Politecnico di Torino

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

Tesi di laurea Magistrale

Two-dimensional Aorta's outflow boundary condition computed using 2-elements Windkessel model.

Candidato
Luigi Maddaluno
Matricola 232996

Relatore
Prof. Domenic D' Ambrosio

Correlatore
Ing. Florian Bernard

Dicembre 2018

Abstract

Computational fluid dynamics (CFD) is an engineering field for analysing fluid flow, heat transfer, and associated phenomena, using computer-based simulation. CFD is a widely adopted methodology for solving complex problems in many modern engineering fields. Industries like Aerospace, automotive, chemical, power and bio-medical rely heavily on CFD for design, analysis and even for service & maintenance of Mechatronic systems. The purpose of this thesis is to evaluate hemodynamic of the Aorta through computational fluid dynamic analysis. In the following chapters we will appreciate a bio-medical application as the blood flowing in the Aorta.

As known *Navier-Stokes* equations need boundary condition to be solved hence the numerical value to impose is unknown a priori but appropriate BC can be found via Windkessel model. In particular a *2-elements Windkessel* model has been implemented and imposed on the outflow sides of the numerical domain. Such model is quiet common in this kind of application for biomedical engineering problem, this provides a relationship between pressure and velocity in a vessel, considering the physiological heart pumping phenomena and vessel elasticity. To discretize ODE's equation that make up the previous model, both Eulerian and multi-step method (second order Rounge-Kutta, Heun) have been applied .

Future applications can be a natural three-dimensional extension of the domain and application to real test case using vessel reconstruction starting from heart scan such as computed tomography (CT) and magnetic resonance (MR) imaging or CAT technique.

Abstract

La fluidodinamica computazionale (CFD) è una branca ingegneristica che analizza i moti fluidi, scambi termici ed i fenomeni associati, utilizzando simulazioni implementate al calcolatore. La metodlogia CFD è ampiamente utilizzata per risolvere complessi problemi in molte discipline ingegneristiche. Industrie come quella aerospaziale, automotive, chimica e biomedica fanno affidamento alla CFD per la progettazione, l'analisi e per servizi e manutenzione dei sistemi meccatronici. L'obbiettivo della presente tesi è quello di valutare lo stato emodinamico dell'Aorta attraverso, appunto, la fluidodinamica computazionale ,dunque, seguenti capitoli sarà mostrata un' applicazione biomedica della CFD: il flusso sanguigno in aorta.

Come già noto, le equazioni di *Navier-Stokes* necessitano delle condizioni al contorno per poter essere risolte, e dato che il valore da imporre non è noto a priori è necessario utilizzare un modello Windkessel. In particolare un *2-elements Windkessel* è stato innanzitutto implementato e successivamente imposto sul bordo del dominio nelle sezioni di uscita del vaso sanguigno. Tale modello risulta abbastanza diffuso per questo tipo di applicazoni per problemi biomedici, infatti, esso mette in relazione la pressione e la velocità del fluido all'interno del vaso tenendo in conto l'aspetto fisiologico di pompaggio dell'organo cardiacoe dell'elasticità della vena stessa. Per la discretizzazione matematica delle equazioni che compongono il modello appena illustrato ci si è affidato sia al metodo Euleriano che ad un modello multi-step (Runge-Kutta del secondo ordine, Heun).

Una futura applicazione del presente lavoro potrebbe essere una naturale estensione al caso tridimensionale e l'utilizzo di esso a Test Case realistici utilizzando tecniche di ricostruzione dei vasi sanguigni partendo da immagini derivanti da TAC, come ad esempio le immagini derivanti dalla Tomografia (CT) e dalle Risonanze Magnetiche (MR).

Contents

Introduction

For this thesis project I worked as intern at **INRIA** (*L'Institut national de recherche en informatique et en automatique*) . The whole project was based on a code wrote by Ing. Florian Bernard and Ing. Roman Leguay (for a Start-up project) that, defined a domain, solve Navier-Stokes equations with properly boundary condition using a *prediction-projection-correction* method.

