu \Delta \mathbf{u} + \frac{1}{K}\chi(\mathbf{u_b} - \mathbf{u}) \qquad \text{in } \Omega \tag{1.34}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{1.35}$$

Discretizing the derivative with an implicit scheme

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \nu \nabla^2 \mathbf{u}^n + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n - \frac{1}{\rho}\nabla p^{n+1} + \frac{1}{K}\chi(\mathbf{u_b^n} - \mathbf{u}^*) \tag{1.36}$$

adding and subtracting to the equation $\mathbf{u}^*$ and $\frac{1}{\rho}\nabla p^n$ we obtain:

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^*}{\Delta t}+\frac{\mathbf{u}^*-\mathbf{u}^n}{\Delta t}=-\frac{1}{\rho}\nabla p^{n+1}-\frac{1}{\rho}\nabla p^*+\frac{1}{\rho}\nabla p^*-(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n+\nu\nabla^2\mathbf{u}^n \quad (1.37)$$

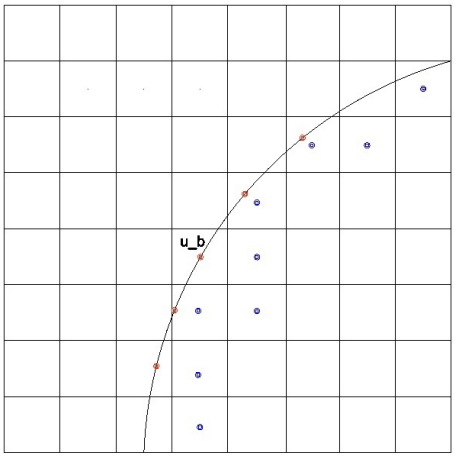
In this work the a-dimensional form of the equations is used.

1.5.1 Prediction step

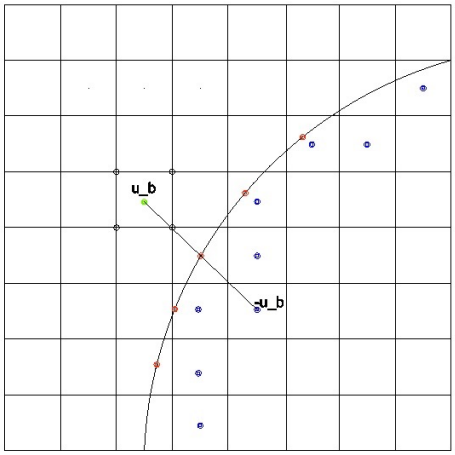
In the **prediction step** we solve the first fractional step given a guess pressure field and in order to obtain $\mathbf{u}^*$.

$$\frac{\mathbf{u}^*-\mathbf{u}^n}{\Delta t}=-(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n-\nabla p^*+\frac{1}{Re}\nabla^2\mathbf{u}^n+\frac{1}{K}\chi(\mathbf{u_b^n}-\mathbf{u}^*) \quad (1.38)$$

The Penalization term is taken into account in this step, so to solve it and evaluate $\mathbf{u}^*$ we need tho know $\mathbf{u_b}$ in the ghost cells.
If we use a first order prediction the velocity (figure 1.9a) into the ghost cell (in blue) is enforced to be the velocity of the body itself, if the body does not move $\mathbf{u_b}=0$.



(a) First order interpolation to evaluate $\mathbf{u_b}$



(b) Second order interpolation to evaluate $\mathbf{u_b}$

If we use a second order penalization the velocity enforced on the ghost cell is the opposite of the value on the symmetric point respect to the boundary of the body, this because if the body does not move the velocity on the red point is null. Referring to figure 1.9b, the value of $\mathbf{u_b}$ in interpolated using only fluid points (black circles).

$$\mathbf{u_b}=\sum_i \omega_i\mathbf{u}_i^*=F(\mathbf{u}_i^*)$$

where ω are the interpolation weights and $F(\mathbf{u}_i^*)$ represent the interpolation function.
If the body is in motion the velocity of the red points is $\mathbf{u_c}\neq 0$ and the penalized velocity on the ghost cell is $\mathbf{u_b}=[2\mathbf{u_c}-F(\mathbf{u}_i^*)]$

1.5.2 Projection and Correction steps

Since this velocity does not satisfy the continuity equation, we need to correct it, in the **Projection Step** we solve the second fractional step. using the velocity field $\mathbf{u}^*$ evaluated in the prediction step we can evaluate the correction on the pressure.

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -(\nabla p^{n+1} - \nabla p^*) \tag{1.39}$$

What we want to be verified is $\nabla \cdot \mathbf{u}^{n+1} = 0$ we take the divergence of the equation above and we get the Poisson equation

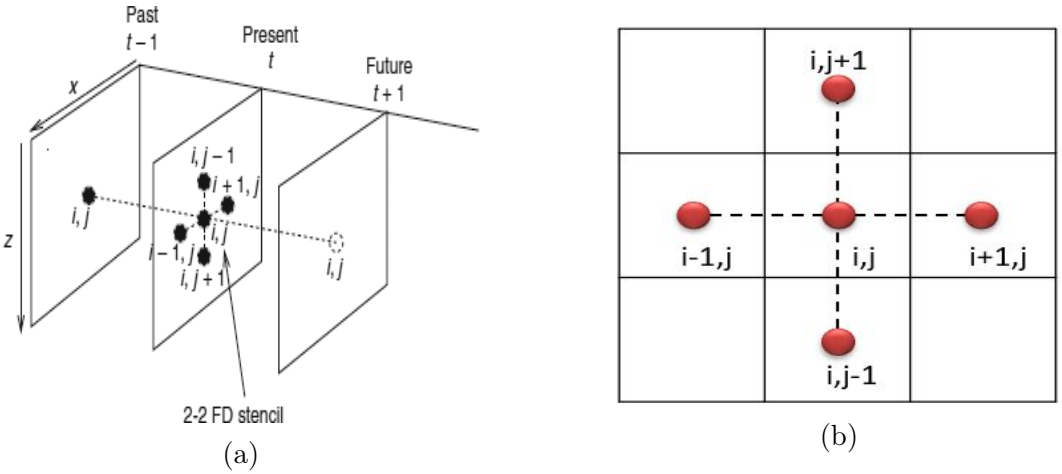
$$\frac{\nabla \mathbf{u}^*}{\Delta t} = -\nabla^2 \Phi \tag{1.40}$$

where $\Phi = p^{n+1} - p^*$
Solving with a Poisson solver we get the **correction step** of the field:

$$p^{n+1} = p^* + \Phi \tag{1.41}$$
$$\mathbf{u}^{n+1} = \mathbf{u}^* + \Delta t \nabla \Phi \tag{1.42}$$

1.6 Space Discretization

For the discretization in space of our problem a first order upwind scheme is used for the advection equation. The original idea was to use a *Semi-Lagrangian Scheme*, but there were several problem using it in three dimensional cases, so it is been decided to use a more conventional first order *Upwind scheme*. The scheme will be soon improved to reach the second order of accuracy. But for the first test the first order is enough



(a) (b)

Figure 1.10: The schematization of the method used to evaluate the second Derivatives.

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = \Big[\frac{u_j - u_{j-1}}{\Delta x} + \frac{v_j - v_{j-1}}{\Delta y}\Big]\mathbf{u}_j \tag{1.43}$$

For the Diffusive part of the equation a second order finite difference scheme is used, that leads to:

$$\nu\nabla^2\mathbf{u} = \nu\Big(\frac{u_{x+\Delta x,y} - 2u_{x,y} + u_{x-\Delta x,y}}{\Delta x^2} + \frac{u_{x,y+\Delta y} - 2u_{x,y} + u_{x,y-\Delta y}}{\Delta y^2}\Big) \quad (1.44)$$

$$\nabla p = \frac{p_{j+1} - p_{j-1}}{2\Delta x_i} \quad (1.45)$$

The Navier-Stokes solver was already implemented inside the code. First a Lagrangian scheme was adopted to discretize the convective part of the equation in space, but due to the problems this kind of discretization have in three-dimension, a more simple upwind scheme has been used for more complex or 3D problems. We will focus on the method used to evaluate the forces over a surface using the pressure and velocity fields evaluated using the fractional step method.

Chapter 2

Numerical Grid

In classical approaches for many problems like incompressible multi-phase fluid flow or fluid-structures interaction, the interfaces are considered to be internal boundaries using interface fitted meshes. These methods leads to the use of simple discretization and gives accurate results, but has very long computational times and generating and handling these grid can be hard for non-stationary problems. In general case, a uniform mesh results efficient for problem without regions where a greater accuracy is required. Such regions could be place where discontinuity occurs, or even shocks. The original idea was to implement a uniform cartesian grid with a finer refinement to catch this particular region, but this kind of approach leads to an always increasing computational cost. On the other hand, many problems in numerical analysis do not require an uniform precision in the entire discretized domain, but eventually only in some areas. In order to achieve high precision numerical simulation avoiding excessive computational cost, adaptive mesh refinement (AMR) method is used. It consist of an adaptive time dependent mesh during the calculation or fixed statically at its beginnings. It provides a focus on the precision of numerical computation based on those areas while leaving the other region of the domain at lower levels of precision and resolution. This kind of reproach allows the user to solve problems that are intractable on an uniform grid.The approach we use is presented in [6] and uses a mesh with hierarchical nature that makes the grid easy to be generated and partitioned and can be easily evolutive. This being more efficient with a low need of memory that makes it faster.

2.1 Discretization of the domain

For the space discretization of the whole fluid domain we use a hierarchical data structure that consist in a consequential subdivision of the cell. At each level a cell (parent) is subdivided into 4 equal cells called children for the two dimensional case (quadtree) or 8 children for a three-dimensional problem (octree). As the quadtree is defined in a square is easier to describe the structure of the grid in two dimensions (figure 2.1).
Referring to figure 2.2 the root cell is the base of the tree and is the whole domain before the discretization. The leafs are the cells that have no child (red cells in the figure). The tree is composed both by leaf and no leafs cells,

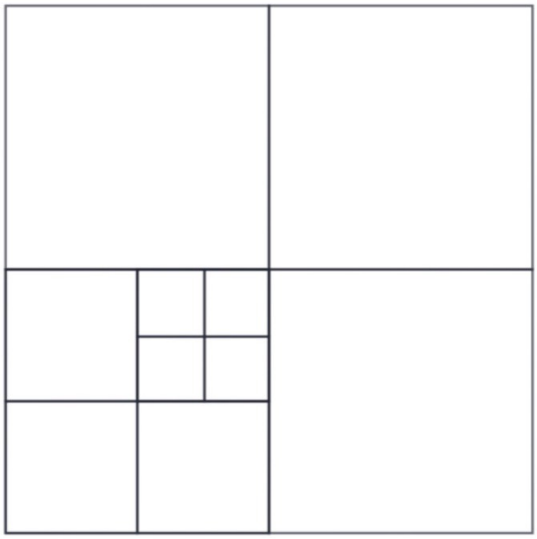


Figure 2.1: A quadtree subdivision of the grid

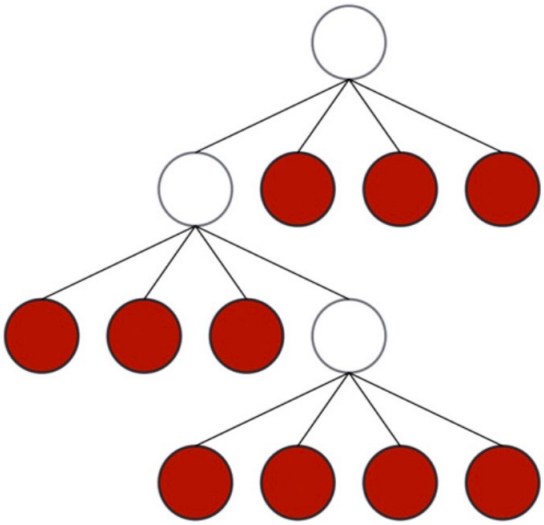


Figure 2.2: The decomposition of the grid represented as a quadtree

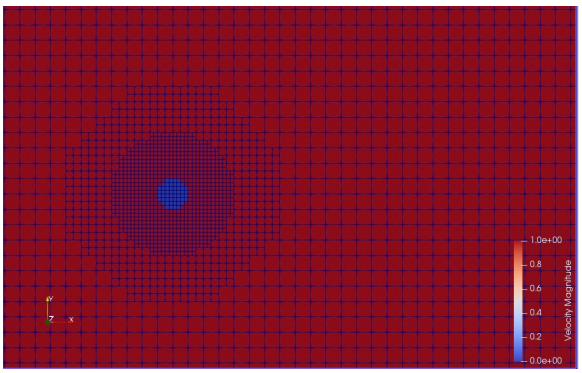


Figure 2.3: A solid cylinder in a fluid domain discretized with a cartesian hierarchical quadtree mesh

only leafs are stored in memory. The level of the cell is defined giving to the root cell the level zero and adding one each time there is a subdivision into 4 children, i.e. every time a new group of children is placed below the tree. Neighbors to the cells can be defined through the surfaces or through the corners.

If each cell has all the neighbors that has at most 1 level difference with the cell itself, then the grid is called **Graded** or balanced by definition. In this work only graded grids are used, but un-graded grid can be used as well, a progressive increment of two levels at the time it is also fine to be used.
The generation and handling of this kind of grid is guaranteed by the use of the library PABLO [7].

The main advantages of using this kind of grid are:

1. efficient access to the data stored in memory.
2. easy access to the cell number level and position.
3. easy neighbors identification.

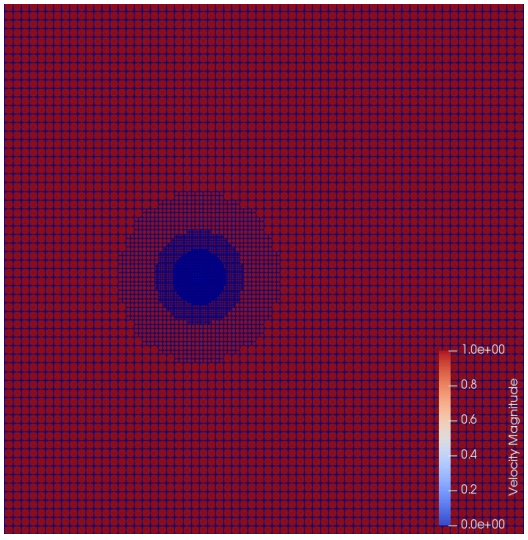
2.1.1 Refinement criteria of the cell

The final aim of the code is to have an automatic adaptive refinement of the grid. The code can create a dynamic mesh refinement, based on the computed value of the velocity gradient in each cell. In each cell the gradient is computed and compared to the limit value imposed in the code. If the value in the cell is greater than the limit value, the cell is refined Obviously, the dynamic refinement results in large computational costs each time the grid is refined.
For the purpose of this work this algorithm that evaluates the adaptive grid was ignored. The grid used is static, does not change in time and the cells to be refined are chosen by the user.
For the test case of the flow past a rigid cylinder the grids used are several:

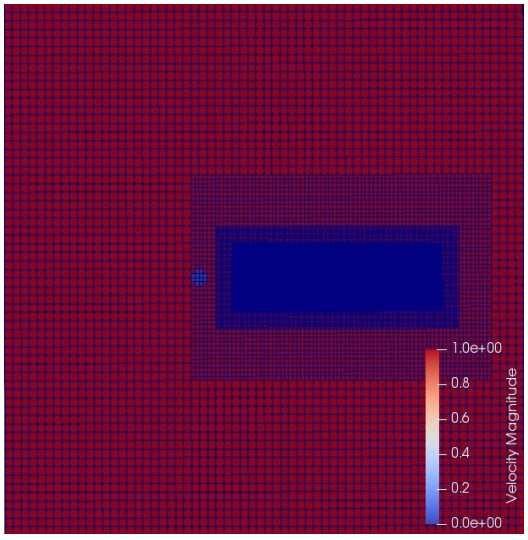
- Circular refinement: this approach leads a low computational costs since only the areas around the cylinder is refined, on the other hand the eddies that occurs in the wake are not represented in the best way since there is no refinement provided.
- Rectangular refinement: This approach avoid the low resolution problem that occurs in the wake for the previous refinement. There will be areas where the refinement is not required.
- Combined refinement: such approach is a good balanced mix between the two previous method. A circular refinement on the cylinder wall is imposed as well as a rectangular refinement that provides the high resolution required on the wake to catch all the vortex structures.

If a very high precision on the force evaluation is needed a circular refined grid is use as is more accurate on the surface of the body. If a good accuracy on

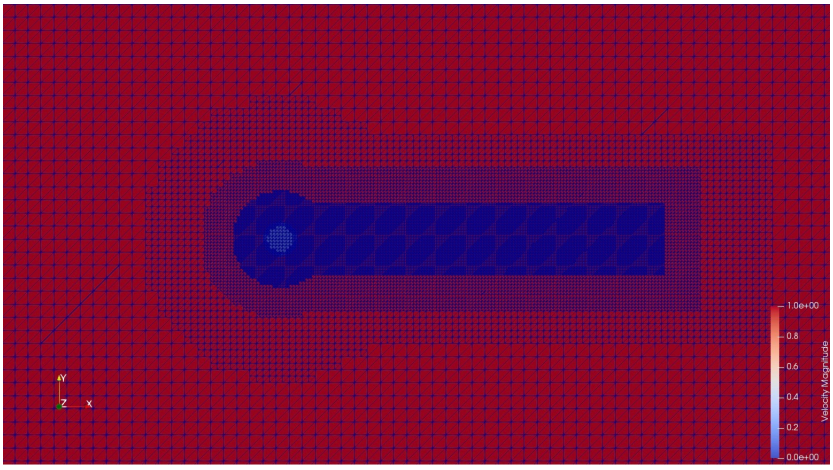
the vortex structures on the wake is needed, a rectangular refined grid is used. The combined grid presents both the advantages but the computational cost is larger.



(a) Circular Refinement around the surface of the cylinder



(b) Reptangular refinement of the wake of the cilinder



(c) A combination of both the cyrcular and the reptangular refinement

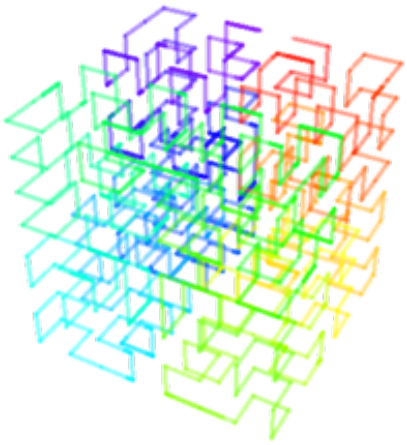
2.2 Z ordering

To arrange multi dimensional data into a mono-dimensional array several space filling curve can be used to number the cells. There are two kind of curves mostly used in multidimensional problems: the Hilbert Curve and the Z-Ordering Curve [10].

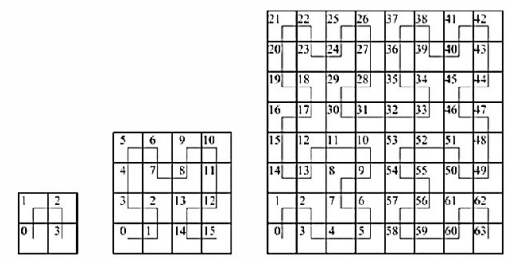
Hilbert curve

This curves was introduced by Hilbert in 19th century. Using this pattern each cell of a two-dimensional space can be represented with an index, with order

shown in figure 2.5a . So it makes possible to map 2D spaces into 1D. To find Hilbert order many algorithms can be used that generates, and handle it.



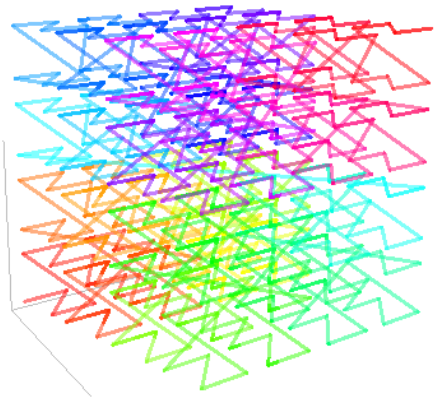
(a) A representation of the Hilbert curves in three dimensions



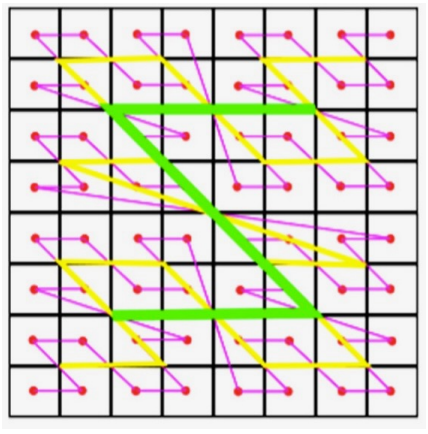
(b) A representation of the Hilbert curve ordering at different levels of refinement

Z-Order

The Z-ordering was first introduced by Morten in 1966 [10]. It can be used to generate an ordered numeration of the cells while using an quadtree grid. The basic idea is to sort the cell index following a space filling line with an inverse Z shape. At each subdivision of a cell into four children to each child an index is assigned from 0 to 3. 0 will be the bottom-left cell, 1 to the bottom-right cell, 2 to the top-left cell and 3 to the top-right one, as its possible to see in figure 2.6b.



(a) A representation of the Z-ordering in 3D at different levels of refinement



(b) A representation of the Z-ordering in 2D at different levels of refinement

The approach we use, once the cells have been sorted, is to store the indices in a binary search tree and used directly, which is called a linear quadtree,[3] or they can be used to build a pointer based quadtree.

Table 2.1: Geometric parameters

Dimension domain	16x16
Diameter	1
x center	-4
y center	0

2.3 Quadtree and Cartesian grid comparison

To appreciate the improvement given by using a quadtree grid we compared the computational cost performing simulations for the same test case with a cartesian grid versus a quadtree grid. The test case we use is the flow past a circular cylinder at Reynolds 100 (Figure 2.7) and CFL condition set to 0.8.s The cylinder diameter is 1 and his center is positioned in [-4, 0]. The Drag coefficient will be evaluated and compared with the results found by B.N Rajany et al. [9].

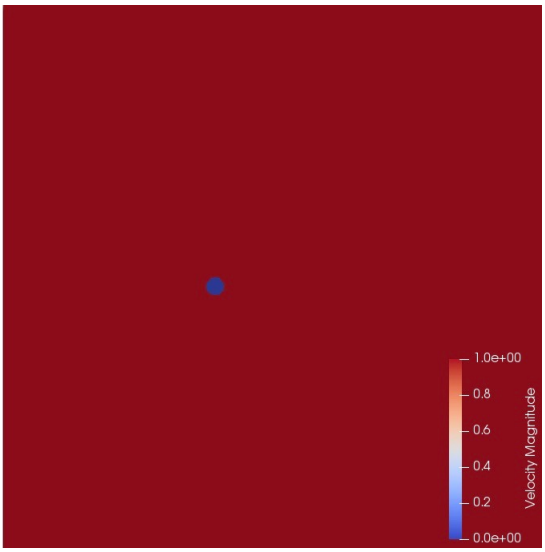


Figure 2.7: The circular cylinder in the fluid domain at t=0

To solve Navier-Stokes equation boundary and initial condition has been imposed. Table (2.3) summarize the overall geometrical condition. In particular, a free flow condition has been fixed for the outflow area while the inflow area needs Dirichelet boundary condition on the velocity field in order to have flow motion in the square domain. Upper and lower domain walls have been provided of Dirichelet BC on the velocity field, and a *Neumann* condition ($\nabla p = 0$) on the Pressure field. This kind of approach simulates a flow paste a 2D cylinder in *free field* condition.
For this test we suppressed the automatic generation of the grid. We imposed a rectangular refinement of the grid to include the wake of the cylinder in order to get a more accurate evaluation of the force (How the force are evaluated on the cylinder surface will be the next chapter's task). The simulation has been first performed on a cartesian grid and then we compared the values of the *Drag Coefficient* obtained with the results acquired using a quadtree domain's

Table 2.2: Initial condition imposed to the flow at the first time step.

Flow initial condition	
ρ	1
U	1
T	1
p	1
Re	100

Table 2.3: Boundary condition used for this analysis

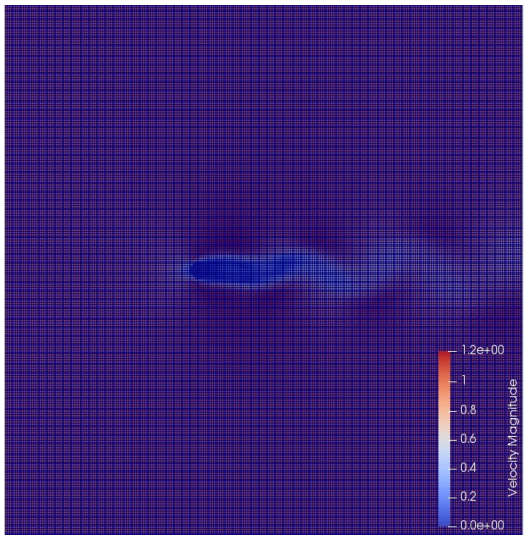
Boundary Condition	
Inflow	U=1 $\nabla p = 0$
Ouflow	Freeflow
upper&lower boundaries	U=1 $\nabla p = 0$

discretization.
The uniform cartesian mesh has cells dimensions which correspond to the one we would have using a Level 10 quadtree subdivision of the domain . That means, using a square domain, dividing ten times the domain following a quadtree pattern. Fixed the most refined level to 10 we will use 1, 2 and 3 levels of refinement to get to level 10 in the wake of the cylinder, starting respectively from level 9 8 and 7 as less refined levels. All the simulation has been performed in parallel using the same number of processors (196 processors on 24 Nodes). The value of the drag coefficient is taken after 500 seconds of physical time, after which the transient has ended and the flow is fully developed. According with the above explained the grids used will be:

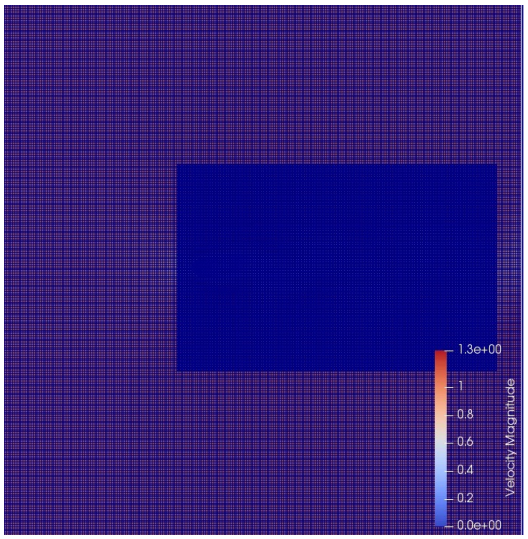
1. Level 10 with 0 refinements, figure 2.8a
2. Level 9 with 1 refinements, figure 2.8b
3. Level 8 with 2 refinements, figure 2.8c
4. Level 7 with 3 refinements, figure 2.8d

Table 2.4: The results obtained for different grids in terms of Drag coefficient, the time reduction and the cell reduction is evaluated with respect to the cpu time and number of cell of the finest uniform grid L 10-0.

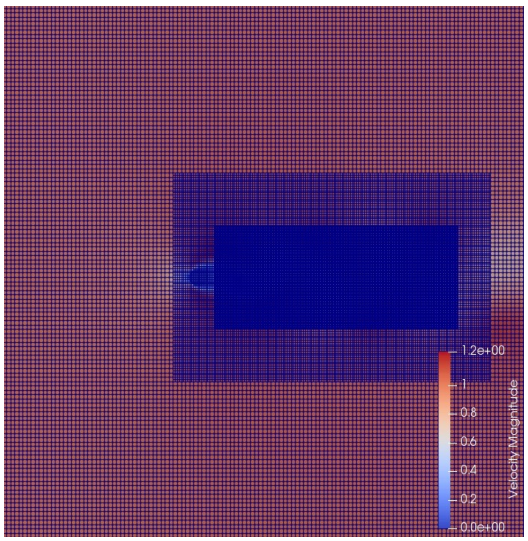
Grid	Cd_{av}	$\|\|\varepsilon\|\|$	Tcpu [h]	reduction	#cell	#cell red
L10-0	$1,2534$	$6,13\%$	$12:42:13$	$0,00\%$	1048576	$0,00\%$
L9-1	$1,2483$	$6,52\%$	$05:08:42$	$59,50\%$	441856	$57,86\%$
L8-2	$1,2756$	$4,47\%$	$02:40:09$	$78,99\%$	179584	$82,87\%$
L7-3	$1,2620$	$5,49\%$	$01:41:47$	$86,65\%$	84832	$91,91\%$



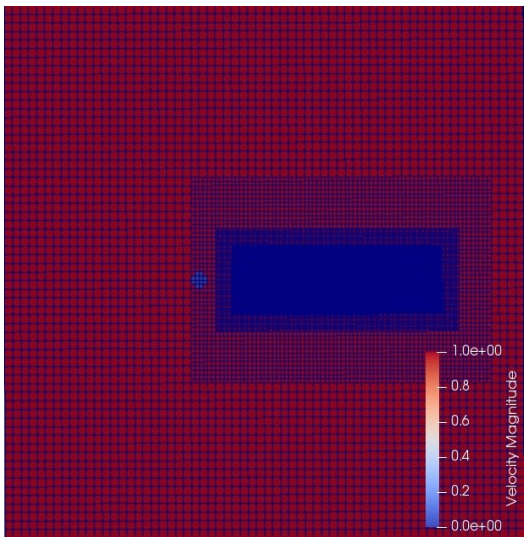
(a) Cartesian mesh, Level 10 with no refinement



(b) Initial Level 9 with one refinement on the wake

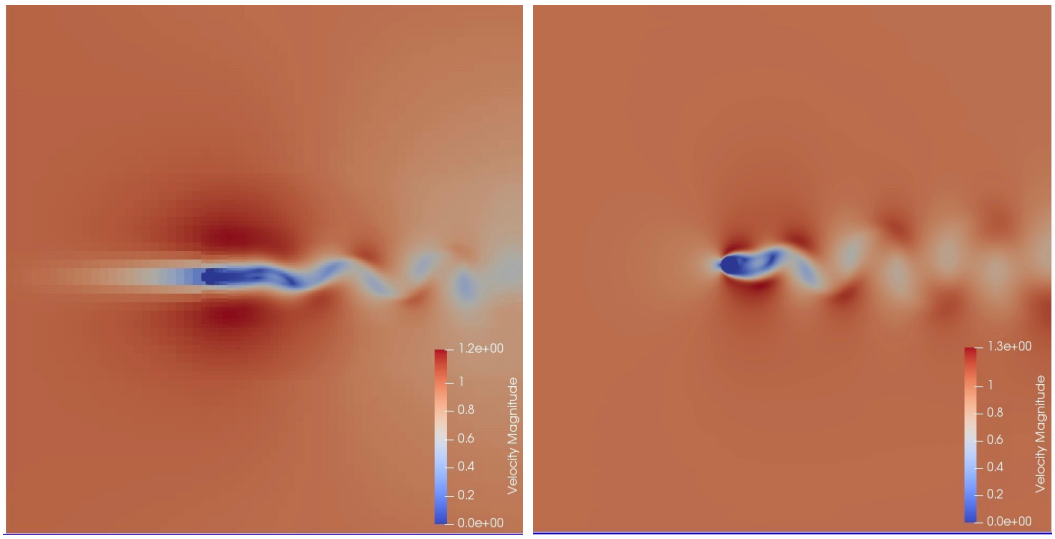


(c) Initial Level 8 with two refinement on the wake



(d) Initial Level 7 with three refinement on the wake

Figure 2.8: he grids used to discretize the domain



(a) (b)

Figure 2.9: In figure 2.9a is presented the velocity field after 500s of physical time using the grid level 7 with three refinements. In figure 2.9b the velocity field after 500s of physical time using the cartesian grid

The result are presented in table 2.4. the computational time is significantly reduced when using the octree mesh as well as the number of cells, even using only one level of refinement on the wake of the cylinder we have a reduction of the 59% on the cpu time and 58% in number of cell used to discretize the domain. The error does not change significantly respect to the one evaluated on the cartesian grid.

To have a significant reduction of The computational time without increasing the error too much, we take into consideration all the result with a percentage error less or equal to the 5% The first graph represented in figure 2.11 the cpu time needed for the simulation , summarized in table 2.4, compared with the cell's number of the discretized domain. It's possible to appreciate the *quasi-linear* dependence between the computational cost and the cell's number.
In order to have a graphical representation that highlights an optimized simulation configuration, in figure 2.10 fo each simulation performed is reported the percentage relative error with respect to the values obtained in [9], related to the corresponding percentage cpu time reduction with respect to the cartesian grid. This result are shown in table 2.4. The best grid set up seems to be the red circled point in figure 2.10. Such point corresponds to the initial grid level 8 with 2 refinement, and represent the minimum relative error4.47% obtained with a significant time reduction, the 80% less time, respect to the simulation performed using a cartesian uniform grid.

Figure 2.10: The percentage cpu time reduction with respect to the cartesian grid, versus the relative error with respect to the value presented in [9]. From this graph we can enlighten the best compromise grid that permit to reach a very high reduction in cpu time without increase the relative error.

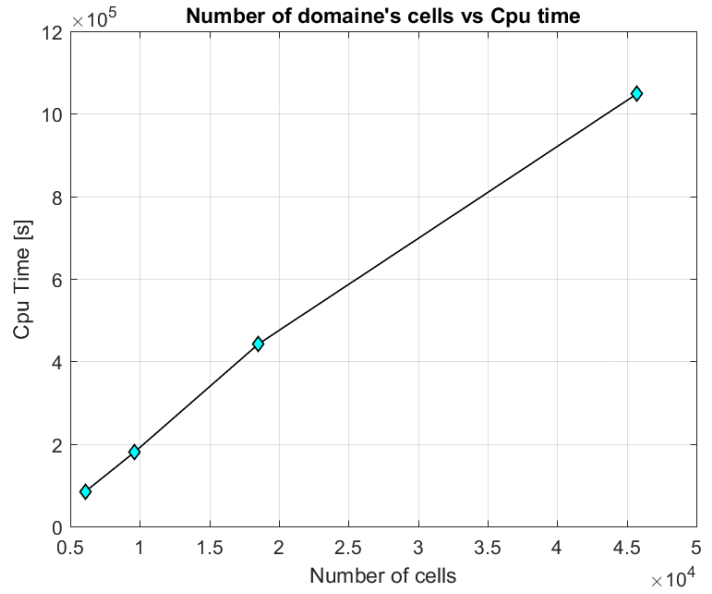


Figure 2.11: In this graph is shows how the computational time increase linearly increasing the number of cells used.

in order to have a larger view of the error and number of cells behavior further simulation has been accomplished. Maximum level discretization has been fixed as level 9 (each octant has been sub-divided 9 times), and the related simulation were:

- Initial level 9 with 0 refinement
- Initial level 8 with 1 refinement

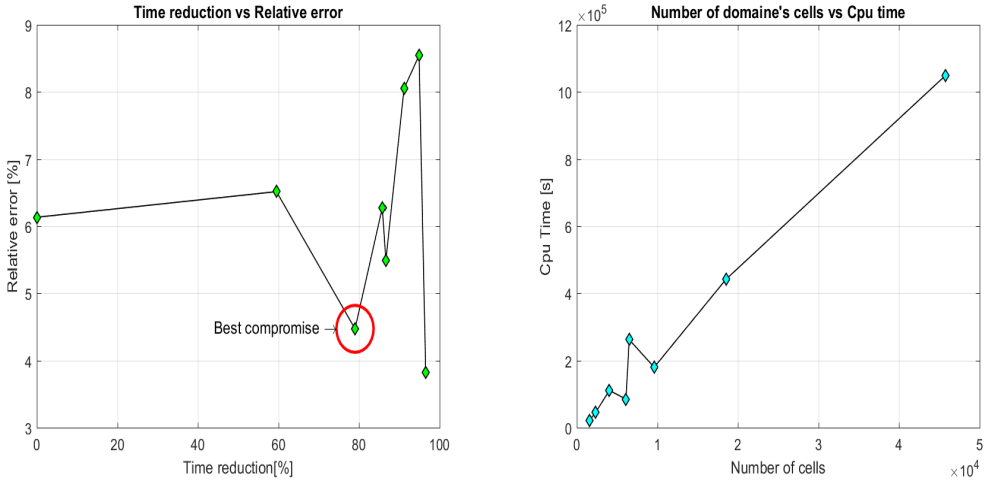
- Initial level 7 with 2 refinement
- Initial level 6 with 3 refinement

Calculating the mean drag coefficient obtained after transient phenomena, we can compare these results with the ones we obtained using as maximum discretization level 10.

Table 2.5: We repeat the analysis using several grids to have a better overview.

Grid	Cd_{av}	$(err\%)$	Tcpu [h]	reduction	#cell	#cell reduction
L10-0	$1,2534$	$6,13\%$	$12:42:13$	$0,00\%$	1048576	$0,00\%$
L9-1	$1,2483$	$6,52\%$	$05:08:42$	$59,50\%$	441856	$57,86\%$
L8-2	$1,2756$	$4,47\%$	$02:40:09$	$78,99\%$	179584	$82,87\%$
L9-0	$1,2515$	$6,28\%$	$01:48:30$	$85,77\%$	262144	$75,00\%$
L7-3	$1,2620$	$5,49\%$	$01:41:47$	$86,65\%$	84832	$91,91\%$
L8-1	$1,2278$	$8,05\%$	$01:07:26$	$91,15\%$	110464	$89,47\%$
L7-2	$1,2212$	$8,54\%$	$00:38:59$	$94,89\%$	44896	$95,72\%$
L6-3	$1,2843$	$3,82\%$	$00:26:27$	$96,53\%$	21208	$97,98\%$

Even if the grid at level 6 with 3 level of refinement gives the lowest error, we don't take into account this result as the grid is not enough refined and it is impossible to catch all the vortex structures inside the wake of the cylinder.



(a) (b)

Figure 2.12: From the graphs above is it possible to chose the discretization that can guarantee the best compromise between a fast computation and a small error.

Chapter 3

Evaluation of the forces

In this chapter will be shown the mathematical and numerical approach to describe forces distribution over a generic body immersed in a fluid domain. Since the cylinder is the most common case study for tests that includes force evaluation, and since the literature provides experimental and others numerical data, in the following paragraph a two-dimensional cylinder will be simulated using the Nurea code. Two mathematical approach are presented in this chapter. Integrating the force over the surface of the cylinder, this approach is based on the the principle for which dynamic force around a solid body immersed in a fluid flow can be calculated integrating the shear stress on the surface of the body. The second method presented by Noca [11], is based on the conservation of momentum. The force acting on the body is equal to the difference between the inflow and outflow momentum inside an arbitrary chosen control volume. The two methods has been both developed, implemented in the code and tested in order to chose the most suitable to use to compute the force distribution over the aneurysm.

3.1 Force evaluated on the surface

Starting from the momentum balance equation, and neglecting the gravity contribute we obtain a general formulas for continuous field. The total force acting on the surface of the cylinder can be evaluated from the equation that describes the total force acting on the surface of a body immersed in a fluid flow.

$$\mathbf{F} = \int_S p\mathbf{n}\ dS + \int_S \nabla\mathbf{u}\cdot\mathbf{n}\ dS \tag{3.1}$$

This equation in order to be solved numerically has to be discretized, considering the integrals as a sum of the values in all the cell of the surface.

$$\sum_i \mathbf{F}_i = \sum_i p_i \Delta S_i \mathbf{n} + \sum_i \nabla\mathbf{u}_i \cdot \mathbf{n}\Delta S_i \tag{3.2}$$

As the surface of the cell is the same we can rearrange the equation:

$$\sum_i \mathbf{F}_i = \Big(\sum_i p_i + \sum_i \nabla\mathbf{u}_i\Big) \cdot \sum_i \Delta S_i \cdot \mathbf{n} \tag{3.3}$$

3.1.1 Lagrangian markers method

In order to evaluate the fluid dynamic forces acting on the surface of the cylinder we will integrate the forces acting on each point of the surface. According to 5.1 to compute the force on a single cell we need the knowledge of pressure, velocity gradient, normal vector to the cell and surface of the cell. The points used to define the geometry of the cylinder (in red in fig 3.1) are called Lagrangian markers, and are not points of the grid so we don't know neither the pressure nor the velocity gradient. Furthermore in these points we cant evaluate the normal and obtaining the surface between two of them is difficult. To over come this problems we use a Lagrangian marker approach. The forces will be computed on the middle point of the segment connecting two consequential Lagrangian markers. On this control point we can easily compute the normal vector. The surface of the cell is equal to the length of the segment in the two-dimensional case. However Pressure and Gradient on the control points are interpolated using the values in the neighbor cells. Referring to figure 3.1 the red points are the Langrangian markers while the blue points are the control points. in the following sections of this work the control points will be represented as points on the surface of the body. Even if this points are not exactly on the boundary it is easier to illustrate how they will be used.

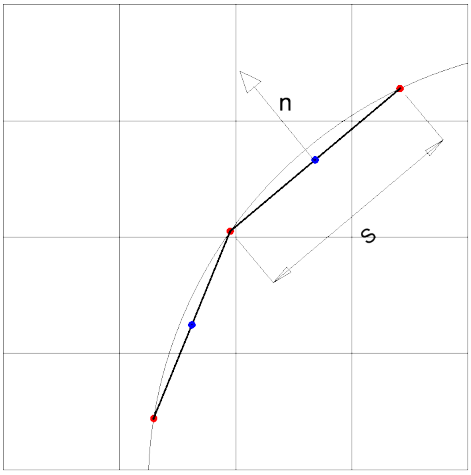


Figure 3.1: The points used to define the geometry of the solid body are used as lagrangian marker. The forces are computed on the middle point of the segment connecting two consecutive marker, where it is easy to calculate normal vector and surface of the cell, this control point are represented in blue.

3.2 Original force evaluation.

From equation 5.1, the total force acting on the cylinder is the sum of the forces acting on each single control point The force on each single control point (red points in figure 3.2) is:

$$\mathbf{F}_i = \Big(p_i + \nabla \mathbf{u}_i\Big) \cdot S_i \cdot \mathbf{n} \tag{3.4}$$

In order to evaluate it, interpolations of the velocity gradient and pressure are needed so we need a criteria to chose the neighbors to use. Referring to figure 3.2 from the i-th control point we move to the closest vertex and we chose as neighbors the four cells insisting on the vertex.

The pressure is defined both in solid and fluid cells, but the gradient of velocity is defined only in fluid cells as the velocity is null inside the body. So to interpolate we need to select only fluid cells that can be used to interpolate the force. The strategy is to evaluate $p_s + \nabla \mathbf{u}_s$ for each fluid neighbor (cells $s = [1\ 3\ 4]$ in figure). The value of the pressure is known in the cell centers of the neighbors cell and the gradient in each s-th point can be approximate using a second order finite difference scheme.

$$\mathbf{F}_i = \frac{1}{N} \sum_s \Big(p_s + \nabla \mathbf{u}_s\Big) \cdot S_i \cdot \mathbf{n}_i$$

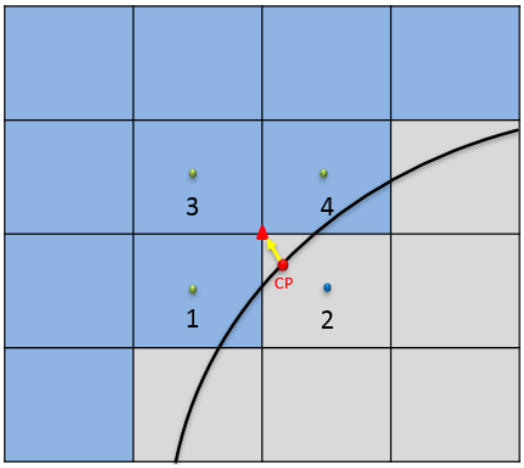
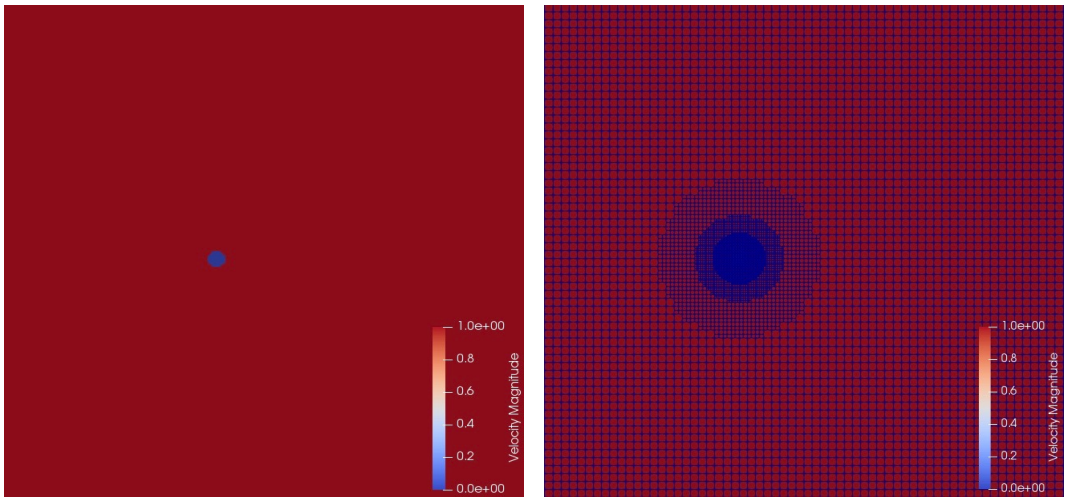


Figure 3.2: the Neighbors used for the interpolation are selected moving from the control point to the nearest vertex and the neighbors are the 4 cell insisting on this vertex.