As a first approach to the *C++* code several simulation has been carried out regarding the flow paste a simple circular cylinder varying different flow and grid parameters and comparing the force results with the available value from literature. To achieve a deeper code knowledge the section of the code that provide the force computation, via surface discretization, has been modified in order to improve the value accuracy.

The next task to be accomplished has been the aorta's outflow boundary condition computed using a **Windkessel** model.
Such model is quiet common in this kind of application for biomedical engineering problem, this gives a relationship between pressure and velocity in a vessel, considering the physiological hearth pumping phenomena.
Hence, in order to impose the *Dirichelet* condition on the outflow boundaries for the pressure p, the computed velocity after the prediction step (V^*), is stored in a *map*. Maps, in *C++* language are associative containers that store elements composed by a combination of a key value and a mapped value following a specific order given by the user. Obviously only the cells on the boundary were interesting for our purpose, so a mechanism of *if-else* has been built before storing the velocity. Such values are used from the model to relate the velocity to the pressure following the numerical solution of the ODE's that the model is composed of.

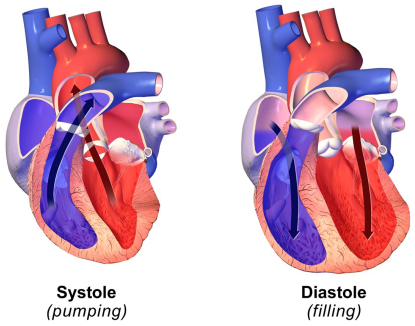
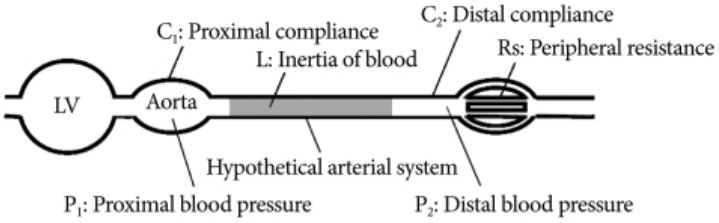
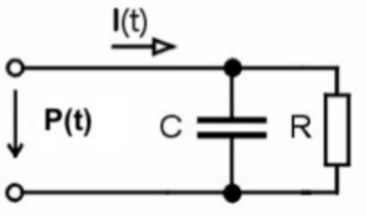


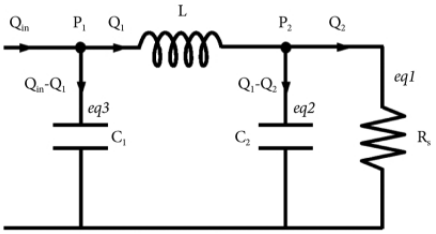
Figure 1: Systole and diastole from
[12]

The model used is a *2-Elements* Windkessel and uses the electrical analog as done in [10]. The two elements that the model is composed of, simulates the arterial compliance represented as a capacitor (C in $cm^3/mmHg$) with electric charge storage properties, and the resistance due the arterial system (R





(a) Analog: the 2 elements electrical model



(b) Analog: a more complex electrical model from [13]

in $mmHg \cdot s/cm^3$) represented by a classical resistor. The blood flow can be seen as the current flowing in the circuit ($I(t)$ in cm^3/s) and the pressure can be seen as a time-varying electric potential ($P(t)$ in $mmHg$). A more complex Windkessel model could be used, introducing several resistance or capacitor as represented in fig 2b improving similarity to a real case but the general idea about the mechanism doesn't changes.

To discretize ODE's equation that make up the previous model, both Eulerian and multi-step method (Heun or modifier Euler) have been applied. The computed value of the pressure for each boundary cell has been inserted in the *right hand side* of the Projection step as boundary condition, and the Navier-Stokes algorithm goes on.