3.2.1 Results

With the method previously presented we obtain the two components of the total force on the surface of the cylinder $\mathbf{F} = [F_x\ F_y]$. The previous works found in literature present the non-dimensional force. The *Drag coefficient* is computed and used to compare the result. The reference value obtained in [9] is used to evaluate the error on the C_D.
In table 3.1 the result in terms of drag coefficient are presented. The simulation is performed for different grid both uniform and quadtree with circular refinement.

$$C_D = \frac{F_x}{1/2\rho u^2 S} \tag{3.5}$$



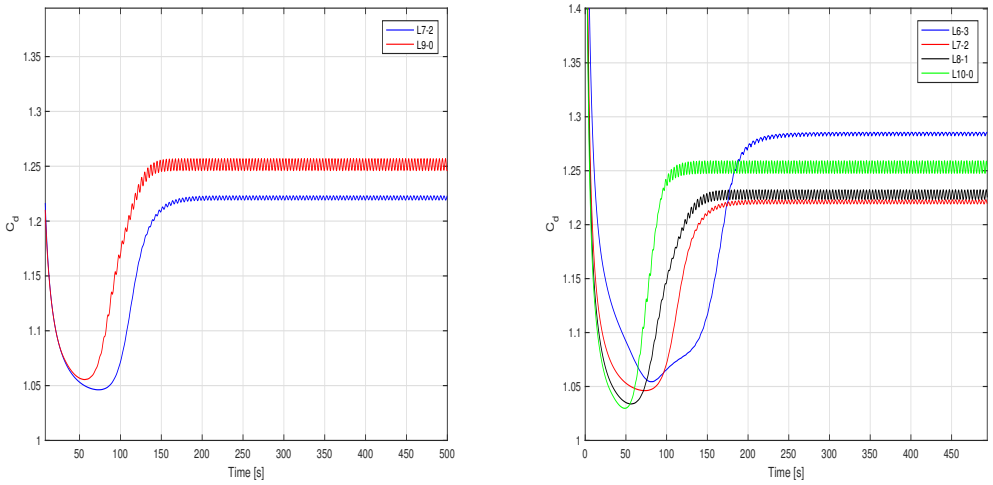
(a) (b)

Figure 3.3: The fluid domain used for the test, in 3.3a the velocity field at the initial condition, in 3.3b the quadtree grid with initial level 7 and three circular refinement around the surface of the cylinder.

This results are the same already used to compare the quadtree grid to the uniform cartesian grid presented in the previous chapter.

Table 3.1: The drag coefficient obtained with several grids both cartesian or quadtree, the error respect to the reference value and the computational time needed to reach 500 seconds of physical time.

Grid	Cd_{av}	$err\%$	Tcpu [h]	#cell
L10-0	$1,2534$	$6,13\%$	$12:42:13$	1048576
L9-1	$1,2483$	$6,52\%$	$05:08:42$	441856
L8-2	$1,2756$	$4,47\%$	$02:40:09$	179584
L9-0	$1,2515$	$6,28\%$	$01:48:30$	262144
L7-3	$1,2620$	$5,49\%$	$01:41:47$	84832
L8-1	$1,2278$	$8,05\%$	$01:07:26$	110464
L7-2	$1,2212$	$8,54\%$	$00:38:59$	44896
L6-3	$1,2843$	$3,82\%$	$00:26:27$	21208



(a) The comparison between the *Drag Coefficient* obtained using a uniform cartesian grid and a quadtree grid with circular refinement

(b) the result are compared with the C_D obtained using different discretizations

Figure 3.4: Original method: The results in terms of *drag coefficient* obtained using the Lagrangian markers method. The forces on the control points are calculated averaging the force on the neighbors.

3.3 New method to evaluate the Forces

The method presented above is not very accurate as the force result from an average of the values of the neighbors, and so doing we have to sum to the interpolation error, the error made approximating of the value of the velocity gradient in the neighbor's centers. In order to improve the accuracy of the method, p and $\nabla \mathbf{u}$ will be interpolated o the control point directly, only knowing p and $\mathbf{u}$ in each neighbor cell center. With this method we don't need to approximate the velocity gradient on the neighbors, so the error is expected to be smaller as we only have the interpolation error.

3.3.1 Interpolation of the Pressure with RBF scheme

In practical applications we have to face the problem of reconstructing an unknown function f from a set (usually small) of data. These data consist of two sets: the data sites $X = x_1, ..., x_N$ and the data values $f_j = f(x_j)$, $j = 1, ..., N$. The reconstruction has to approximate the data values at the data sites. In practice we are looking for a function s that either interpolates the data, i.e. it satisfies the conditions $s(xj) = fj$, $1 \leq j \leq N$ or approximate the data, i.e. $s(x_j) \backsimeq f_j$. This latter is important, for example, when the data come from some measurement or contain noise.
A radial basis function is a function ϕ whose value depends only from the distance from the origin or alternatively on the distance from a generic point called $\mathbf{c}$, so that

$$\phi(\mathbf{x}, \mathbf{c})) = \phi(||\mathbf{x} - \mathbf{c}||) \tag{3.6}$$

Such functions, as wrote, depend only from the radial distance from a fixed point. The norm usually is the Euclidean distance.Sums of radial basis functions are typically used to approximate given functions. RBFs functions are typically used to build up function approximations of the form

$$y(x) = \sum_{i=1}^{N} w_i \phi(||\mathbf{x} - \mathbf{x_i}||) \tag{3.7}$$

Where, our approximating function has been written as a sum of N radial basis functions, each associated with a different center and with a proper weight w_i. Such weights can be estimated using the matrix method of linear least squares.

3.3.2 Interpolation of the gradient with the least square interpolation

To interpolate the velocity gradient on the control point a *least square* interpolation approach has been used. This allow to obtain the value of the gradient on the desired point without the need to compute the gradient on the neighbors cell center first. So the error made is only the interpolation error and there is no error committed approximating the gradient on the neighbors anymore.

Denoting with i the index of the neighbors, the Taylor expansion of the velocity on the i-th point is

$$\mathbf{u}_{(x_i)} = \mathbf{u}_{(x_0)} + (\mathbf{X_i} - \mathbf{X_0})\nabla \mathbf{u}_{(x_0)} \tag{3.8}$$

Where for the two-dimensional case:

- $\mathbf{u} = [u\ v] \in \mathbb{R}^2$
- $\mathbf{X} = [x\ y] \in \mathbb{R}^2$
- $\nabla \mathbf{u} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} \in \mathbb{R}^{2\times 2}$

Interpolating in the sense of the least square is based on minimizing the sum, of all the distances between the values in the neighbors, used to interpolate, and the interpolated value itself, From the 3.8 we can get the sum of the distances.

$$I = \sum_i \frac{1}{2} \Big[\mathbf{u_{x_i}} - \mathbf{u_{x_0}} - \nabla \mathbf{U}\|_{x_0} (\mathbf{X_i} - \mathbf{X}) \Big]^2 \tag{3.9}$$

This is a vectorial equation, in order to minimize this quantity the derivative is taken with respect to the gradient of the velocity, and imposed null. To simplify we will split the gradient into two vectors, consequently the equation will be split in two equations as well.

$$\nabla \mathbf{u} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = \left\{ \begin{array}{c} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{array} \right\} \cup \left\{ \begin{array}{c} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{array} \right\} \tag{3.10}$$

$$\nabla \mathbf{U}\|_{x_0}(\mathbf{X_i} - \mathbf{X}) = \sum_{d=1}^{2}(\mathbf{X_i^d} - \mathbf{X^d}) \tag{3.11}$$

Where d is an index that represent the two vectors in which the gradient was divided. According to that and expanding the square, the 3.9 can be rewritten.

$$I = \sum_i \frac{1}{2}\Big\{\Delta \mathbf{u}^2 - 2\Delta \mathbf{u}_i \nabla \mathbf{u}\|_{x_0}(\mathbf{X_i} - \mathbf{X}) + \Big[\sum_{d=1}^{2}(\mathbf{X_i^d} - \mathbf{X^d})\Big]^2\Big\} \tag{3.12}$$

Where for convenience $\Delta \mathbf{u} = \mathbf{u_{x_i}} - \mathbf{u_{x_0}}$. Now we take the derivative of each component of the 3.12 with respect to the two vectors in which the gradient was divided. and we obtain two linear sistems.

$$\begin{cases} \frac{\partial I_u}{\partial \nabla_x u} = 0 \\ \frac{\partial I_u}{\partial \nabla_y u} = 0 \end{cases} \tag{3.13}$$

$$\begin{cases} \frac{\partial I_v}{\partial \nabla_x v} = 0 \\ \frac{\partial I_v}{\partial \nabla_y v} = 0 \end{cases} \tag{3.14}$$

Where $\nabla_x = \frac{\partial}{\partial x}$, $\nabla_y = \frac{\partial}{\partial y}$ and $I = \{I_u\ I_v\}$.
The system 3.13 is obtained deriving the first component of I and imposing it equal to zero we can evaluate the first two components of the gradient.

$$\begin{cases} \frac{\partial I_u}{\partial \nabla_x u} = \sum_i \frac{1}{2}\Big[0 - 2\Delta \mathbf{u}_i(x_i - x) + 2\Big((x_i - x)\frac{\partial u}{\partial x} + (y_i - y)\frac{\partial u}{\partial y}\Big)(x_i - x)\Big] = 0 \\ \frac{\partial I_u}{\partial \nabla_y u} = \sum_i \frac{1}{2}\Big[0 - 2\Delta \mathbf{u}_i(y_i - y) + 2\Big((x_i - x)\frac{\partial u}{\partial x} + (y_i - y)\frac{\partial u}{\partial y}\Big)(y_i - y)\Big] = 0 \end{cases} \tag{3.15}$$

$$\begin{cases} \sum_i \nabla_x u(x_i - x)^2 + \sum_i \nabla_y u(x_i - x)(y_i - y) = \sum_i \Delta u_i(x_i - x) \\ \sum_i \nabla_x u(y_i - y)^2 + \sum_i \nabla_y u(x_i - x)(y_i - y) = \sum_i \Delta u_i(y_i - y) \end{cases} \tag{3.16}$$

And rearranging in matrix form we obtain the linear system implemented in the code used to interpolate the gradient in the control point.

$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 \end{bmatrix} \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \sum_i (x_i - x)\Delta u_i \\ \sum_i (y_i - y)\Delta u_i \end{Bmatrix} \tag{3.17}$$

The system 3.14 is obtained deriving the second component of I and imposing it equal to zero we can evaluate the last two components of the gradient.

$$\begin{cases} \frac{\partial I_v}{\partial \nabla_x v} = \sum_i \frac{1}{2}\Big[-2\Delta \mathbf{u}_i(x_i - x) + 2\Big((x_i - x)\frac{\partial v}{\partial x} + (y_i - y)\frac{\partial v}{\partial y}\Big)(x_i - x)\Big] = 0 \\ \frac{\partial I_v}{\partial \nabla_y v} = \sum_i \frac{1}{2}\Big[0 - 2\Delta \mathbf{u}_i(y_i - y) + 2\Big((x_i - x)\frac{\partial v}{\partial x} + (y_i - y)\frac{\partial v}{\partial y}\Big)(y_i - y)\Big] = 0 \end{cases} \tag{3.18}$$

$$\begin{cases} \sum_i \nabla_x v(x_i - x)^2 + \sum_i \nabla_y v(x_i - x)(y_i - y) = \sum_i \Delta v_i(x_i - x) \\ \sum_i \nabla_x v(y_i - y)^2 + \sum_i \nabla_y v(x_i - x)(y_i - y) = \sum_i \Delta v_i(y_i - y) \end{cases} \tag{3.19}$$

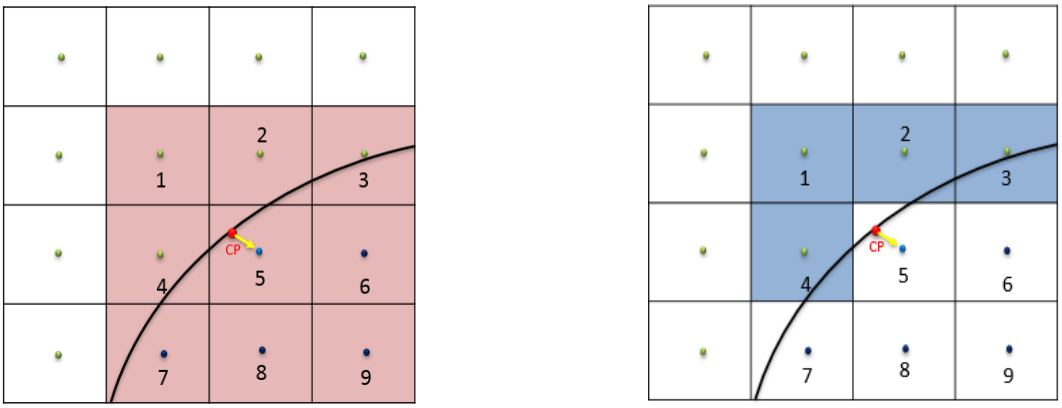
And rearranging in matrix form we obtain the linear system implemented in the code used to interpolate the gradient in the control point.

$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 \end{bmatrix} \left\{ \begin{array}{c} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{array} \right\} = \left\{ \begin{array}{c} \sum_i (x_i - x)\Delta v_i \\ \sum_i (y_i - y)\Delta v_i \end{array} \right\} \tag{3.20}$$

This two system can be easily be solved using a PETSC library function to invert the matrix. As is it possible to notice the matrix is the same for both system so it will be inverted once by the code, this allow to spare computational time and memory.

Neighbors selection

The preceding method didn't have many restrictions in terms of neighbors selection, the method could be used with any number of fluid neighbors of the cell containing the control point. However with this new method it is important to guarantee always the existence of at least 3 fluid neighbors, as in the case brought in figure 3.5a. So the choice of the neighbors used to interpolate the gradient consist in finding the closest grid cell center to the control point, all the border cell are taken into account, as well as the cell itself. In order to guarantee the existence of the gradient in the neighbors, the interpolation is made using only the fluid cell among the nine taken into account.



(a) Pressure

(b) Velocity gradient

Figure 3.5: From control point, in red, we move to the closest grid cell center. All the bordering cell, of this cell are taken into acount, also the cell itself is added. Only the fluid cell among the 9 neighbors are used to interpolate the gradient, while all the neighbors are used to interpolate the pressure.

3.4 Convergence Analysis

The aim of this section is to find a different method to interpolate the forces on the control point that can improve the accuracy of the computation, but without compromising the computational time. We want to evaluate the convergence rate of the method through a convergence analysis. An analytic field is imposed and the numerical solution is compared with the analytical solution.

The analytical solution Φ can be expressed as the numerical solution ϕ pus a discretization error ϵ_d.

$$\Phi = \phi + \epsilon_d \tag{3.21}$$

The discretization error can be estimated from the truncation error of the Taylor expansion $\epsilon_d = \alpha \Delta x^n + H$.

$$||\epsilon_d|| = ||\Phi - \phi|| = O(\Delta x)^n \tag{3.22}$$

Taking the logarithm of both sides of the equation 3.22

$$log(||\Phi - \phi||) = nlog(\Delta x) \tag{3.23}$$

$$log(||\epsilon_d||) = nlog(\Delta x) \tag{3.24}$$

From a graphical point of view the slope of the line identified plotting $log(||\epsilon||)$ in function of $log(||\Delta x||)$ is the convergence rate p.

We can use the the definition of the truncation error to verify the order of accuracy of a discretization scheme. If we have at least three solutions obtained on systematically refined grids, for example with spacing $4\delta x$, $2\delta x$ and δx, we can write:

$$\begin{cases} \phi_{\Delta x} + \alpha \Delta x^n + H = \phi_{2\Delta x} + \alpha (2\Delta x)^n + H \\ \phi_{2\Delta x} + \alpha (2\Delta x)^n + H = \phi_{4\Delta x} + \alpha (4\Delta x)^n + H \end{cases} \tag{3.25}$$

$$\begin{cases} \phi_{\Delta x} - \phi_{2\Delta x} = \alpha (2\Delta x)^n - \alpha \Delta x^n = \alpha \Delta x^n (2^n - 1) \\ \phi_{2\Delta x} - \phi_{4\Delta x} = \alpha (4\Delta x)^n - \alpha (2\Delta x)^n = \alpha 2^n \Delta x^n (2^n - 1) \end{cases} \tag{3.26}$$

If we now make the ratio of the two equations above, we get:

$$\frac{\phi_{2\delta x} - \phi_{4\delta x}}{\phi_{\Delta x} - \phi_{2\delta x}} = 2^n \tag{3.27}$$

$$n = \frac{log \frac{\phi_{2\delta x} - \phi_{4\delta x}}{\phi_{\Delta x} - \phi_{2\delta x}}}{log 2} \tag{3.28}$$

The ratio of the error obtained halving the cell dimensions is equal to 2^n. This means that if we have a $n = 1$ it is a first order convergence order and $\epsilon_{d(\Delta x)} = 2\epsilon_{d(2\Delta x)}$, if we have a second order accuracy $n = 2$ and $\epsilon_{d(\Delta x)} = 4\epsilon_{d(2\Delta x)}$

3.4.1 Convergence order for a single point

The convergence analysis is performed on a single control point first, exact and numerical solution for pressure and gradient of the velocity will be evaluated after imposing simple analytical fields to pressure and velocity.
The coordinate of the point are represented in table 3.2
Such point is the first control point considered in the code. The analytical field imposed is represented in table 3.3. The function $sin^2(x)$ for the pressure ensure the pressure has always a positive value.

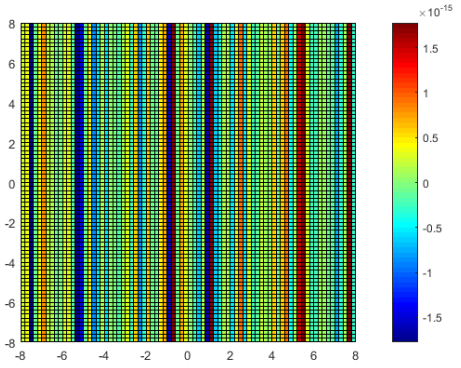
Table 3.2: Coordinate of the first control point

Control point coordinate	
x	y
$-3,00049$	$0,0314108$

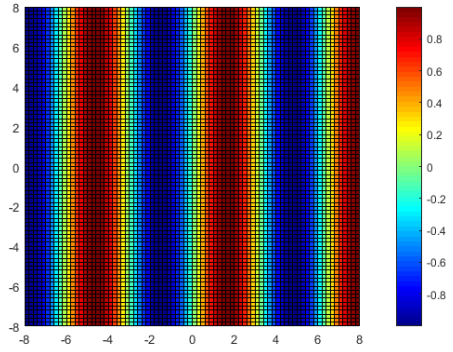
Table 3.3: Imposed Flow field

Imposed field	
u	$sin(x)$
v	$cos(y)$
P	$sin^2(x)$

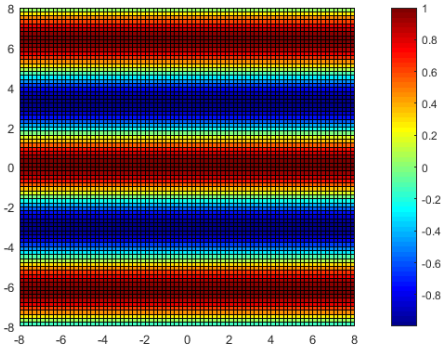
In table 3.4 are shown the analytic fields of pressure and gradient.



(a) $p = \cos^2 x$



(b) $u = \sin x$



(c) $v = \cos y$

Table 3.4: Analytical ∇V of velocity and pressure value for the control point of table 3.2

∇V analytic				P analytic
du/dx	du/dy	dv/dx	dv/dy	-
$-0,99006$	0	0	$-0,03141$	$0,019778174$

∇V convergence order

In table 3.5 are presented the numerical result obtained with different levels of refinement. The errors presented in table 3.6 are evaluated subtracting the numerical solution from the analytical solution.

Table 3.5: Numerical ∇V over one single point

Level	du/dx	du/dy	dv/dx	dv/dy	Δx
6	$-0,9706124$	$0,005310934$	$-0,082124$	$-0,3313$	$0,5$
7	$-0,9786534$	$-3,6717 \cdot 10^{-}17$	$-0,13009$	$-0,12338$	$0,25$
8	$-0,984979$	$-3,64299 \cdot 10^{-}17$	$-0,07483$	$-0,0623$	$0,125$
9	$-0,987664$	$9,80707 \cdot 10^{-}18$	$-0,04099$	$-0,03122$	$0,0625$
10	$-0,988886$	$-6,95525 \cdot 10^{-}17$	$-0,01277$	$-0,04685$	$0,03125$
11	$-0,989467$	0	$-0,00627$	$-0,03905$	$0,015625$
12	$-0,989749$	$5,83438 \cdot 10^{-}17$	$-0,00303$	$-0,03515$	$0,007813$

Only the component du/dx has been token into account for the convergence analysis, since the errors on the components dv/dx and du/dy are about the same order of magnitude of the machine precision, so would not be significant a convergence analysis.

Table 3.6: ∇V convergence order over one single point

Level	$\varepsilon_{du/dx}$	$\varepsilon_{du/dy}$	$\varepsilon_{dv/dx}$	$\varepsilon_{dv/dy}$	Δx
6	$-0,01945$	$-0,00531093$	$0,082118$	$0,299895$	$0,5$
7	$-0,01141$	$3,6717 \cdot 10^{-}17$	$0,130089$	$0,091974$	$0,25$
8	$-0,00508$	$3,64299 \cdot 10^{-}17$	$0,074831$	$0,030891$	$0,125$
9	$-0,0024$	$-9,80707 \cdot 10^{-}18$	$0,040985$	$-0,00018$	$0,0625$
10	$-0,00118$	$6,95525 \cdot 10^{-}17$	$0,012767$	$0,015445$	$0,03125$
11	$-0,00059$	0	$0,006269$	$0,007645$	$0,015625$
12	$-0,00031$	$-5,83438 \cdot 10^{-}17$	$0,003028$	$0,003743$	$0,007813$

To evaluate the rate of convergence we need to plot the $log||\epsilon||$ in function of the $log||\Delta x||$, and evaluate the slope of the linear regression line. The slope is exactly the convergence order n.

As expected this method of interpolation of the velocity gradient does not increase the rate of convergence respect to the previous method, but the errors are smaller and the accuracy is improved.

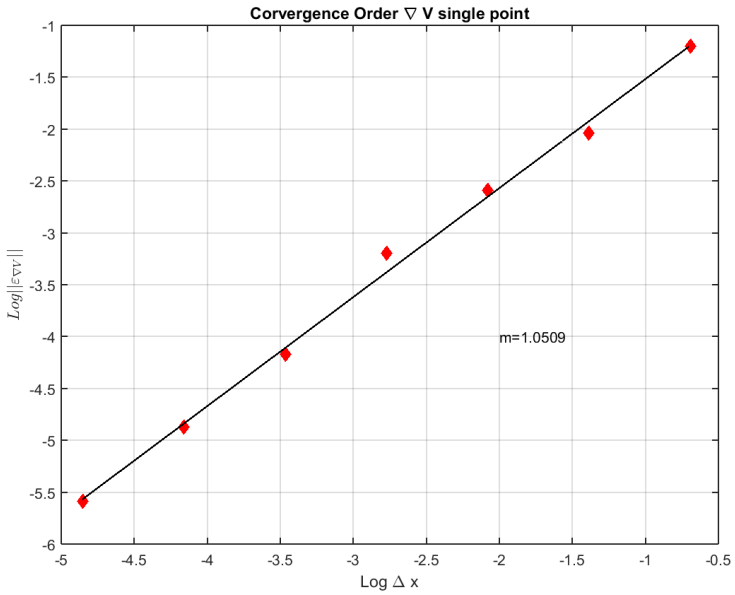


Figure 3.7: Convergence order of the gradient interpolation using *least square*.

Pressure convergence order

The same approach is used for the convergence analysis on the pressure interpolation. An analytical field for the pressure is imposed, so the exact value is known in each cell of the domain. The pressure is interpolated in the control points using an *RBF* interpolation. The interpolated pressure will be compared to the analytical solution to evaluate the absolute error of the method.

Table 3.7: Pressure convergence order over one single point

Error on the pressure			
Level	p	$Log\|\varepsilon_p\|\|\|$	$Log\Delta x$
6	$0,07855340$	$-2,83403483$	$-0,69315$
7	$0,03470310$	$-4,20472255$	$-1,38629$
8	$0,02352310$	$-5,58735333$	$-2,07944$
9	$0,02071440$	$-6,97365327$	$-2,77259$
10	$0,02001140$	$-8,36350107$	$-3,46574$
11	$0,01983560$	$-9,76500712$	$-4,15888$
12	$0,01979160$	$-11,21829058$	$-4,85203$

In order to evaluate the convergence rate, the bi.logarithmic graph is made as presented above and the convergence rate n is the slope of the regression line.

Inside the code an object that interpolates with an RBF method is was already implemented, in the class called *interpolator.tpp*.

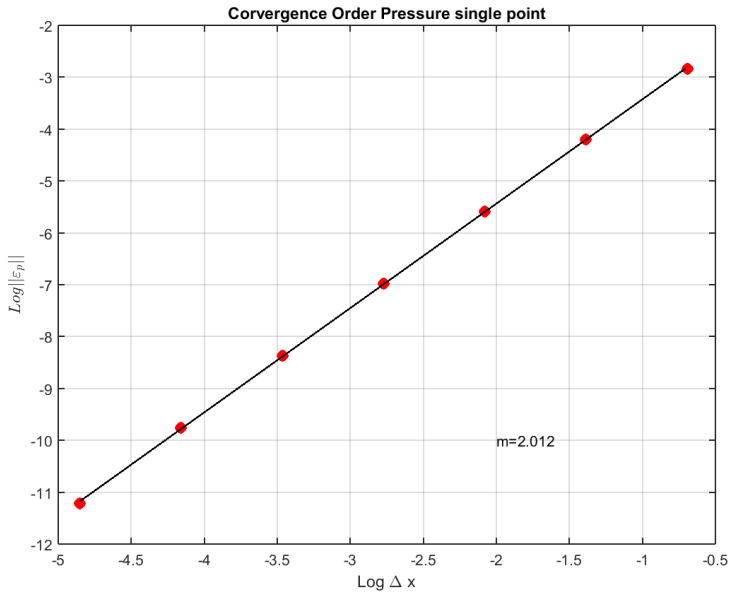


Figure 3.8: Convergence order of the Pressure interpolation using RBF

3.4.2 Convergence Order for the whole cylinder surface

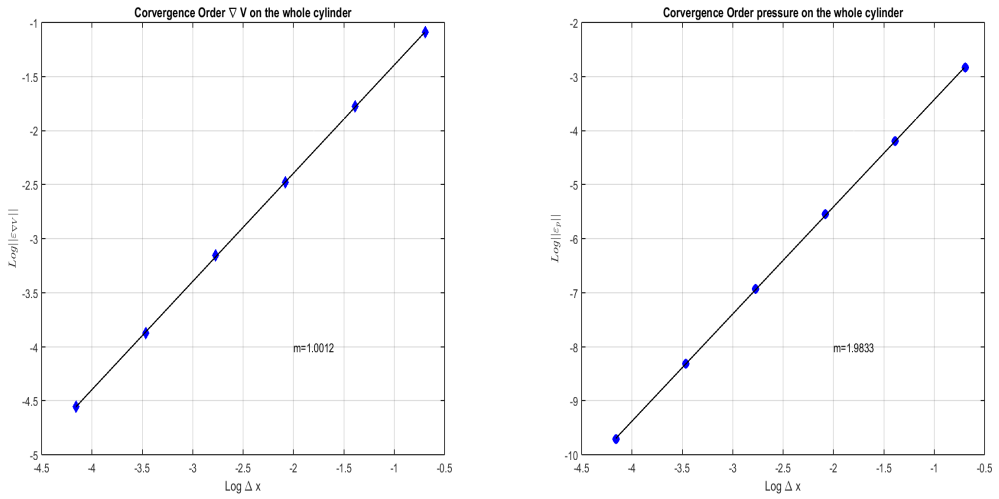
Until this moment all the convergence order test has been done using just the first control point identified by the coordinate shown in 3.2. Once the convergence on a single control point has been achieved is necessary to verify if such convergence is confirmed integrating over the surface of the cylinder.

We use the same analytical fields imposed above to evaluate the analytical and numerical values of p and $\nabla\mathbf{u}$. In this case we will consider the convergence of the sum of the solutions evaluated on each control point.

The graph reported in figure 3.9 represent the convergence order of the first component of the gradient.

Table 3.8: Convergence analysis over all the Control points of the geometries

Level	$\|\|\varepsilon_{\nabla V}\|\|$	$\|\|\varepsilon_P\|\|$	$Log\|\|\varepsilon_{\nabla V}\|\|$	$Log\|\|\varepsilon_P\|\|$	$Log(\Delta x)$
6	$0,335492$	$0,058803$	$-1,092157168$	$-2,8335641$	$-0,69315$
7	$0,169286$	$0,015$	$-1,776165687$	$-4,1997051$	$-1,38629$
8	$0,083714$	$0,00389$	$-2,480352635$	$-5,5493204$	$-2,07944$
9	$0,042739$	$0,000973$	$-3,152645766$	$-6,9346477$	$-2,77259$
10	$0,020799$	$0,000243$	$-3,87285037$	$-8,3207345$	$-3,46574$
11	$0,010489$	$6,08E-05$	$-4,557466326$	$-9,7071201$	$-4,15888$



(a) $\nabla \mathbf{u}$ (b) p

Figure 3.9: The convergence order for $\nabla \mathbf{u}$ and p considering the whole cylinder surface.

Even if te convergence rate is nt significantly improoved from the original method,first order rate of convergence on the ∇V and a second order for the pressure, we expect a much more accurate evaluation of the forces using this second approach.

3.4.3 Results

To test the New method, the simulation of the same test case used before has been performed. We tested the method using an uniform cartesian grid first. The drag coefficient has been evaluated and the results are then compared tho the C_D obtained using quadtree grids with different discretization levels.. In table 3.9 are reported the Drag coefficients obtained with the improved Lagrangian Markers Method on different grids. As already done before the results obtained with the uniform grid at level 9 and 10 are compared with the C_D obtained with differently discretized quadtree grids. The reference value is the results proposed from [9] for a flow past a cylinder at reynolds 100.

$$C_{D_{REF}} = 1,3353$$

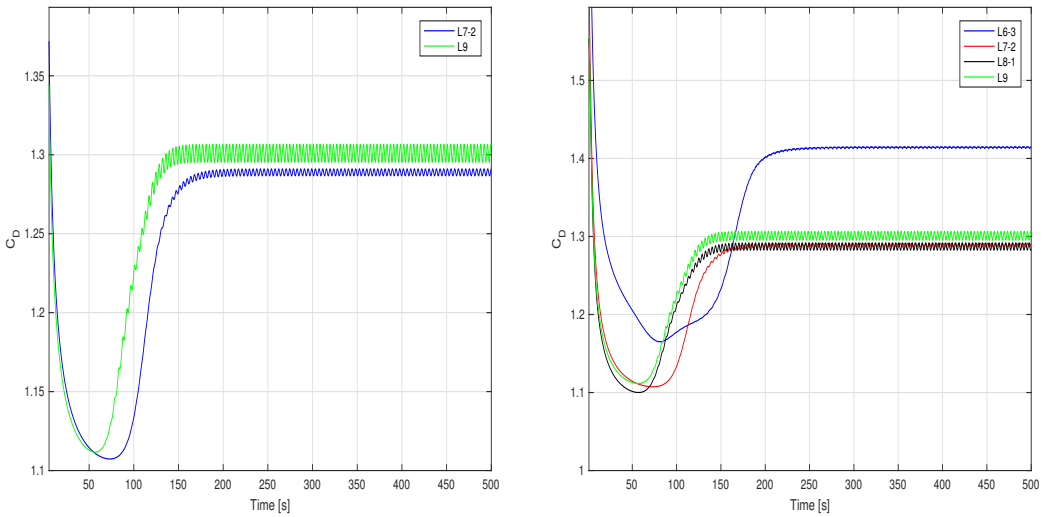
The relative error is evaluated as:

$$err\% = \frac{C_D - C_{D_{REF}}}{C_{D_{REF}}} \times 100$$

Table 3.9: Results in terms of drag coefficient, obtained with the new approach to the Lagrangian marker method. Is it possible to notice that using a quadtree discretization of the domain the computational time is much shorter and the number of cells is much smaller.

Grid	Cd_{av}	$err\%$	T_{cpu} [h]	#Cells
L10	1,318	1,30 %	10h:17m:16s	1056820
L9-1	1,3599	1,84 %	06h:27m:14s	500000
L8-2	1,3158	1,46%	03h:24m:18s	115554
L9	1,3009	2,58 %	01h:48m:01s	266443
L7-3	1,3269	0,63%	01h:41m:12s	38143
L8-1	1,2872	3,60 %	01h:08m:35s	89572
L7-2	1,2888	3,48 %	00h:41m:41s	29780
L6-3	1,4142	5,91 %	00h:27m:00s	10124

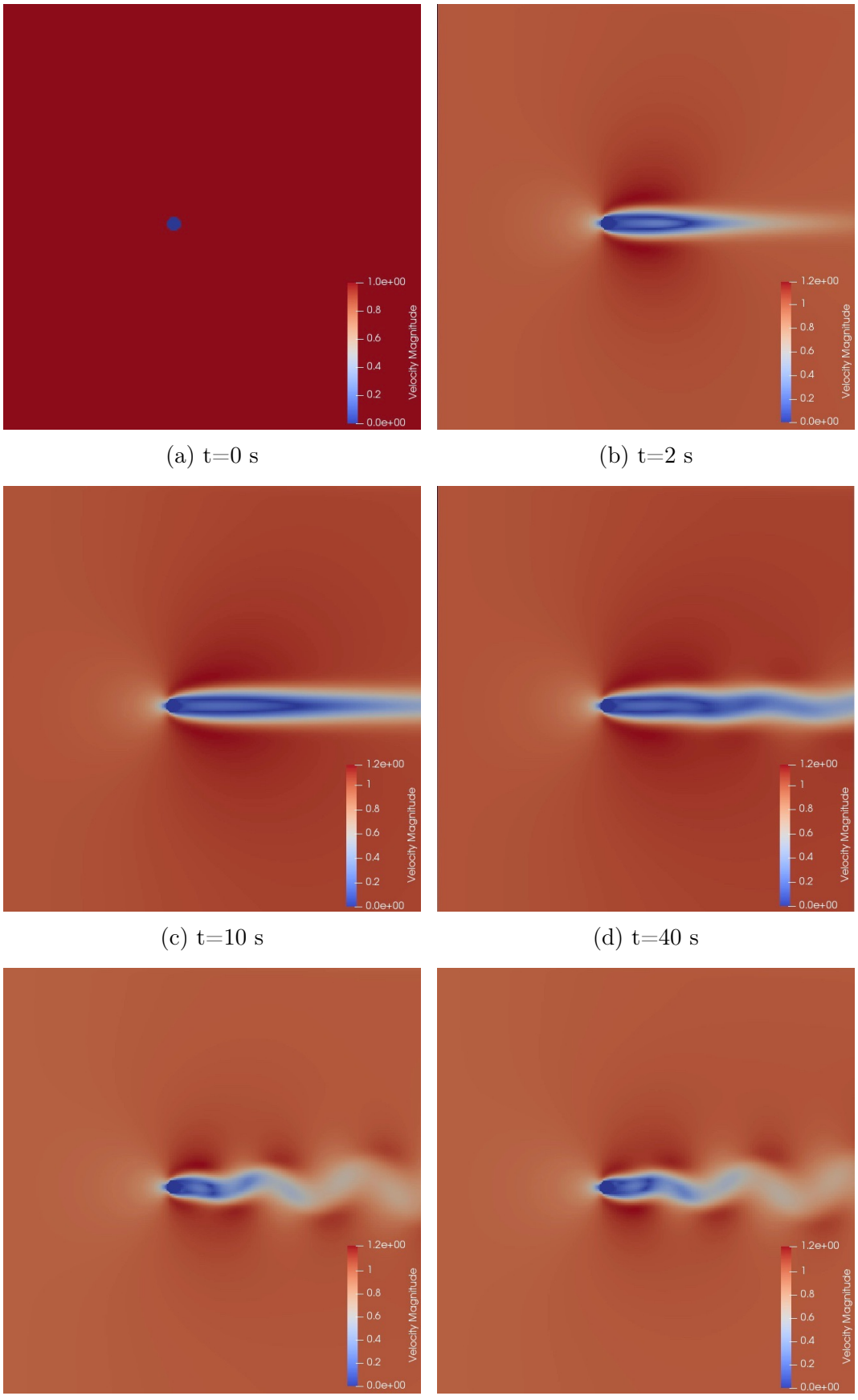
The *Drag Coefficients* are computed using different quatree discretization, all with the same refinement lever around the surface of the cylinder. The result (table 3.9)are compared with the forces obtained using a uniform cartesian grid at the same level of the fines quadtree refinement.



(a) The comparison between the C_D obtained using a cartesian uniform grid al Level 9 and a quadtree grid at Level 7 with 2 circular refinement.

(b) The results compared with the C_D obtained using several grids with differen discretizedation.

Figure 3.10: The results for the forces, expressed in terms of *Drag Coefficient* obtained using the improved Lagrangian markers method. The forces on the control points are evaluated interpolating the pressure using an RBF interpolation and the gradient using the least square methods.



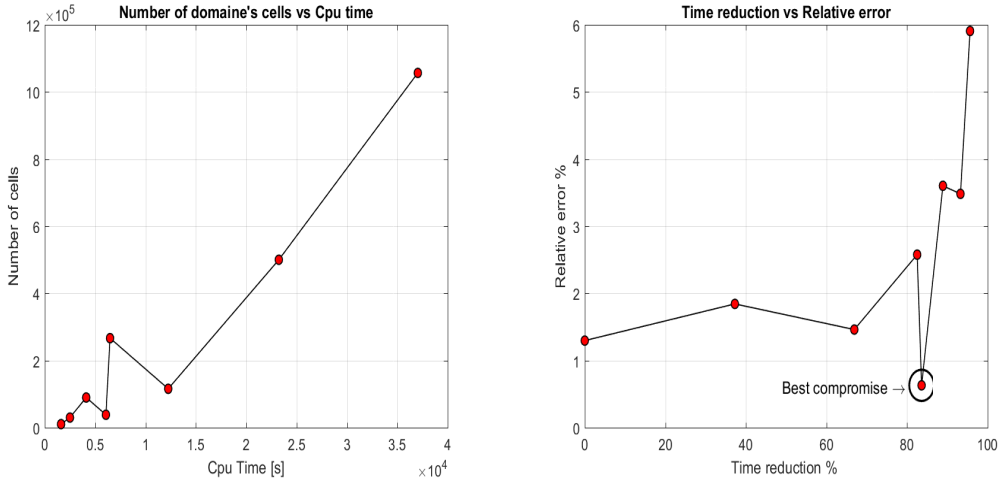
(a) t=0 s (b) t=2 s

(c) t=10 s (d) t=40 s

(e) t=100 s (f) t=500 s

Figure 3.11: The velocity field of the flow past a unit diameter cylinder at $Re = 100$. In 3.11a are reported the initial conditions t=0s, in 3.11fthe field after the end of the transient, the flow is completely developed at t=500s

In order to underline again the convenience in using a quadtree mesh instead of a uniform cartesian grid we can focus on figure 3.12 , where are presented the same graphics already seen in section 2.3. In 3.12a is presented, for each simulation performed , the number of cell used in function of the cpu time needed to reach 500 seconds of physical time. Is it possible to see how this behavior is nearly linear, a part for the more coarse grids. In 3.12b is plotted the relative error respect to the reference value from [9], in function of the percentage of saved computational time that comes from using a quadtree grid. As seen before the discretization that gives the best result in a shorter time is the grid 7-3 that presents a relative error of $0,63\%$ saving more than 60% of computational time.



(a) The cpu time needed to reach 500 s of physical time versus the number of cell of the discretized domain.

(b) The reduction in time respect to the cartesian grid versus the percentage error.

Figure 3.12: From the graphs above is it possible to chose the discretization that can guarantee the best compromise between a fast computation and a small error.

3.5 Comparison between the two methods used to interpolate the force on the control points.

In this section the result obtained above with the new method are compared with the original forces evaluated averaging the forces on the neighbors. We will compare the relative errors obtained using the same grids. The improvement using the interpolation of pressure and gradient is clear from table 3.10. The relative reports a significant reduction using the second method. For the uniform grid at level 10 the error using the first method is $6,13\%$ while it decrease to 1.30% using the second approach. In addition to this the error for a quadtree grid with initial level 7 with 3 refinement in $< 1\%$. The accuracy of the methods for the evaluation of the forces is significantly improved interpolating the pressure and gradient on the points of the cylinder.

Table 3.10: Comparison between the Drag coefficient obtained with the previous approach and the results of the new approach.

Grid	C_DI	err%	C_DII	err%
$L10$	$1,2534$	$6,13\%$	$1,318$	$1,30\%$
$L9-1$	$1,2483$	$6,52\%$	$1,3599$	$1,84\%$
$L8-2$	$1,2756$	$4,47\%$	$1,3158$	$1,46\%$
$L9$	$1,2515$	$6,28\%$	$1,3009$	$2,58\%$
$L7-3$	$1,262$	$5,49\%$	$1,3269$	$0,63\%$
$L8-1$	$1,2278$	$8,05\%$	$1,2872$	$3,60\%$
$L7-2$	$1,2212$	$8,54\%$	$1,2888$	$3,48\%$
$L6-3$	$1,2843$	$3,82\%$	$1,4142$	$5,91\%$

The behavior of the Drag coefficient curve in time in time is not affected from this improvement, the curve is translated towards the exact value.

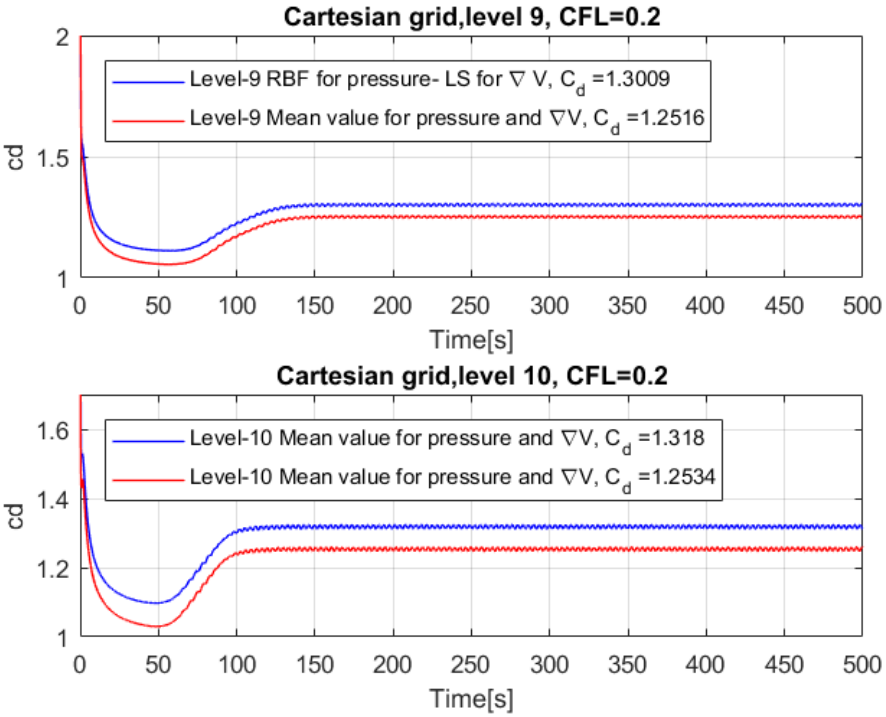


Figure 3.13: The *Drag Coefficient* obtained with the new method is compared with the result previously obtained for two different uniform grids.