All the simulations I used to run have been performed by PlaFRIM (Plateforme Federative pour la Recherche en Informatique et Mathematiques). PlaFRIM is a scientific instrument (Cluster) designed to support experiment-driven research in all areas of applied mathematics related to modelling and high performance computing.

Moreover, a general overview about future developments are provided. A spontaneous development of the present work can be a natural three-dimensional extension of the domain and application to real test case using vessel reconstruction starting from heart scan such as computed tomography (CT) and magnetic resonance (MR) imaging.

Chapter 1

Navier Stokes

1.1 Introduction to Navier-Stokes

In order to describe flow behaviour and it's macroscopic and microscopic characteristics it is necessary to introduce fluid mechanics equation. Fluid statics or hydrostatics is the branch of fluid mechanics that studies fluids at rest and is contrasted with fluid dynamics, 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 large so that it still 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 (both due originally to Leohnard 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**: is a way of looking at fluid motion that 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:

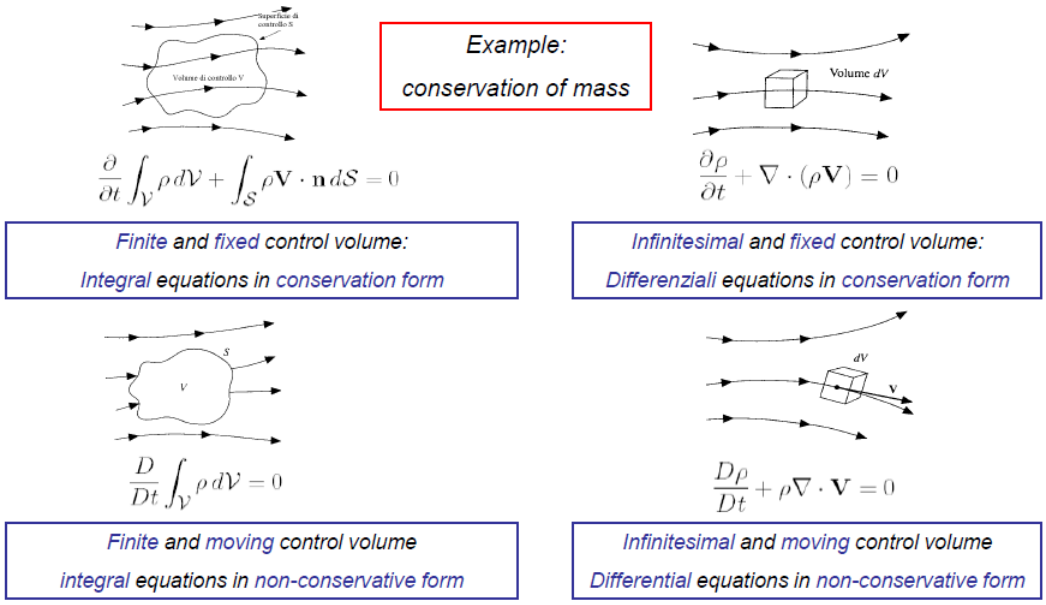


Figure 1.1: Different control volume examples

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 [1] 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 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

- Let's define a Cartesian frame of reference x, y, z, where density and velocity are function of space and time t.
- Let's consider an infinitesimal volume element placed in a generic point x, y, z, those dimensions are dx, dy, dz.
- Let's consider the mass flux across the six surfaces of the volume, then we evaluate the nett flux difference along x axis, so the flux across the