3.6 Noca method

In this section is presented an other way to evaluate the fluid dynamic force around a rigid body immersed in a fluid domain, using a moving control volume approach [11]. This method consist in evaluating the volume integral over an arbitrary control volume. The force will be the variation in time of the momentum contained in the control volume. In two dimensions this can be seen

as the difference between the incoming momentum inside the control volume and the out-coming momentum, but as this method will be after extended to a 3D case we will present a more general case. Applying the conservation of linear momentum we get:

$$\frac{d}{dt}(m\cdot\mathbf{u}) = \mathbf{F} = -\frac{d}{dt}\int_V \rho\mathbf{u}dV + \int_{S_{CV}} \mathbf{n}\cdot[-p\mathbf{I}-(\mathbf{u}-\mathbf{u}_S)\rho\mathbf{u}+\mathbf{T}]dS - \int_{S_b} \mathbf{n}\cdot(\mathbf{u}-\mathbf{u}_S)\rho\mathbf{u}dS \tag{3.29}$$

where T is the viscous stress tensor:

$$\mathbf{T} = \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{3.30}$$

and $\mathbf{n}$ is the normal unit vector to the surface of the control volume, positive if pointing outwards. V is the volume contained in the control volume without taking into account the volume of the body itself, $\mathbf{u}_S$ is the velocity of the surface over which the volume is taken, i.e. is the velocity of S_{CV} and can generally be chosen arbitrary to always contain the rigid body if it is in motion $\mathbf{u} \neq \mathbf{u}_S$. But in our case the control volume is fixed as the rigid body is not moving and $\mathbf{u}_S = 0$. The equation became:

$$\mathbf{F} = -\frac{d}{dt}\int_V \rho\mathbf{u}dV + \int_{S_{CV}} \mathbf{n}\cdot[-p\mathbf{I} - \mathbf{u}\rho\mathbf{u} + \mathbf{T}]dS \tag{3.31}$$

$$\mathbf{F} = -\frac{d}{dt}\int_V \rho\mathbf{u}dV + \int_{S_{CV}} \mathbf{n}\cdot[\mathbf{T} - p\mathbf{I}]dS + \int_{S_{CV}} \mathbf{u}\rho(\mathbf{n}\cdot\mathbf{u})dS \tag{3.32}$$

Discretizing each terms of the equation to make is suitable for the integration in the code we get:

$$\sum_i \mathbf{F} = \frac{\rho}{\Delta t}\Big[\sum_i \mathbf{u}_i^{n+1}\cdot V_i - \sum_i \mathbf{u}_i^n \cdot V_i\Big] + \sum_i \rho(\mathbf{T}-p\mathbf{I})_i\cdot\mathbf{n}_i\cdot S_i + \sum_i \rho\mathbf{u}_i(\mathbf{n}_i\cdot\mathbf{u}_i)\cdot S_i \tag{3.33}$$

3.6.1 The control volume

The case presented is the same used in the previous chapter, a flow past a cylinder at Re 200, but in this case we use a CFL condition of 0.2. The same simulations will be performedThe control volume to evaluate the fluid dynamic forces can be chosen arbitrary. The one we used is a rectangle containing the cylinder as shown in figure 3.14. The sides of the cylinder is set to coincide with the interface between the cell in order to obtain an easier evaluation of the volume integral. to define the control volume in the code we take as reference a cartesian grid at level 6 that correspond to a cell dimension of $\Delta x = \Delta y = 0.5$. Refining the grid or using a quadtree mesh will keep the sides of the control volume on the interface between the cell.

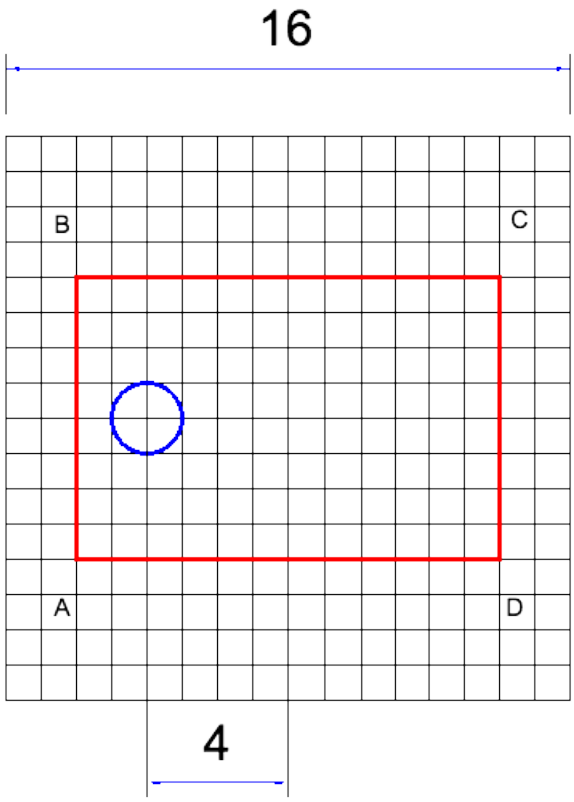


Figure 3.14: The control volume defined for this problem is a rectangle containing the cylinder

The control volume is defined through imposing the coordinates of the rectangle vertex. Referring to figure 3.14 the coordinates of the vertexes will be:

1. Point A $(-4 - N\Delta x, -N\Delta y)$
2. Point B $(-4 - N\Delta x, N\Delta y)$
3. Point C $(-4 + 2N\Delta x, N\Delta y)$
4. Point D $(-4 + 2N\Delta x, -N\Delta y)$

Where N is an integer chosen to define the size of the control box for us $N = 4$. The control points are chosen by the user, in this case we use 200 control point on the longer side and 100 points on the shortest side of the rectangle. A the longest side is defined to be twice as long as the short side, the distance between two control points d is the same all around the control box.

$$d = \frac{N_P}{P_R} \tag{3.34}$$

where N_P is the number of points on the perimeter of the control box and P_R is the perimeter of the control box.

3.6.2 Forces evaluations

In order to get the drag coefficient we need the aerodynamic forces over the cylinder. The three terms of the 3.33 have been evaluated separately, in particular the first one it has been evaluated on each cell contained in the control box, the second and the third terms have been calculated for each control point on the surface and then added to the total force.

To calculate the first term we need to compute a time derivatives, that implies the necessity to know the values of the field at the previous time step.

$$\mathbf{F}' = \frac{\rho}{\Delta t}\Big[\sum_i \mathbf{u}_i^{n+1} \cdot V_i - \sum_i \mathbf{u}_i^n \cdot V_i\Big] \tag{3.35}$$

In two dimensions the volume of the cell is a surface evaluated by a function included in the manager of the grid PABLO [7], and the three components of the force are:

$$F'_X = \frac{\rho}{\Delta t}\Big[\sum_i u_i^{n+1} \cdot S_i - \sum_i u_i^n \cdot S_i\Big] \tag{3.36}$$

$$F'_Y = \frac{\rho}{\Delta t}\Big[\sum_i v_i^{n+1} \cdot S_i - \sum_i v_i^n \cdot S_i\Big] \tag{3.37}$$

$$F'_Z = \frac{\rho}{\Delta t}\Big[\sum_i w_i^{n+1} \cdot S_i - \sum_i w_i^n \cdot S_i\Big] \tag{3.38}$$

$$\tag{3.39}$$

The second term is the shear stress force on the surface of the control box and we use the same algorithm used to evaluate the force with the Lagrangian markers method.
The third term is:

$$\mathbf{F}'' = \sum_i \rho \mathbf{u}_i (\mathbf{n}_i \cdot \mathbf{u}_i) \cdot S_i \tag{3.40}$$

and the three components of the force are:

$$F''_X = \sum_i \rho(\mathbf{u_i} \cdot n_X) \cdot u_i d \tag{3.41}$$

$$F''_Y = \sum_i \rho(\mathbf{u_i} \cdot n_Y) \cdot v_i d \tag{3.42}$$

$$F''_Z = \sum_i \rho(\mathbf{u_i} \cdot n_Z) \cdot w_i d \tag{3.43}$$

$$\tag{3.44}$$

3.6.3 Results

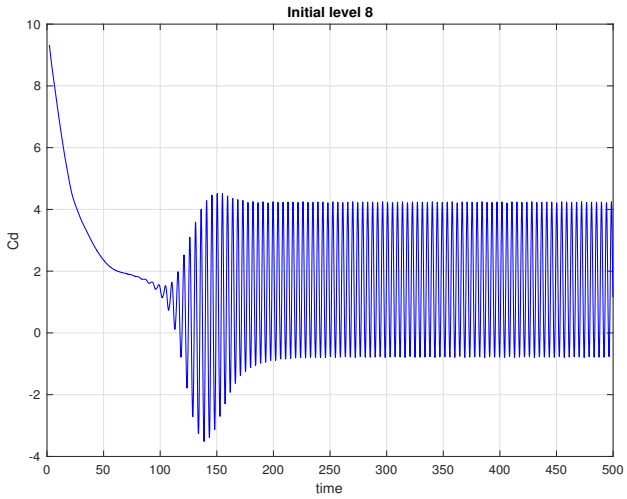
in table 3.11 are shown the values of the average drag coefficient obtained with the Noca method. Is it possible to see that to obtain significant results with this method, very refined grid are needed, way more refined than the grid used with the Lagrangian markers method. Furthermore the solution seems to not

be convergent for low refined grid. This limit imply the adoption of grids with millions of points and the simulations has a very high computational cost both in terms of resources and cpu time. Using a very refined grid (level 10) we have a percentile error of 1.43%, while using the Lagrangian markers method we found 3.82%.

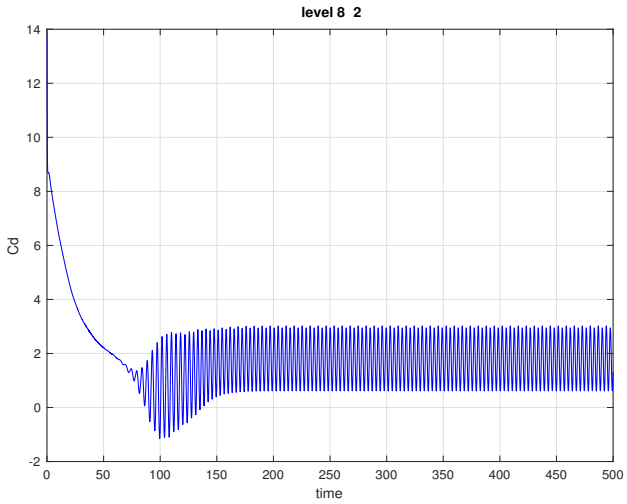
Table 3.11: Drag coefficient evaluated with the Noca method

Lin	Cd ave	Err %
6-1	1.6812	-25.90%
7-2	1.4334	-7.35%
8-0	1.6659	-24.76%
8-1	1.4389	-7.76%
9-0	1.5726	-17.77%
8-2	1.7775	-33.12%
9-1	1.518	-13.68%
10-0	1.3162	1.43%

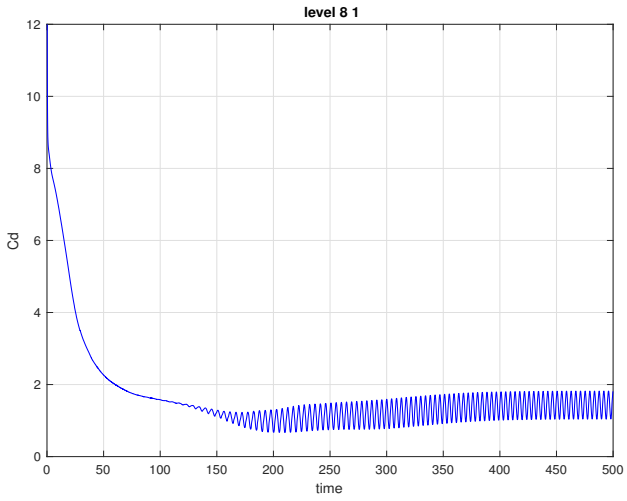
In figure 3.15 is shown the drag coefficient plotted with respect to the physical time. It is possible to note very big oscillations of the drag coefficient. This may be due to the numerical noise that affect this method as reported in [11], to investigate the source of this noise the three terms of the 3.33 have been plotted separately to locate which of these is the source of the noise (figure 3.16). Both the first therm and the shear stress have big oscillations but they compensate each other in amplitude. The problem could be caused by the characteristic frequencies of the oscillations. By performing a Fourier analysis we found out that the two terms in object present two frequencies of oscillations after the transient is ended.



(a) Cartesian grid Level 8



(b) Level 8 with two levels of refinement



(c) Level 8 with one levels of refinement

Figure 3.15: Temporal evolution of the drag coefficient evaluated with the Noca method for a circular refined grid.

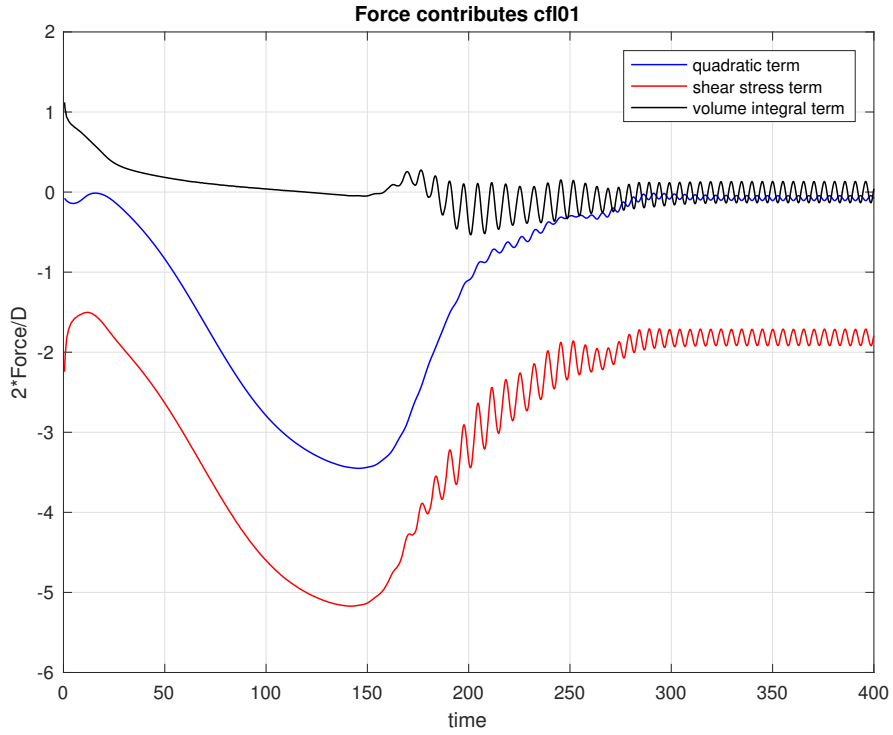


Figure 3.16: The three terms of the 3.33 have been plotted separately to found the source of the oscillations

We can conclude by saying that the Noca method gives More accurate results but it imply high computational costs, and the presence of the numerical noise. Also it is very difficult to define a control volume in three dimensions for an internal flow, so this method will not be used for the simulation of the flow inside the aorta.

3.7 Noca and Lagrangian markers comparison

The forces obtained with the Noca method, in terms of *Drag coefficient*, are compared with the results obtained previously using the improved Lagrangian markers method.

Table 3.12: Drag coefficients obtained with the two methods on L 9 and L 10 uniform cartesian grid

Grid	Cd ave	err %
L10 LM	1.318	1.30%
L10 Noca	1.3162	1.43%
L9 LM	1.3009	1.30%
L9 Noca	1.2756	17.77 %

From table 3.12 is it possible to notice that using a very refined grid at Level 10 the results are pretty much the same , the Noca method is a little more

accurate. For more coarse grid as Level 9 the error is very big using the Noca method. So the Noca methods needs the use of more refined grids to get acceptable results. While the Lagrangian Markers gives small error even on more coarse grid. On very refined grids The *Noca Method* is more accurate but the trend of the C_D in time is not smooth at all, it presents numerical noise that result in very big oscillation of the forces around the cylinder. Even if the results are slightly less accurate using the *Lagrangian markers method* we can get accetable results even using coarse grid and the trend of the C_D in time is very smooth.

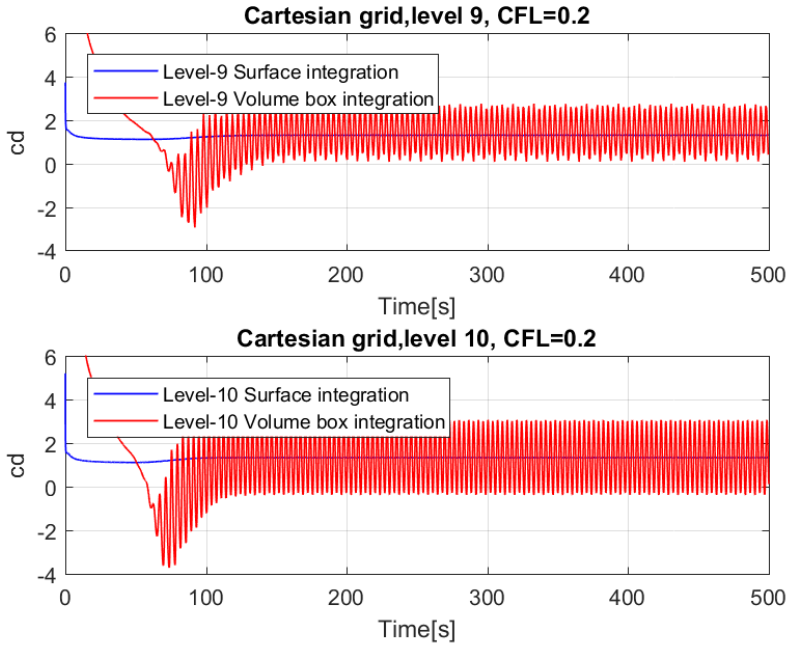


Figure 3.17: Comparison between the 2 methods used, the *Drag Coefficient* obtained using the Lagrangian method is smooth but the result are less accurate, the results obtained with the Noca method are more accurate but the numerical noise affecting the solution make difficult to use this method

Chapter 4

Validation of the code

Before using the method to calculate the force distribution acting on an aneurysm, the code has to be tested and validated. To do so, we will use our algorithm to simulate test case found in literature and the result obtained will be compared to the ones obtained in past works from the literature. In this section the forces will be evaluated using the Lagrangian marker method only.
The first 2D test case is the flow past a circular cylinder presented in the previous chapter. in order to moreover validate the code in 2D the second test case chosen is the so called Turek test presented in [12]. After that is presented how the code has been modified to be usable in 3D. To validate the code in three dimensions a flow past a rigid sphere is used as presented in [13]. If the solution obtained for this cases will match the result presented in literature, we will pass to our final test case: the internal flow inside an aneurysm of the aorta.

4.1 Turek test case

In this sectiion we will use our algorithm to evaluate the force around a 2D rigid body in a channel, the result will be compared to the test case presented in [12], The Turek test-case is a specific configuration developed to test and compare different numerical methods and different code implementations used for fluid-structure interaction problems.

4.1.1 Problem definition

The rigid body is composed by a circle with a rectangular bar attached on its back. The length of the channel is 2,5m, the eight is 0,41m. The origin is positioned in the left bottom corner of the channel. The center of the circle is positioned at {0.2,0.2} and it has radius 0,05m. The tail is 0.35m long with 0,02m height, its right bottom corner is positioned in $\{0,6,0,019\}$. All the geometric parameters are summarized in table 4.1.

Table 4.1: Geometric parameters

Parameters		value[m]
channel length	L	2.5
channel width	H	0.41
circle center	C	[0.2 , 0.2]
circle radius	r	0.05
tail length	l	0.35
tail thickness	h	0.02

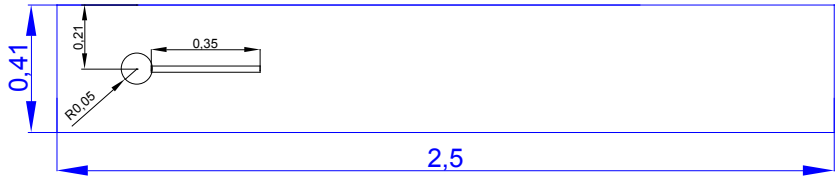


Figure 4.1: The geometry used for this test

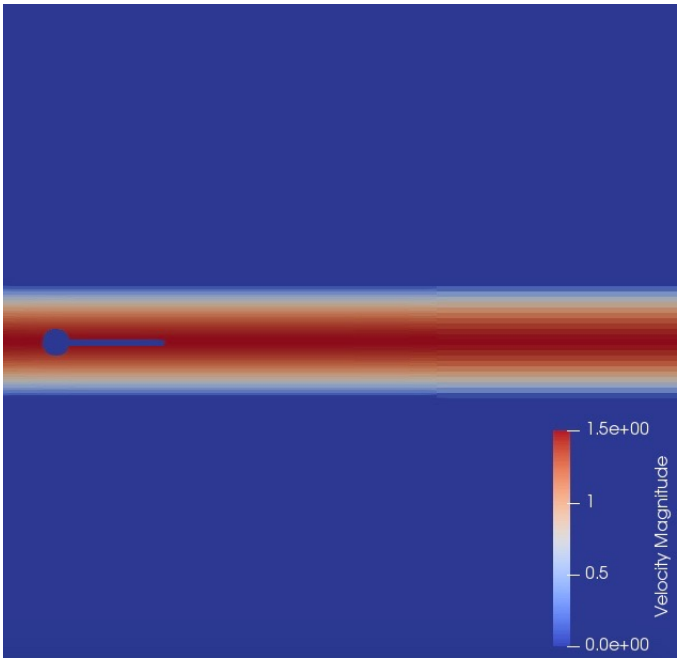


Figure 4.2: The 2D Turek test case, the body is not symmetric respect to the axis of the channel, that leads to the presence of a lift force.

Is it possible to notice that this configuration is not symmetric, the axis of the body does not coincide with the channel axis. This is made in order to get a lift different to zero besides the drag force. This allow us to prevent periodic oscillations that could effect the precision of the computation [12].