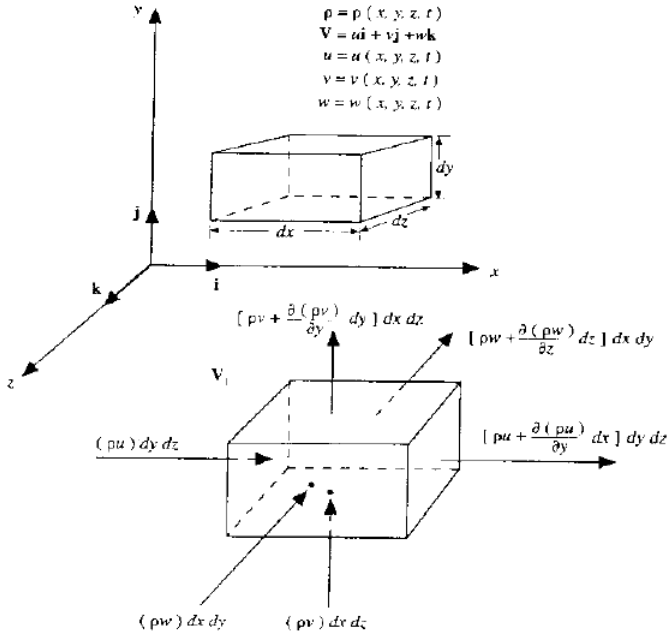


Figure 1.2: Volume forces acting on a control volume

left surface and the right one, we obtain the gradient of the mass along x direction.

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

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* compare the expression 1.4 and 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

$$\int_V \text{div}\,\underline{F} dV = \int_S \underline{F} \cdot d\underline{A} \tag{1.10}$$

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.

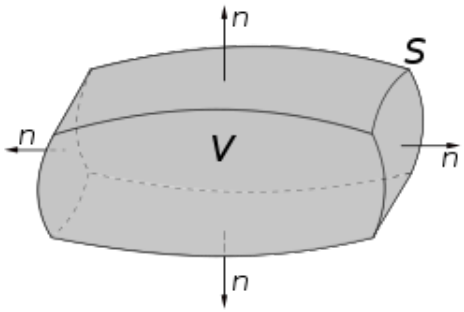


Figure 1.3: Volume example for Gauss theorem

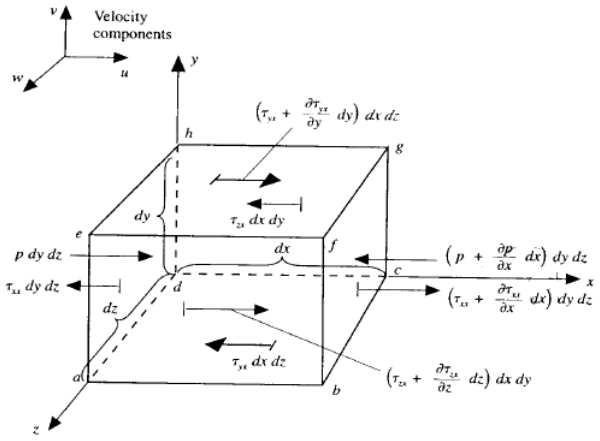


Figure 1.4: Forces and shear stress acting on a control volume

1.1.2 Momentum balance equation

Using the second Newton's law on the 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 to the only x components of the vectorial equation

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

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

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

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

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

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

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.

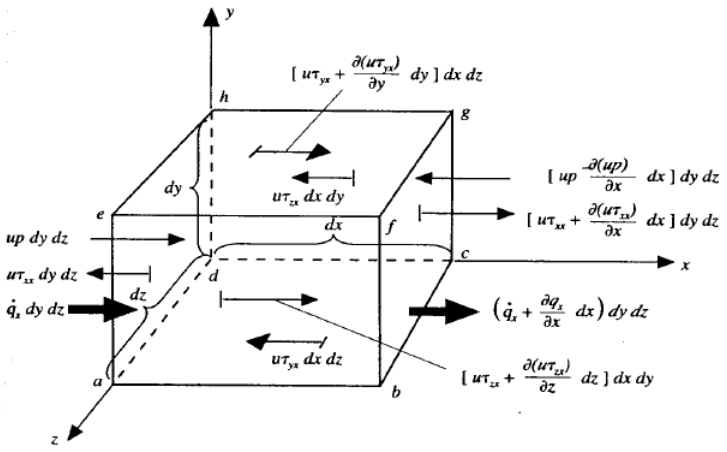


Figure 1.5: The net heat flux due to thermal conduction that flows along x axis

We have although 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.