4.1.2 Boundary and initial conditions

The boundary condition imposed on the inflow of the channel is a parabolic profile for the velocity such that the average velocity i $\bar{U}$ and the maximum value of the inflow velocity is $1.5\bar{U}$ (figure 4.2).

$$\mathbf{u}_{(0,y)} = 1.5\bar{U}\frac{y(H-y)}{(\frac{H}{2})^2} = 1.5\bar{U}\frac{4y}{0.1681}(0.41-y) \tag{4.1}$$

For the outflow the [12] states that any boundary conditions can be chosen, for example the free-flow or stress-free condition. We chose to use a free-flow condition for the velocity and Dirichelet boundary conditions for the pressure. As we are in the incompressible case the value for the pressure can be chosen arbitrary [12]. We set the value to have a null mean value.
On the walls and on the body surface the *no-slip condition* is imposed.

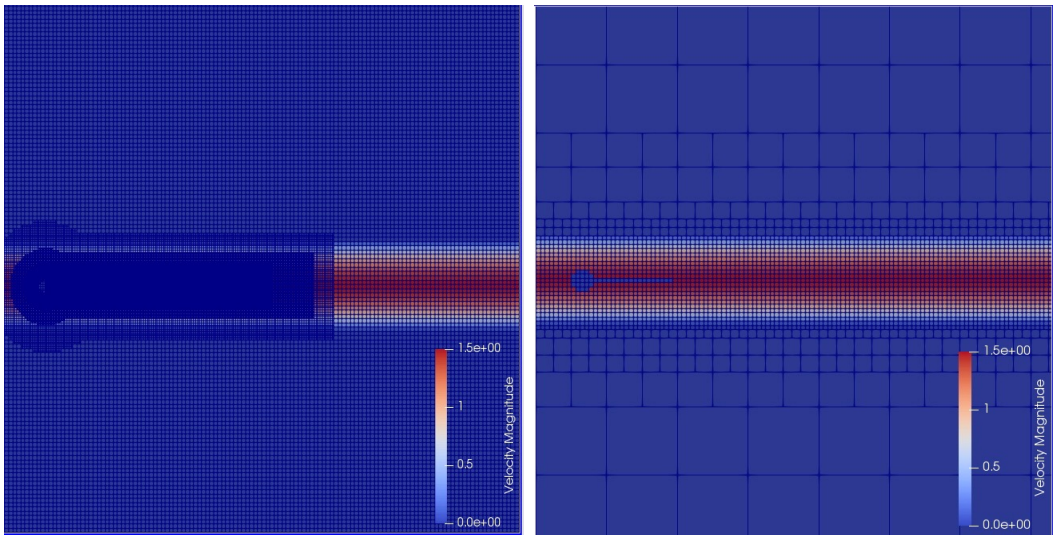
The initial condition on the velocity is set to be the parabolic profile (equation 4.1) used as boundary condition.

4.1.3 Results

We performed the simulation setting the parameters shown in table 4.2. We compare the result obtained using a quadtree grid with the ones obtained using a cartesian discretization inside the channel. Using the cartesian grid we expect to have a more accurate result but longer cpu time. Instead using a quadtree grid less computational power is required but the result are slightly less accurate.

Table 4.2: Fluid parameters

Dimensional Parameters	value
$\rho^f[10^3\frac{kg}{m^3}]$	1
$\nu^f[10^{-3}\frac{m^2}{s}]$	1
$U[\frac{m}{s}]$	1
non-Dimensional Parameters	.
$Re = \frac{Ud}{\nu^f}$	100
$\bar{U}$	1



(a) Quadtree grid. (b) Cartesian grid.

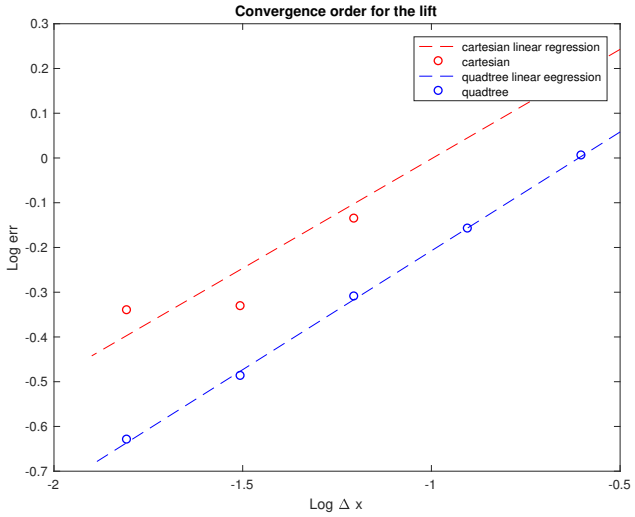
Figure 4.3: The two grid used to perform the simulation, in the first simulations (figure 4.3a) the grid is refined using the octree around the body, after that the result are compared to a uniform cartesian grid inside the channel (4.3b).

Table 4.3: Result obtained for lift and drag on the body.

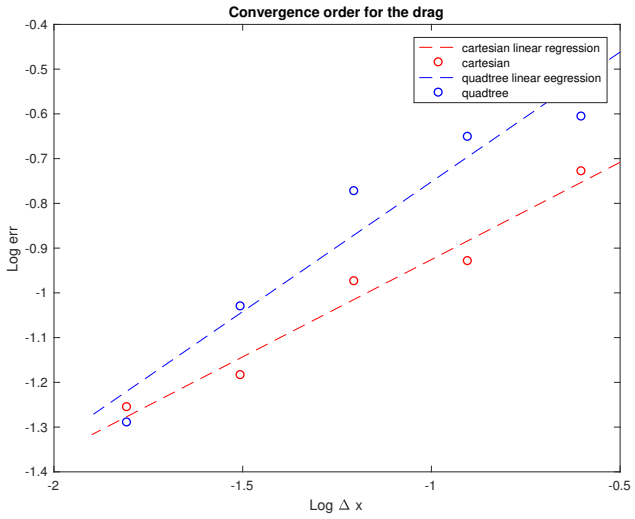
Grid	Lift	err %	Drag	err %
Quadtree grids				
L4-3	26.3406	150.15%	113.086	-17.27%
L5-3	22.1852	110.69%	116.773	-14.58%
L6-3	19.5035	85.22%	124.949	-8.60%
l7-3	17.3514	64.78%	136.287	-0.30%
L8-3	16.1598	53.46%	142.589	4.31%
L9-3	13.0895	24.31%	150.298	9.95%
Cartesian grids				
L7	-4.00158	-138.00%	116.642	-14.67%
L8	-3.31084	-131.44%	126.5621	-7.42%
L9	2.38659	-77.34%	128.2216	-6.20%
L10	4.74355	-54.95%	134.054	-1.94%
L11	4.8292	-54.14%	135.4788	-0.89%
L12	8.88562	-15.62%	143.44	4.93%
ref	10.53	0.00%	136.7	0.00%

In table 4.3 are presented the result obtained in term of lift and Drag. Is it possible to notice how the result obtained using a cartesian grid present more accurate value. For example using an octree grid at initial level 7 with 3 refinements the error on the lift is 64.78%, instead discretizing the channel with an uniform grid at level 10 the error is 54.14%.

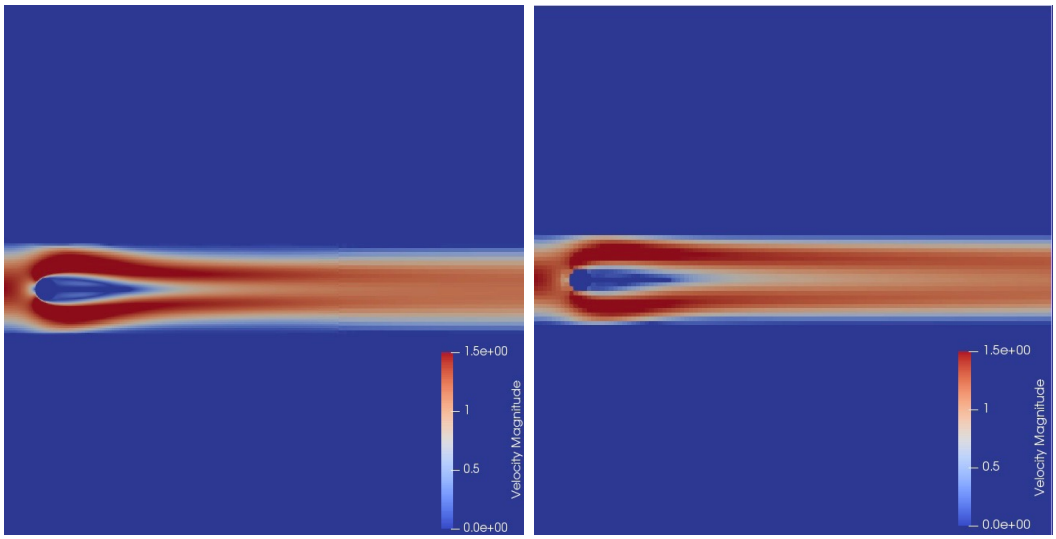
The Lagrangian marker method used to evaluate the force's components converge(figure ref) but is not consistent for the Drag, that means that refining the grid the error decrease but the solution tends to a value that is bigger than the one presented in [12]. Furthermore to get a solution with a reasonable error, very refined grids has to be used.



(a) Convergence order on the lift

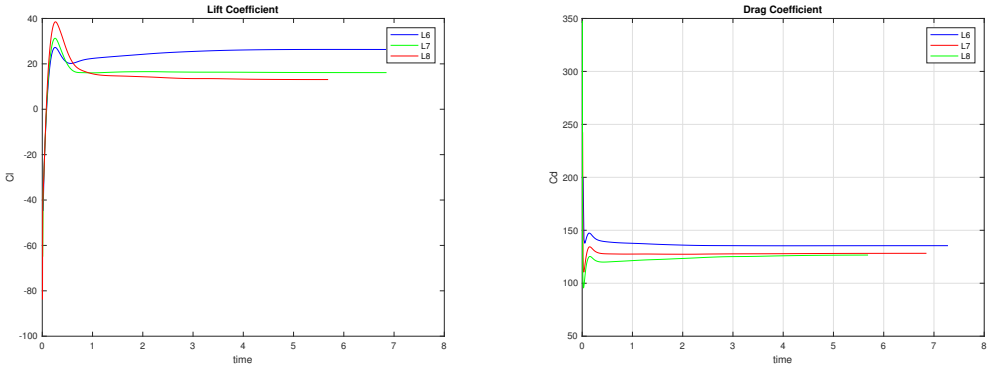


(b) Convergence order on the drag



(a) Result obtained with Quadtree grid (b) Result obtained with cartesian grid

Figure 4.5: The result in term of velocity field obtained using the quadtree grid 4.5a, and the cartesian grid into the channel 4.5b.



(a) The Lift obtained for the Turek test case at Reynolds100.

(b) The Drag obtained for the Turek test case at Reynolds100.

4.2 Extension to three dimensions

In three dimensions the components of the velocity will be $\mathbf{u} = \{\text{u, v, w}\}$, and the Force has three components as well $\mathbf{F} = \{F_X, F_Y, F_Z\}$ To extend the usability of the force evaluation to 3D cases the only part to be extended is the interpolation of gradient and pressure on the control points.
For the pressure there is no need to intervention, because the algorithm we use to perform the RBF interpolation is already in three dimensions. Although the Gradient interpolation we used in two dimensions only considers two components of the velocity, so we need to compute the least square interpolation considering the whole three components of the velocity.
The least square interpolation used to interpolate the gradient is presented in chapter 3 and is based on the principle to minimize the distance from the data to the interpolated value.

The gradient of the velocity will be a 3X3 matrix in the 3D case, and to get all its 9 components we will have to solve 3 linear system 3X3 instead of the two system we had in the 2D case.

$$\nabla \mathbf{u} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix}$$

The Distance is Defined as

$$I = \sum_i \frac{1}{2} \Big[\mathbf{u_{x_i}} - \mathbf{u_{x_0}} - \nabla \mathbf{U} \|_{x_0} (\mathbf{X_i} - \mathbf{X}) \Big]^2 \tag{4.2}$$

Where $\mathbf{X} = \{$ x y z$\}$. and $\mathbf{u_{x_i}}$ is the velocity in the neighbor cells and $\mathbf{u_{x_0}}$ is the penalized velocity To minimize this distance we impose the derivative equal to zero. In the 3D case I has three components $I = \{I_u \ I_v \ I_w\}$, that leads to three equation systems.

$$\frac{\partial I_u}{\partial \nabla_x u} = 0 \tag{4.3}$$
$$\frac{\partial I_u}{\partial \nabla_y u} = 0 \tag{4.4}$$
$$\frac{\partial I_u}{\partial \nabla_z u} = 0 \tag{4.5}$$
$$\tag{4.6}$$

$$\frac{\partial I_v}{\partial \nabla_x v} = 0 \tag{4.7}$$
$$\frac{\partial I_v}{\partial \nabla_y v} = 0 \tag{4.8}$$
$$\frac{\partial I_v}{\partial \nabla_z v} = 0 \tag{4.9}$$
$$\tag{4.10}$$

$$\frac{\partial I_w}{\partial \nabla_x w} = 0 \tag{4.11}$$
$$\frac{\partial I_w}{\partial \nabla_y w} = 0 \tag{4.12}$$
$$\frac{\partial I_w}{\partial \nabla_z w} = 0 \tag{4.13}$$
$$\tag{4.14}$$

Where $\nabla_x = \frac{\partial}{\partial x}$, $\nabla_y = \frac{\partial}{\partial y}$ and $\nabla_z = \frac{\partial}{\partial z}$, From this we can get the equation systems to solve in order to evaluate the gradient.

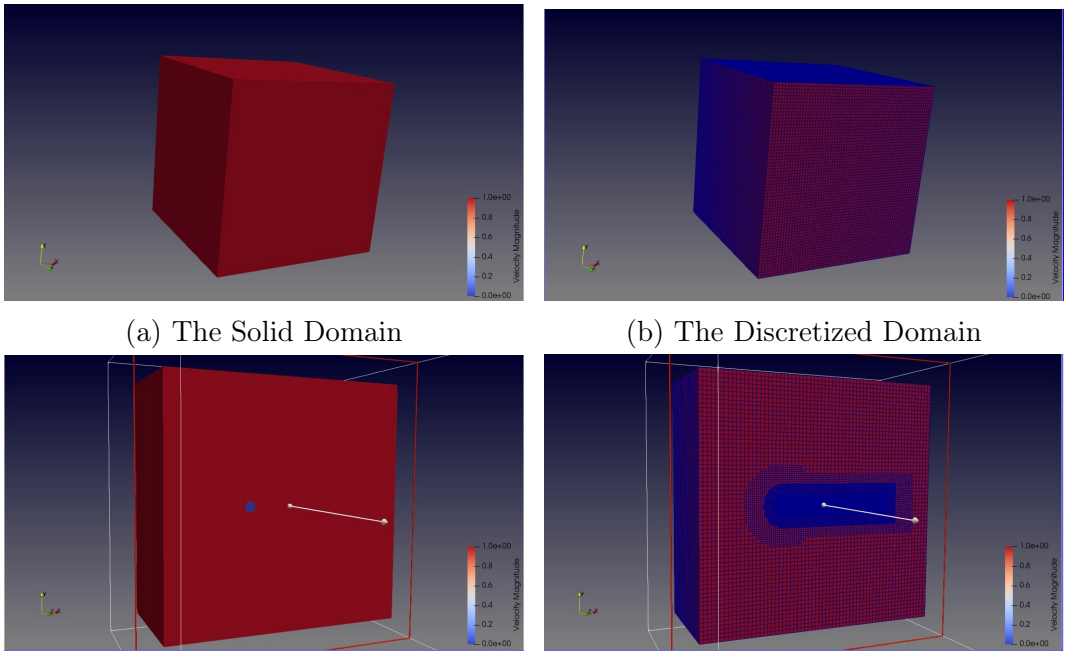
$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) & \sum_i (x_i - x)(z_i - z) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 & \sum_i (y_i - y)(z_i - z) \\ \sum_i (x_i - x)(z_i - z) & \sum_i (y_i - y)(z_i - z) & \sum_i (z_i - z)^2 \end{bmatrix} \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \end{Bmatrix} = \begin{Bmatrix} \sum_i (x_i - x)\Delta u_i \\ \sum_i (y_i - y)\Delta u_i \\ \sum_i (z_i - z)\Delta u_i \end{Bmatrix}$$

$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) & \sum_i (x_i - x)(z_i - z) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 & \sum_i (y_i - y)(z_i - z) \\ \sum_i (x_i - x)(z_i - z) & \sum_i (y_i - y)(z_i - z) & \sum_i (z_i - z)^2 \end{bmatrix} \begin{Bmatrix} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial v}{\partial z} \end{Bmatrix} = \begin{Bmatrix} \sum_i (x_i - x)\Delta v_i \\ \sum_i (y_i - y)\Delta v_i \\ \sum_i (z_i - z)\Delta v_i \end{Bmatrix}$$

$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) & \sum_i (x_i - x)(z_i - z) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 & \sum_i (y_i - y)(z_i - z) \\ \sum_i (x_i - x)(z_i - z) & \sum_i (y_i - y)(z_i - z) & \sum_i (z_i - z)^2 \end{bmatrix} \begin{Bmatrix} \frac{\partial w}{\partial x} \\ \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial z} \end{Bmatrix} = \begin{Bmatrix} \sum_i (x_i - x)\Delta w_i \\ \sum_i (y_i - y)\Delta w_i \\ \sum_i (z_i - z)\Delta w_i \end{Bmatrix}$$

Is it possible to notice that in this case also, the matrix A in all the three systems is identical, that leads to a mole efficient computation as we need to invert one single matrix instead of three to solve the systems and find the gradient of the velocity. Using this method leads to a more efficient way to evaluate the interpolated gradient. Previous approach was to evaluate the approximate value of the gradient on the neighbor's cell centers first, and then interpolate the value of the gradient in the desired point, this brings a first truncation error summed to the interpolation error. With the least square interpolation we can get the interpolated value of the gradient in any point knowing only the velocity values in the neighbor's cell centers, the only error affecting this method is the interpolation one. So this method is more accurate.

4.3 Sphere Re 500



(a) The Solid Domain (b) The Discretized Domain

(c) Section of the Domain (d) Section of the discretized Domain

Figure 4.7: The sphere test case used to validate the code.

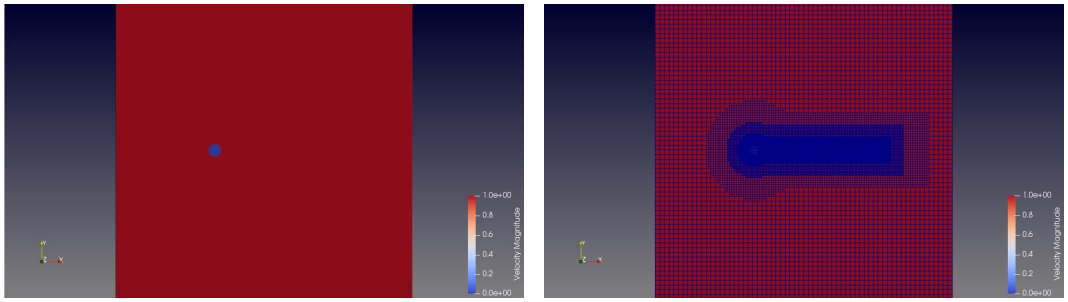
To validate the code in 3D we chose to use the test case of a fluid flow over a rigid sphere at Reynolds 500, this because we have several examples in literature of computational and experimental analysis of the incompressible flow past a sphere. Even if this test case is symmetric and simple, the structures that can be observed at moderate and high Reynolds Numbers can be complex. This two has the main Reasons why this test case is considerate to be a benchmark to validate Navier-Stokes equation solvers.
In [13] Is presented a numerical solution of the flow over a sphere at moderate Reynolds Numbers using an immersed boundary method. We will compare the Drag Coefficient obtained with the result from [13] and other works.

4.3.1 Problem Definition

The test case chosen to validate the data is a unit diameter sphere at Reynolds 500.
The forces will be evaluated in the control points distributed on the surface then integrated on the surface of the sphere. Unit diameter sphere is placed with his center in [-4, 0, 0] the density and pressure are set to 1. we will use a 0.8 CFL condition for the time evolution.
The grid used to discretize the domain is a combination of the circular and rectangle refinement seen before in this work. That allow to have both the advantages of the 2 grid. The force evaluation will be accurate on the surface of the body and the velocity field in the wake is enough refined to represent all the vortex structures.

Table 4.4: Geometric parameters

Dimension domain	16x16
Diameter	1
x center	-4
y center	0
z center	0
ρ	1
Re	500
CFL	0.8



(a) The initial condition of the test case in plane

(b) The Discretization of the grid used to validate the code

Figure 4.8: A plane view of the Domain's section

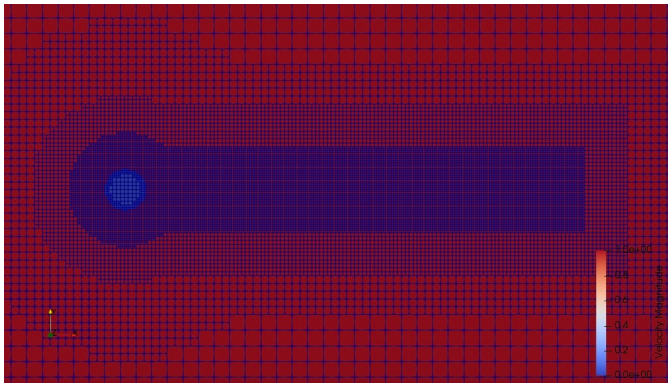


Figure 4.9: zoom of the discretized domain around the sphere

On the inflow of the domain the boundary condition are Neumann type for the pressure and Dirichlet on the velocity equal to 1 parallel to the x axis . On the outflow a free flow condition is enforced. On the upper and lower boundaries of the domain we use the same boundary condition used for the inflow.

4.3.2 Results

The force is evaluated using the Lagrangian marker method previously presented and the Drag coefficient is defined:

$$C_D = \frac{F_x}{1/2\rho u^2 S}$$

The simuation is stopped after 500 seconds of physical time, to be sure the flow is completely developed and the transient is ended. In table 4.5 are presented the result for the force around the sphere in terms of drag coefficient. Is it possible to notice that using quite coarse grids the errors are very big, that means we need to use very refined grids in order to find accettable results. This presumably come from the first order upwind scheme used to discretize the convection equation. It is planned to improve the code switching to a second order upwind scheme. The Use of an adaptive grid based on the gradient criteria could bring to more accurate result even using more coarse grids saving computational time.