1.2 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.

1.3 Incompressible flow hypothesis

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 vectorn notation in a two-dimension configuration is possible to obeserve, as done in [1], 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.
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:

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

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

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

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 [2] 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.

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

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

In [2] 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 $K\to 0$.

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

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in } \Omega \qquad (1.29)$$

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 [3] 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 neighbour.

this is needed to evaluate a more accurate value of the gradient in the zone near the immersed boundary.

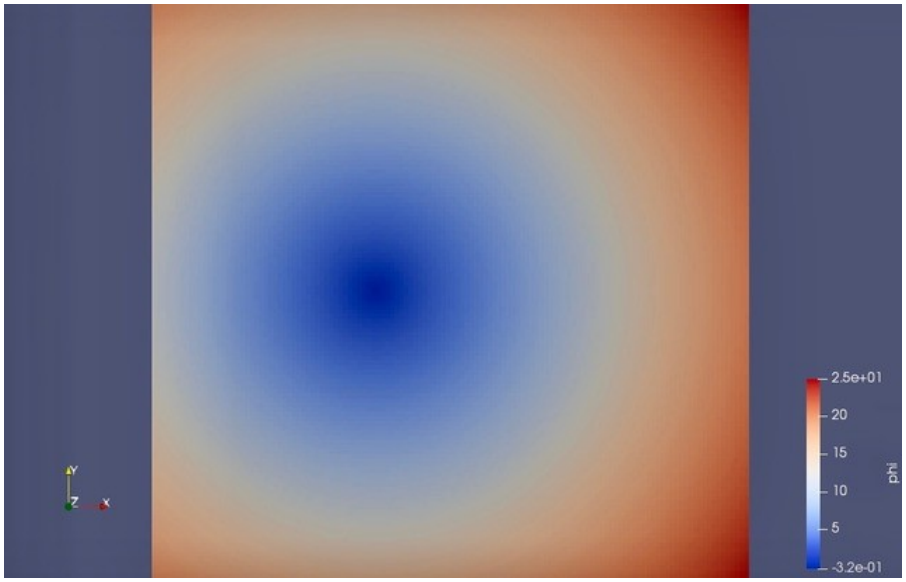


Figure 1.6: Level set function distribution for a solid cylinder immersed in the fluid domain.

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 [4], is positive if outside the solid and negative if inside ($\in \Omega_b$) fig.1.6. The solid-liquid interface is defined by $\phi = 0$. The level set function has to satisfies the condition

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

and the normal vector to the surface is

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

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 $\mathbf{u_B}$.

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

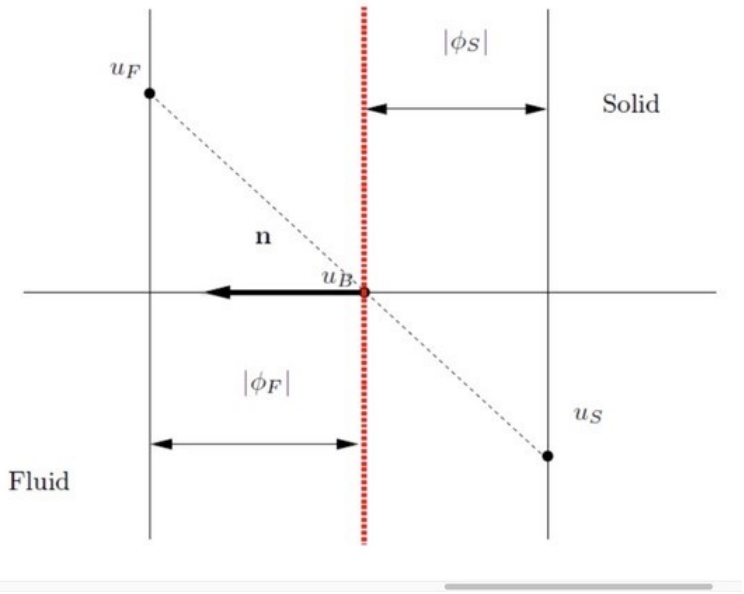


Figure 1.7: [3] one-dimensional scheme for second order penalization

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 neighbours.