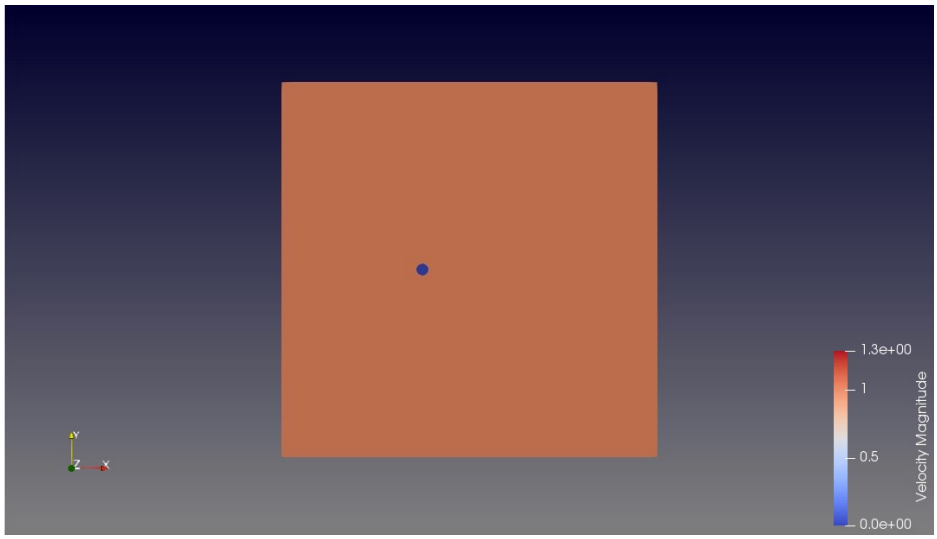
Table 4.5: Result obtained using an octree grid

Grid	Cd	err %
L6-3	0.9832	46.32%
L7-3	0.8532	33.32%
L8-3	0.6945	17.45%
L9-3	0.6034	8.34%
Ref	0.52	0.00%

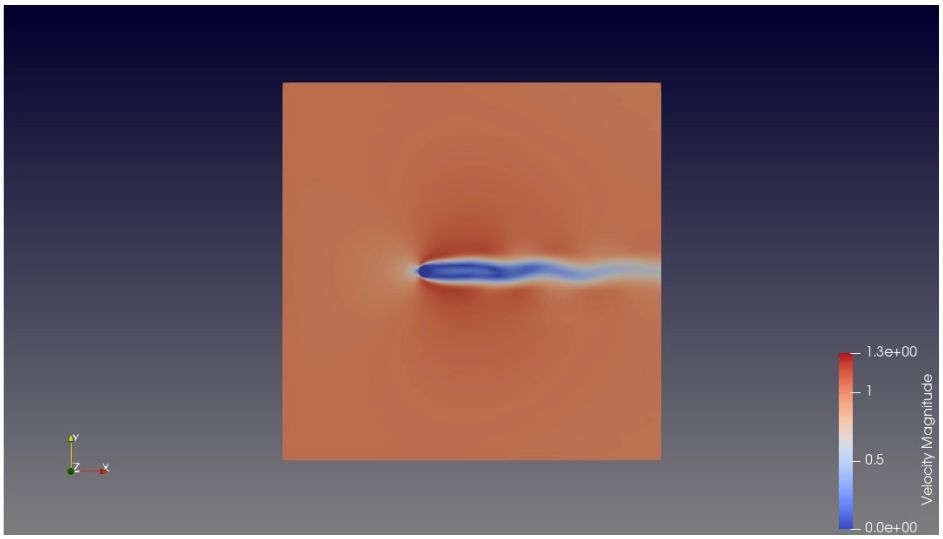
In table 4.6, the Drag coefficient evaluated using the most refined grid, is compared with the result obtained in previous works from literature. As is it possible to notice, different works use different method to evaluate the force or different schemes to solve pressure and velocity fields. This leads to have slightly different result in terms of drag coefficient. To have a more complete ensemble overview we will compare our result with several result obtained from the literature.

Table 4.6: Comparison of the result with other works from literature.

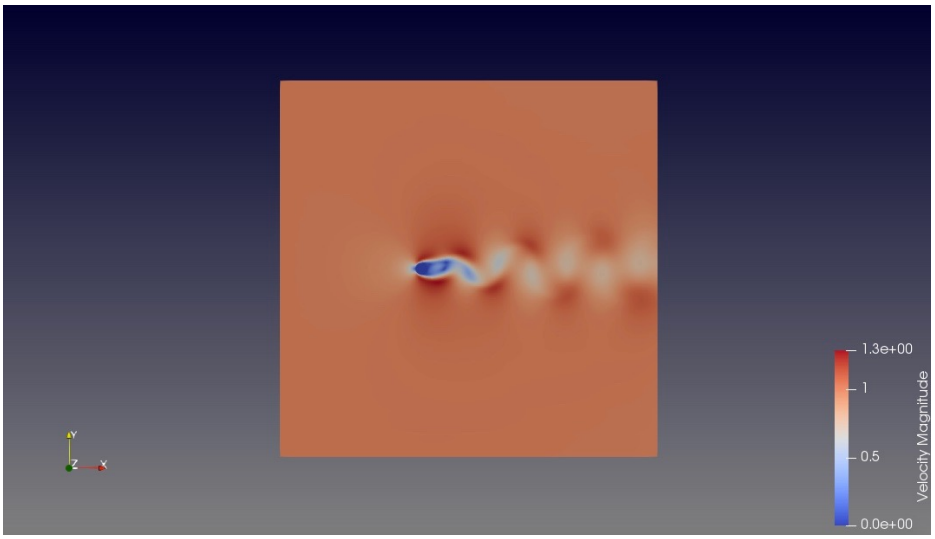
Reference	Cd
Campregher et al.(2009)	0.520
Fornberg (1988)	0.4818
Fadlun et al.(2000)	0.476
This work	0.6034



(a) Time=0s



(b) Time=20s



(c) Time=500s

Figure 4.10: The velocity field obtained for the sphere at $Re = 500$ at three different physical time.

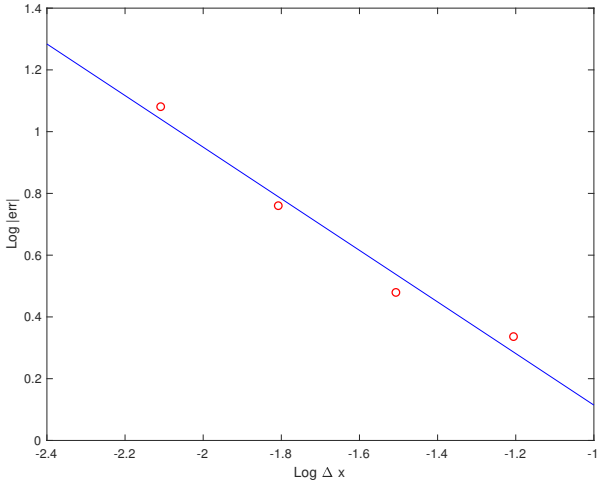


Figure 4.11: The Convergence order for the Drag coefficient evaluated for the sphere at Reynolds 500.

4.3.3 Convergence Order

As we use a first order accurate upwind scheme to discretize the convection equation, we expect to find a convergence order at most one for the force evaluation. We use a second order interpolation for the pressure and a firs order scheme for the gradients. There is no point in using higher orders since the velocity and pressure fields are first order accurate.

Table 4.7: Data used for the convergence analysis.

Grid	C_D	$\|err\|$	Δx	$Log\|err\|$	$Log\Delta x$
L6-3	0.9832	0.46	0.0625	-0.3342314	-1.20412
L7-3	0.8532	0.33	0.03125	-0.477295	-1.50515
L8-3	0.6945	0.17	0.015625	-0.7582046	-1.80618
L9-3	0.6034	0.0834	0.0078125	-1.0788339	-2.10721

To be rigorous we should calculate the absolute error using the exact value, but we don't have an exact solution so we should use the result obtained with the most refined grid. But this value is not an analytic solution so we chose to use the reference value from [13].
The linear regression line has equation:

$$Log\,|err| = 0,8354 \cdot Log\,|\Delta x| + 0,721$$

According to what presented in section 3.4 the convergence order is $p = 0,84 \approx 1$, that means that the error is nearly double using double size cell as explained in chapter 3.

Chapter 5

Force evaluation on the Aortic Aneurysm

Figure 5.1: The 3D image of the aorta obtained by overlaying several 2D images produced with computed axial tomography as described in section 5.1.2.

An aortic aneurysm is an enlargement (dilation) of the aorta to greater than 1.5 times normal size. They are most commonly located in the abdominal aorta, but can also be located in the thoracic aorta. Untreated, aneurysms tend to become progressively larger, although the rate of enlargement is unpredictable for any individual. Aortic aneurysms cause weakness in the wall of the aorta and increase the risk of aortic rupture. When rupture occurs, massive internal bleeding results and, unless treated immediately, shock and death can occur. Rarely, clotted blood which lines most aortic aneurysms can break off and

result in an embolus.The risk of rupture of an Aortic abdominal aneurysm is related to its diameter; once the aneurysm reaches about 5 cm, the yearly risk of rupture may exceed the risks of surgical repair for an average-risk patient. Rupture risk is also related to shape; so-called "fusiform" (long) aneurysms are considered less rupture prone than "saccular" (shorter, bulbous) aneurysms, the latter having more wall tension in a particular location in the aneurysm wall. Aortic aneurysms resulted in about 152,000 deaths worldwide in 2013, up from 100,000 in 1990 [14].

5.1 Problem definition

In this section we will present the test case used to perform some preliminary simulation using the force evaluation method presented previously, to evaluate the force distribution over the surface of the aorta.
The 3D model of the aorta affected by an aneurysm is immersed inside a fluid cube of dimension 30. The test is performed at $Re = 200$. To guarantee the stability of the numerical scheme a CFL condition 0.6 is imposed. The pressure and velocity fields will be evaluated on the fluid grid and the result are used to evaluate the Forces on the control points on the surface of the aorta.

$$\sum_i \mathbf{F}_i = \Big(\sum_i p_i + \sum_i \nabla \mathbf{u}_i\Big) \cdot \sum_i \Delta S_i \cdot \mathbf{n} \tag{5.1}$$

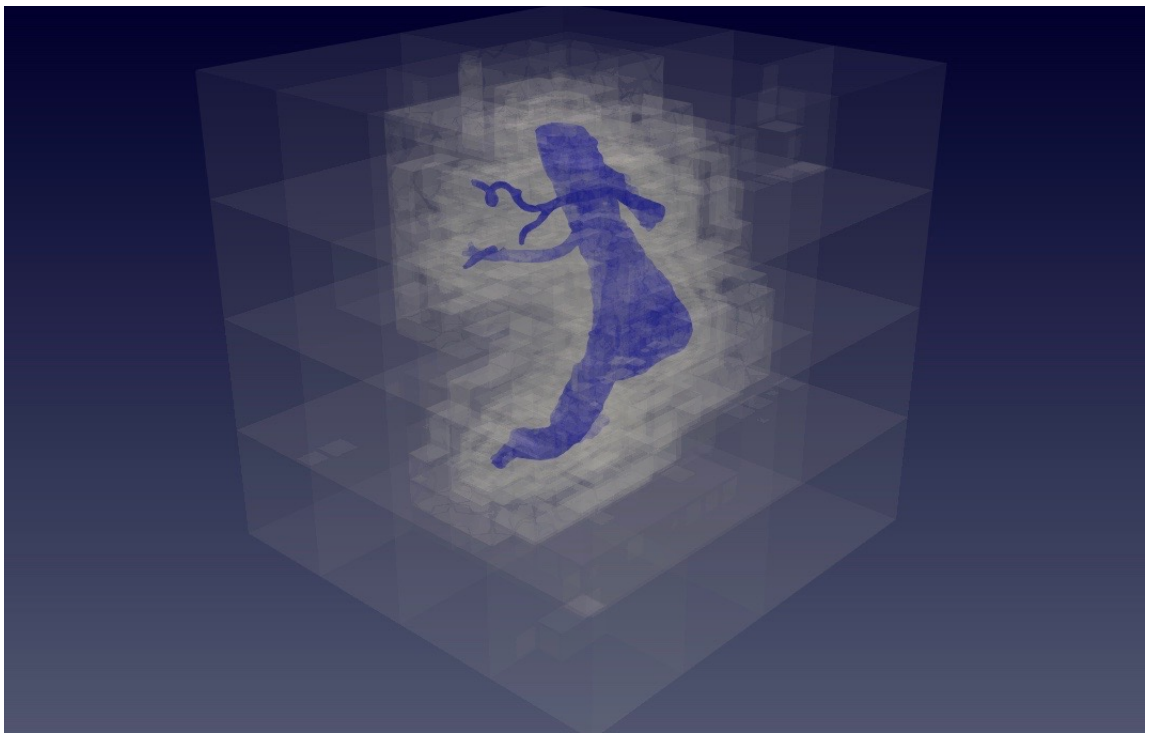


Figure 5.2: The fluid domain is a cube and the 3D model of the aorta is placed inside the cube.

5.1.1 Initial and Boundary conditions

The initial condition imposed consist in placing a semi-sphere at the inlet of the aorta (figure 5.3) where the velocity is defined $\mathbf{u} = [0\ 0\ -1]$.

$$\mathbf{u} = \begin{cases} [0\ 0\ -1]\ \forall \mathbf{X} \in semi - sphere \\ [0\ 0\ 0]\ \forall \mathbf{X} \notin semi - sphere \end{cases} \tag{5.2}$$

For the boundary conditions for the preliminary simulation a free flow conditions are imposed at the outflow ends of the aorta.In the reality the aorta is not immersed in the fluid but is connected to the cardiovascular system. To best simulate the reality, the boundary conditions model will be improved with the implementation of the so called Windkessel model that consist in using the equations of the electrical schemes to simulate the velocity and pressure of the cardiovascular system. On the boundaries of the cube Dirichelet type boundary conditions are imposed on the velocity and Neumann type on the pressure.

$$\begin{cases} \mathbf{u} = 0\ \forall \mathbf{X} \in \partial\Omega \\ \nabla p = 0\ \forall \mathbf{X} \in \partial\Omega \end{cases} \tag{5.3}$$

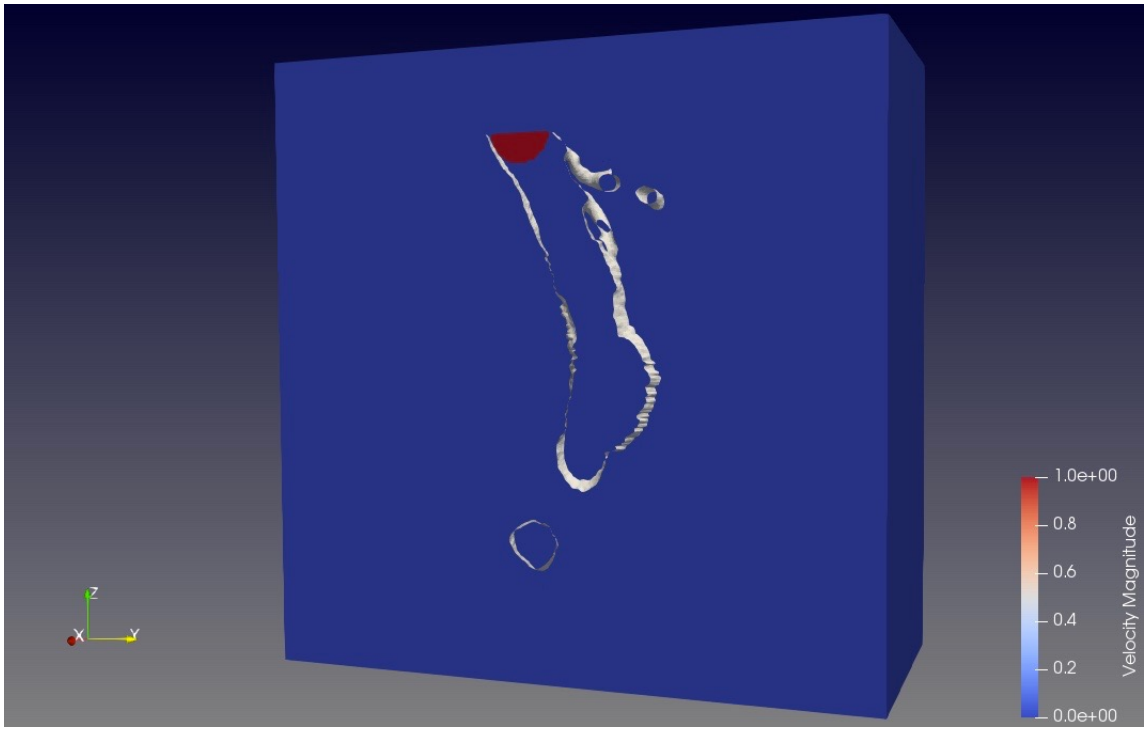


Figure 5.3: The initial condition used is a semi-sphere in which the velocity is defined one

5.1.2 Construction of the 3D model

The 3D image of the aorta, shown in figure 5.1 it is build through the use of the Computerized axial Tomography CAT. The Image is a three dimensional model of the structure within the body created by a computer that takes

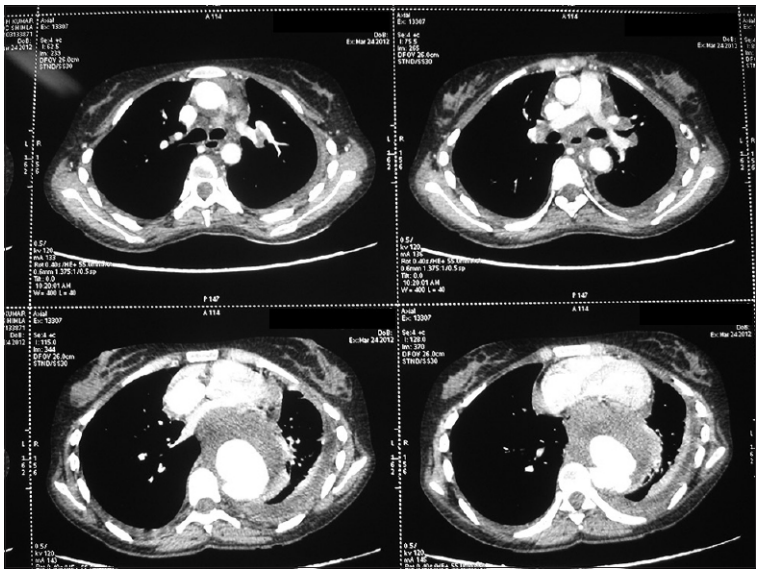
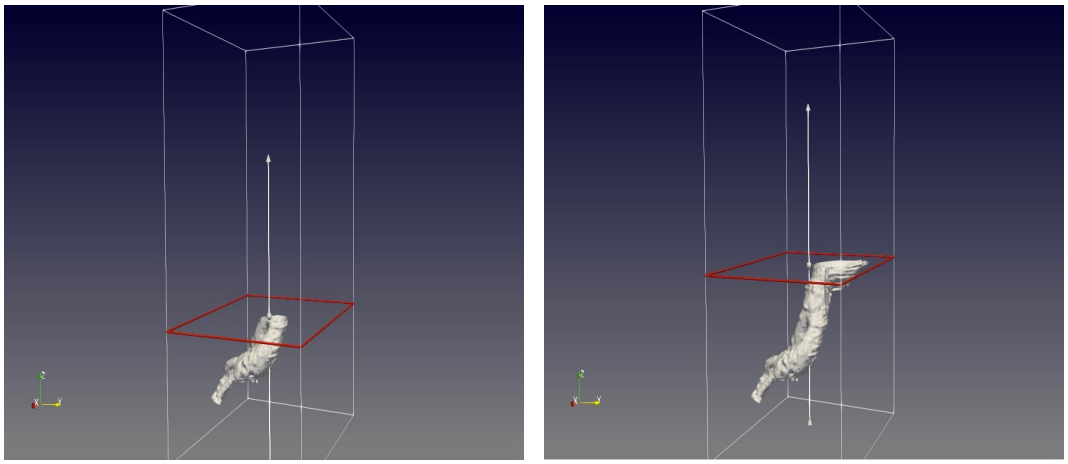


Figure 5.4: In this image four two-dimensional "cuts" of the abdominal CAT are presented

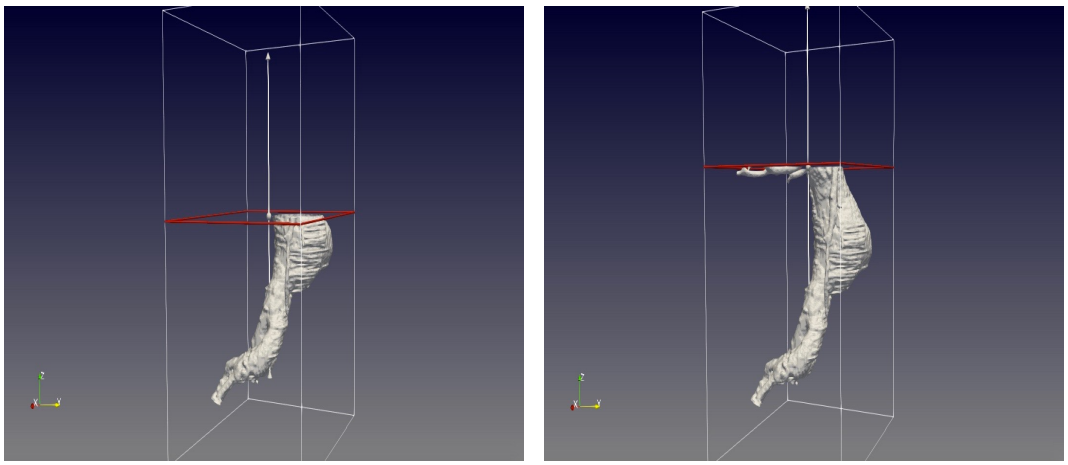
the data from multiple X-ray images and turns them in pictures. The CAT scan can reveal some soft-tissue and other structures that cannot be seen in conventional X-rays. Using the same dosage of radiation of an ordinary X-ray, an entire slice of the body can be made visible with about 100 times more clarity with the CAT scan. Each scan result in a 2D section of the desired structure. The "cuts"(tomograms) are usually made some mm apart. The CAT machine rotates 180 degrees around the patient's body; hence, why is called "axial." The machine produce a slim X-ray beam at 160 different points. Some Crystals positioned at the opposite points of the beam pick up and record the absorption rates of the varying thicknesses of tissue and bone. The data are then relayed to a computer that turns the information into a 2-dimensional cross-sectional image. In order to obtain a three-dimensional model of the desired object, the multiple cuts are than overlaid one on top of the other to have a 3D image of the structure.