1.5 Predictor corrector fractional step method

Solving the unsteady incompressible Navier-Stokes equation needs the introduction of a projection method, which can decouple the momentum and the continuity equations numerically by replacing the latter with the Poisson equation for the pressure term. A non-divergence-free velocity field is first obtained by omitting the pressure term p^{k+1} in the momentum equation and is then corrected using the pressure Poisson equation. Moreover in this work the adimensional form of the equations is used.
In others words 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.

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

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

At this point we need to discretize the equations. In literature a lot of method are outlined but for our purpose an implicit scheme is used.

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

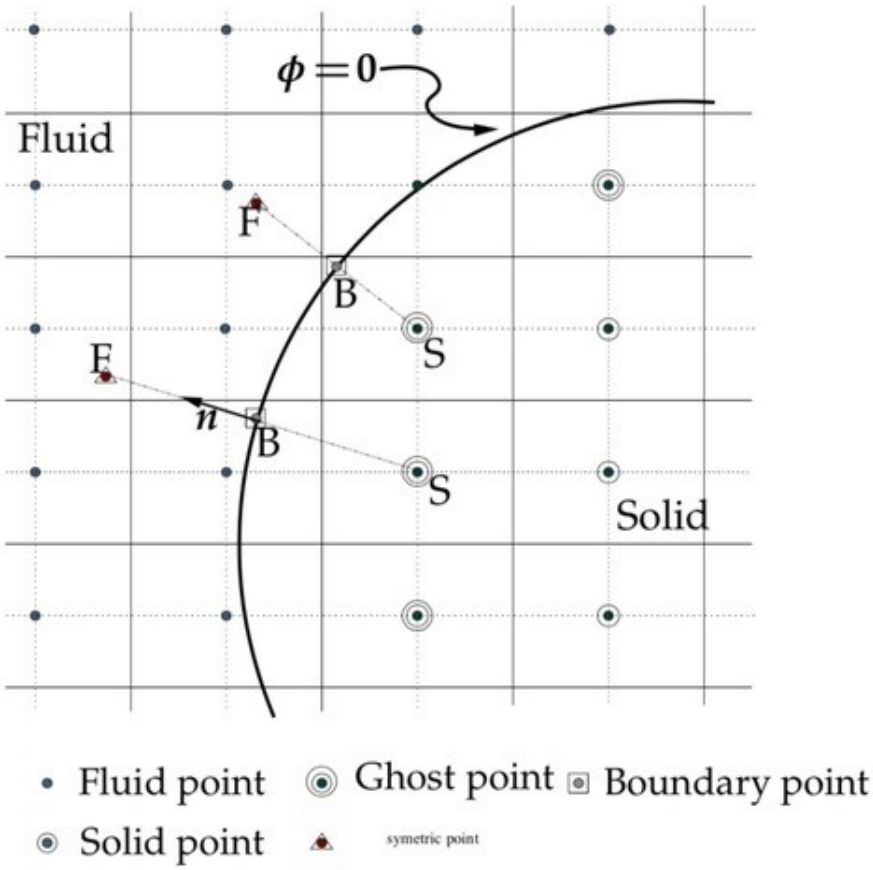


Figure 1.8: [3] sketch of the correction for the velocity in a two-dimensional case.