5.1.3 Grid

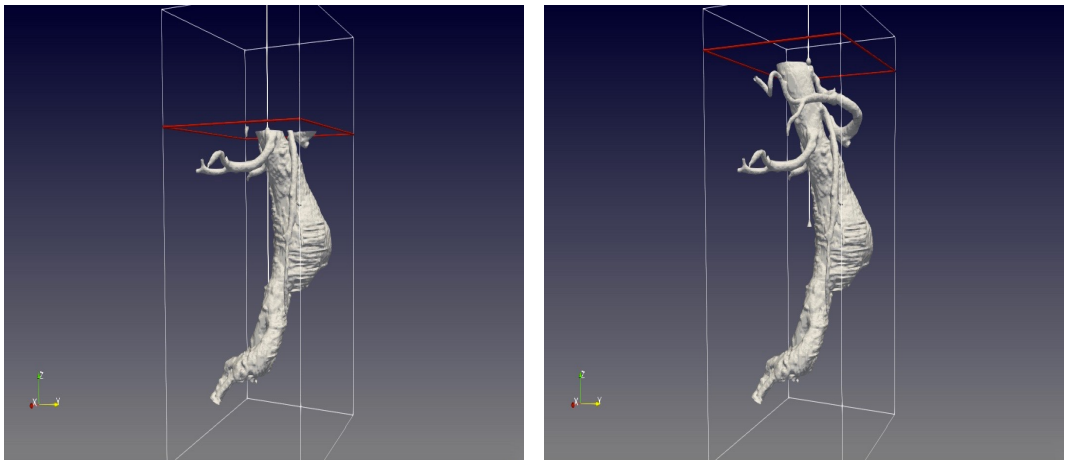
For the fluid domain an octree discretization inside the cube is used. As presented before this kind of grid allows to have a very high precision where needed while it reduce the computational cost in terms of memory and computational time. The forces are evaluated on the points of the surface using the values of pressure and velocity of the nearest cells, as seen in chapter 3. To have an high accuracy for the force evaluation we need a very refined grid near the surface while far from the body the gradients are very small and we can accept a more coarse grid. The fluid domain is discretized at initial level is 5 with three levels of refinement near the surface of the body and inside the aorta as the flux we are interested in is an internal flow.
In figure 5.7a is reported a section of the fluid domain where is it possible to see the grid that is very refined close to the surface and is more coarse as we move far from the body. Contrary to what done untill now, the cells marked



(a) (b)



(c) (d)



(e) (f)

Figure 5.5: A rapresentation of the building of the 3D image of the aorta. It is obtained by overlaying several two-dimensional images of the section of the aorta.

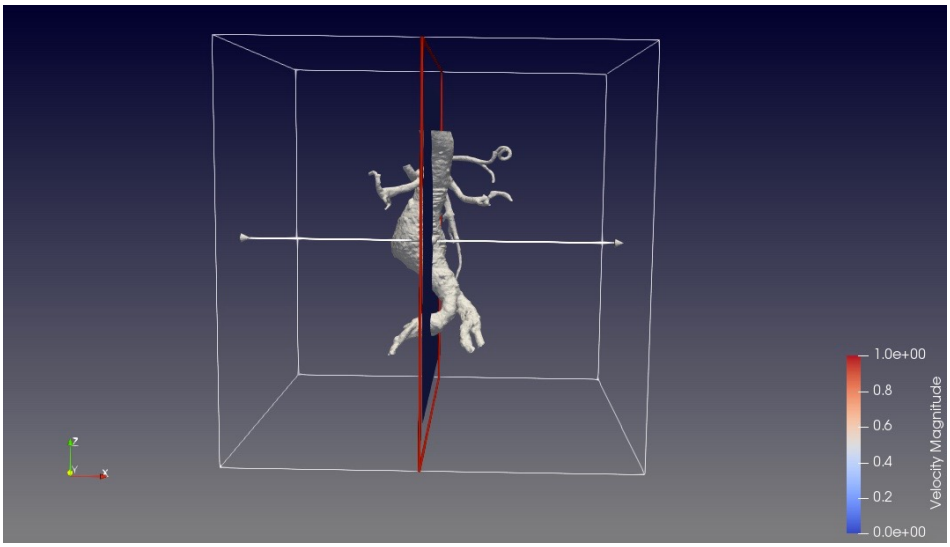


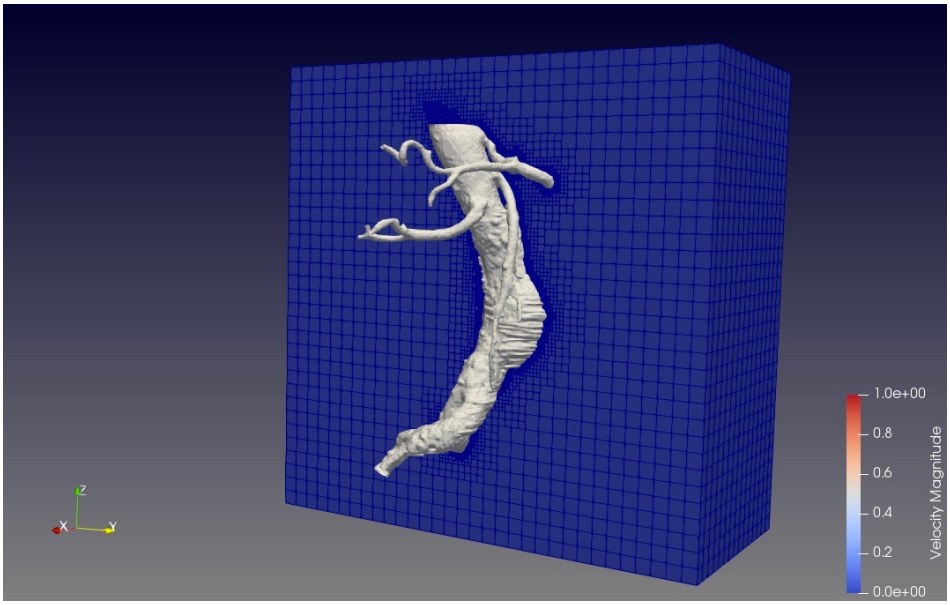
Figure 5.6: To give a better view of the aorta, a section of the fluid domain is reported, the plane used to define the section is the red square in figure.

to refine are not imposed by the user, but the discretization is evaluated automatically by BitPit, the library that manages the grid. The grid is static, does not change in time so the adaptivity of the grid is not used for this preliminary simulations.
For what concern the surface of the aorta, it is discretized using triangles as is it possible to see in figure 5.8. The problem of evaluating the force on the triangle's vertex is to evaluate the normal vector and the surface of the cell. As done before in order to calculate the cell area and the outward normal to the cell the Lagrangian marker approach is used, to find the points on which the forces will be evaluated. In the three-dimensional case the Lagrangian markers are the vertex of the triangle. Cell area and outward normal are easy to evaluate on the surface of the triangle and the force can be evaluated on the center of the triangle, that it is our control point, as seen in section 3.1.1.

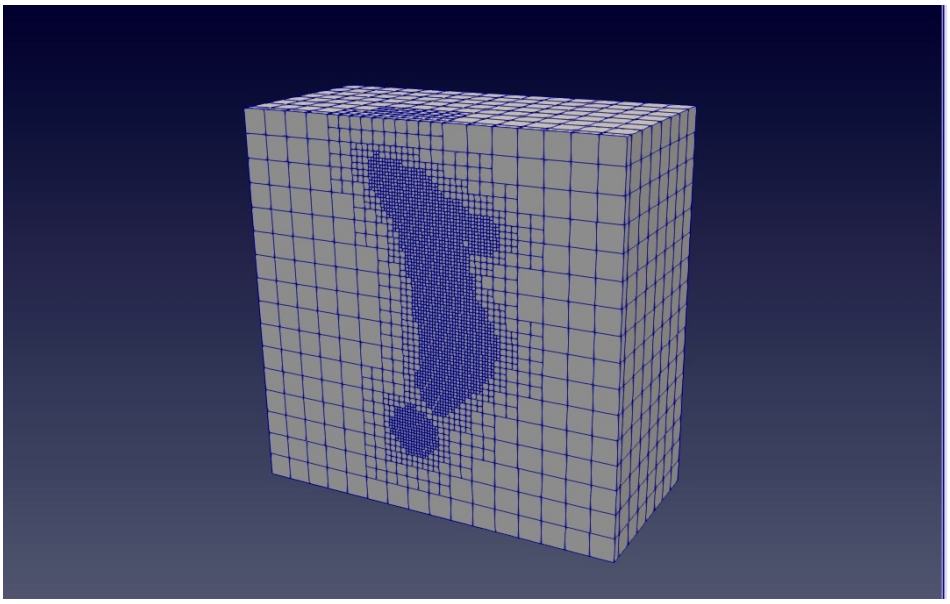
5.2 Force evaluation inside concavities.

As seen in chapter three the interpolation of the gradient using a least square approach present a constraint in terms of number of neighbors needed. The code it has been used, until now, to simulate the flow over convex geometries or more in general on very simple geometries. For the test cases already presented, for each control point, it has always been possible to find an appropriate number of neighbors for the interpolations. There was no evidence of any problem even in presence of concave surface.
But the geometry of the aorta is much more complex and presents concavities with a very small curvature radius. For all the control points inside the concavities there are not enough fluid neighbors to interpolate the gradient and the forces can't be evaluated.

To overcome this problem the value of the force on this "ill" points it is interpolated using all the near control point where the force has been evaluated.



(a)



(b)

Figure 5.7: 5.7a The fluid domain space discretization. 5.7b The discretization of the fluid domain inside the aorta

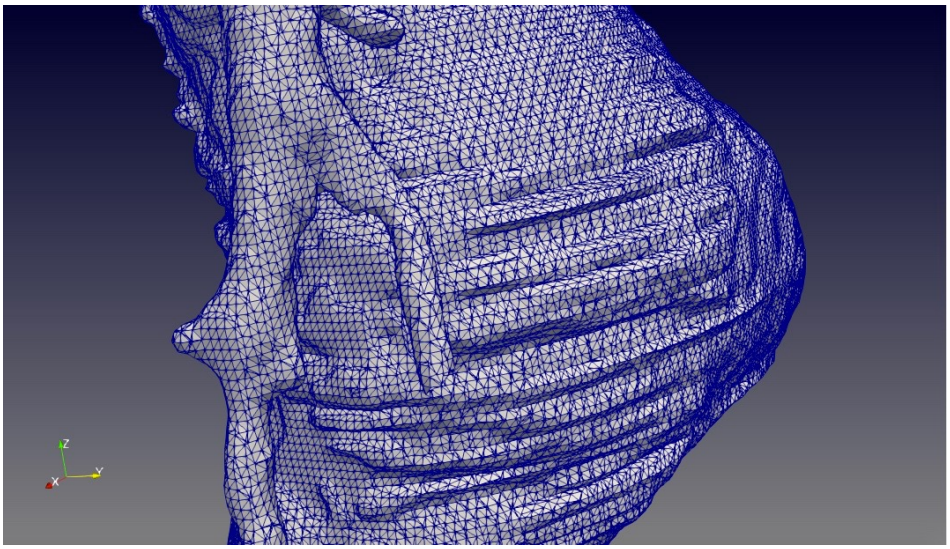


Figure 5.8: A detail of the discretization of the aorta surface.

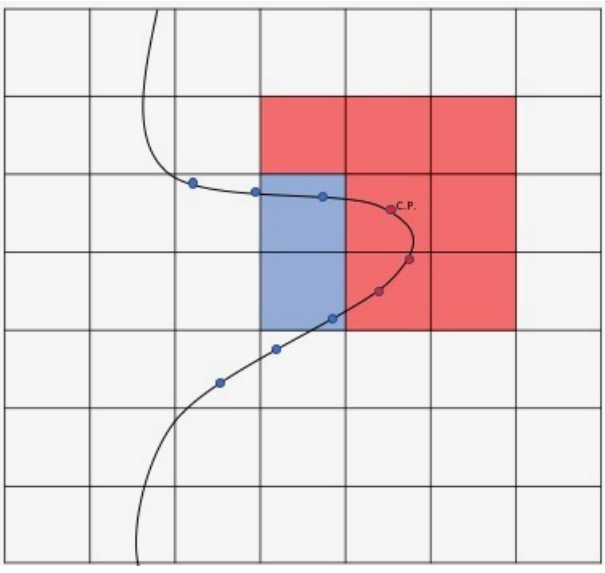


Figure 5.9: Seeing it from the two-dimensinal point of view, the control point in figure has only two fluid neighbors and the Velocity gradiet cant be evaluated. In red the "ill" points, in blue the control points where the gradient can be evaluated normally.

The idea is not to evaluate the force on the "ill" control point. but to express it as a function of the force on the control points where the force can be evaluated. To interpolate the force on the "ill" control point we use the *Radial Basis Function* approach already used to interpolate the pressure.

5.3 Force Distribution

The aneurysms tend to become progressively larger. This means there is a concentration of the stress inside it, this forces continue to enlarge the aneurism until an aortic rupture occur. In this phase the body is rigid, but from the result there should be visible the force concentration inside the aneurism.

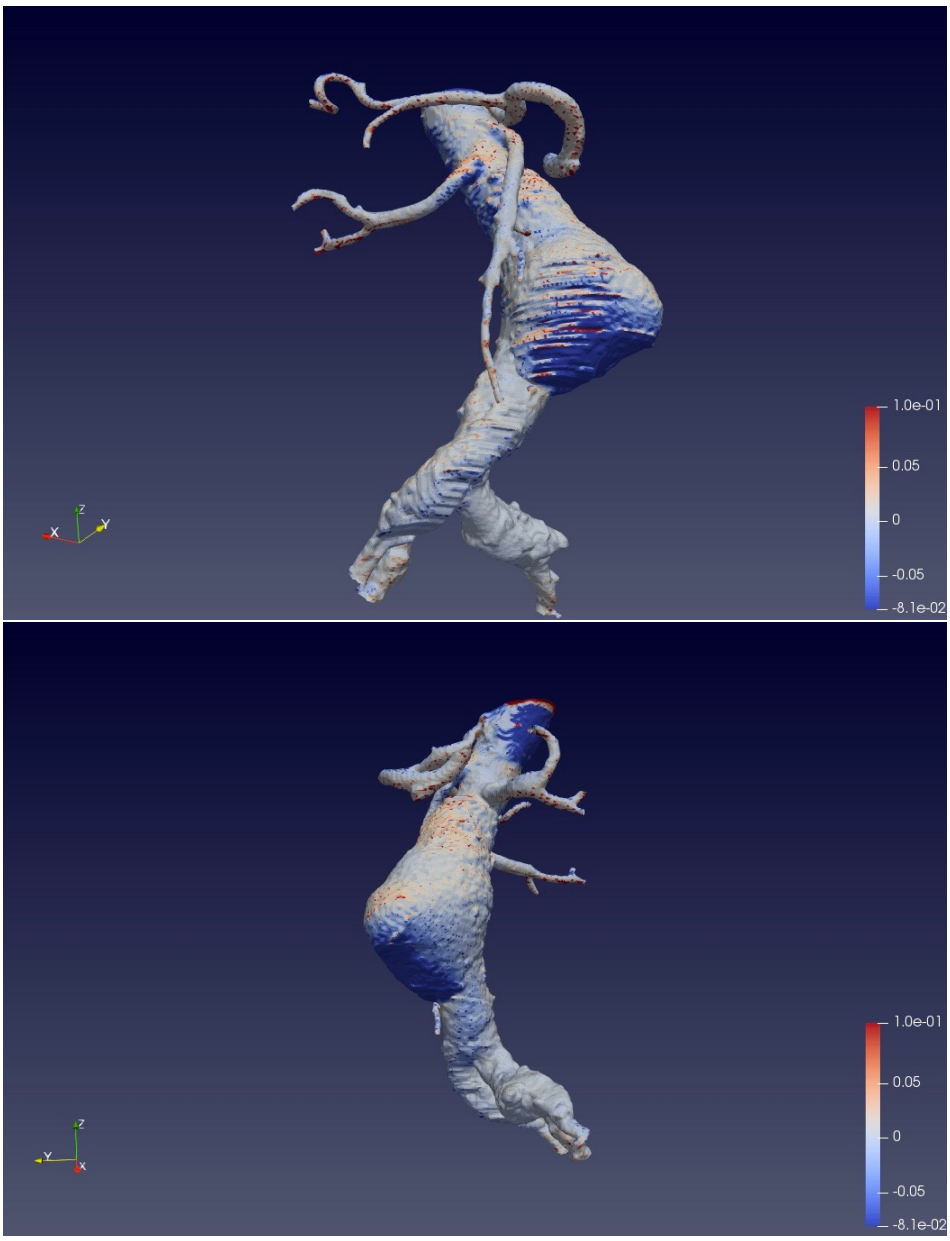


Figure 5.10: From the force distribution over the surface of the aorta is it possible to see a concentration of the stress over the aneurism.

From image 5.10 is it possible to see the concentration of the forces inside the aneurysm, but there is a second zone where the force are concentrate, this is due to the boundary condition used. The semi-sphere is facing downward the z axis, and it is not aligned with the axis of the inflow section of the aorta. This model is not completely realistic, The implementation of the Windkessel model for the boundary conditions will impose at the inflow and outflow a model that can give a realistic modeling of the cardiovascular system.

5.4 Future Developments

As we have seen, even if the scheme presented to evaluate the fluid dynamics forces seems to work fine on a complex geometry, many problem has been underlined during the validation of the code.
The errors obtained when the first order upwind scheme is used are relatively high. This is due to the poor accuracy order of the scheme.The first step to improve the evaluation of the forces is to implement a **second order upwind** scheme to discretize the advection equation.

To have a realistic simulation of the flux inside the aorta we need appropriate boundary conditions at the inlet and outlets of the structures.This can be possible implementing in the code the **winkessel model** for the boundary conditions. This allows to model the cardio vascular system using the principle and the formulations of the electrical circuits.

Till this phase the immersed body is considered to be rigid.In order to develop a more realistic model and to predict the evolution of the aneurysm after his formation, a **structural elastic deformation** model of the body has to be implemented in a way similar to the penalization model, used to define the rigid body. The elastic model allow to **Predict the rate of enlargement** and when the aortic rupture occur.

This work will provide the basis for future research on a more precise prediction of rupture risk.

Ringraziamenti

Vorrei ringraziare il prof. Domenic D'ambrosio, relatore di questa tesi di laurea. Il prof. Angelo Iollo e l'Ing. Florian Bernard, dell'INRIA per avermi dato un'opportunità di crescita personale unica lavorando a questo progetto in Francia, oltre che per l'aiuto fornitomi e la grande conoscenza che questa esperienza mi ha donato, per la disponibilità e precisione dimostratemi durante tutto il periodo di lavoro a Bordeaux. Vorrei ringraziare Luigi con cui ho condiviso la crescita personale e professionale che questa esperienza ci ha dato. Ringrazio tutti i colleghi che ho incontrato durante il mio tirocinio presso l'INRIA per la loro fantastica collaborazione. Mi avete sostenuto e siete sempre stati pronti ad aiutarmi.

Il più grande ringraziamento va a mia madre Adriana e mio padre Herman che, con il loro dolce e instancabile sostegno, mi hanno permesso di arrivare fin qui davanti a voi oggi, contribuendo alla mia formazione personale. A mia sorella Paola per aver sempre creduto in me in qualsiasi circostanza. A Giovanni, Roberto e Riccardo, ci siamo sempre sostenuti a vicenda, nella buona e nella cattiva sorte, sia durante le fatiche e lo sconforto che hanno caratterizzato il nostro percorso, sia nei momenti di gioia e soddisfazione al raggiungimento del traguardo. Ringrazio Gabriele Gerardo e tutti i miei amici, che hanno avuto un peso determinante nel conseguimento di questo risultato, punto di arrivo e contemporaneamente di partenza della mia vita. Grazie per aver condiviso con me in questi anni le esperienze più importanti, vi voglio bene.

List of Figures

List of Tables

Bibliography

[1] Kaufmann, T.A.S., Graefe, R., Hormes, M., Schmitz-Rode, T. and Steinseiferand, U., "Computational Fluid Dynamics in Biomedical Engineering", Computational Fluid Dynamics: Theory, Analysis and Applications , pp. 109-136

[2] D. D'Ambrosio, Computational Fluid Dynamics, Lectures.

[3] P.Angot, C.Bruneau, P.Fabrie, A penalization method to take into account obstacles in a incompressible flow, Numer.Math. 81 (4) (1999) 497-520.

[4] M.Bergman, J.Hovnanian, A.Iollo, An accurate cartesian method for incompressible flows with Moving Boundaries.

[5] M.Bergman, A. Iollo, Modelling and simulation of fish-like swimming, Journal of computational Physics 230, 329-348 (2011).

[6] A.Raeli. M. Bergmann, A. Iollo, A finite-difference method for the variable coefficient Poisson equation on hierarchical cartesian mesh, Journal of computational Physics 355, 59-77 (2018)

[7] http://www.optimad.it/products/bitpit/.

[8] Guy G. Morton, A computer oriented geodetic data base and a new technique in file sequencing, International Business Machines Company,New York, 1966.

[9] B.N.Rajani, A. Kandasami, S.Majumdar, Numerical simulation of laminar flow past a circular cylinder, Applied Mathematical and Computational Science, 33 (2009) 1228-1247.

[10] F.Hou, C.Huang, J. Lu, A multi-dimensional data storage using quadtree and z-ordering, 2nd international Conference on computer science and electronic engineering (ICCSEE 2013).

[11] N.Nangia, H.Johansen N. Patankar, A. Bhalla, A moving control volume approach to computing hydrodynamic forces and torques on immersed bodies, Journal of computational Phisics 347 (2017) 437-462.

[12] S.Turek and J.Hron, Proposal for numerical benchmark of fluid-structure interaction between an elastic object and laminar incompressible flow, Institute for applied mathematics and numerics, University of Dortmund.

[13] R.Campregher et al, Computations of the flow past a still sphere at moderate Reynolds numbers using an immersed Boundary method. Journal of the Brazilian Society of Mechanical Sciences and Engineering 2009, vol. 31, n. 4.

[14] $https://en.wikipedia.org/wiki/Aortic_aneurysm$

[15] $http://www.lungindia.com$

Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), UniversitÈ de Bordeaux, Bordeaux INP and Conseil RÈgional d'Aquitaine (see https://www.plafrim.fr/).