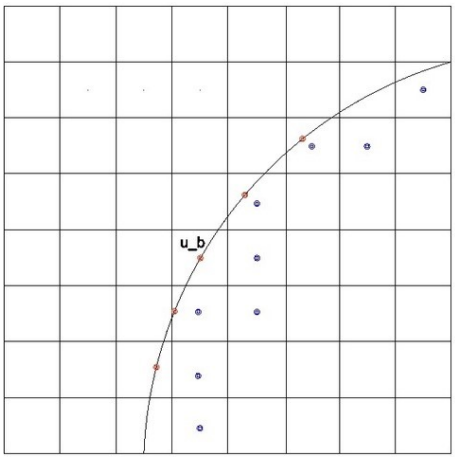
Now, we can add and subtract to the equation the terms $\mathbf{u}^*$ and $\frac{1}{\rho}\nabla p^n$ obtaining an equation that can be splitted in two different ones, the first will be used as the prediction step while the second one is solved as Poisson equation in the pressure known as projection step:

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

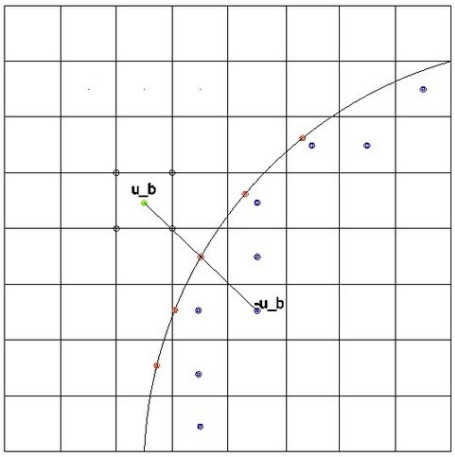
In the ***Prediction step*** we solve the first fractional step given a guess pressure field and in order to obtain $\mathbf{u}^*$. The following is the classical momentum equation but the forward step for the velocity field is substituted from a wrong velocity and the forward gradient of pressure by the pressure imposed. Obviously introducing these two wrong terms leads to a field that must be corrected in a further step.

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

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 $\mathbf{u_b}=[2\mathbf{u_c}-F(\mathbf{u}_i^*)]$

Since this velocity does not satisfy the divergence free condition from the continuity equation, as told before a correction step is required. Using the velocity field evaluated from the previous equation we can write the ***Projection Step*** .

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

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 Φ value that must be corrected by the **correction step** of the field:

$$\begin{aligned} p^{n+1} &= p^* + \Phi \\ \mathbf{u}^{n+1} &= \mathbf{u}^* + \Delta t \nabla \Phi \end{aligned} \tag{1.41}$$

1.5.1 Space discretization

For the discretization in space of our problem a first order upwind scheme is used for the advection equation. Let's consider a typical one-dimensional grid a generic point i in the domain . There are only two direction associated with the point i, the left ones,to the negative infinity and the right one from i to the positive infinity. Supposing a positive (to the right) velocity, the travelling wave solutiongoes to the right, we call the left side of the point i *downwind* while the left side *upwind*. Obviously in the same way we can define upwind and downwind side if the wave solutions moves to the left, since the velocity is negative.
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

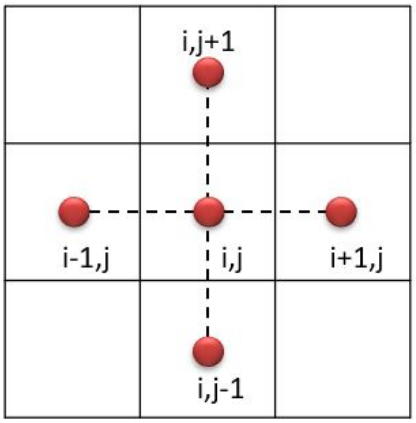
$$(\mathbf{u} \cdot \nabla)\mathbf{u} = \Big[\frac{u_i - u_{i-1}}{\Delta x} + \frac{v_j - v_{j-1}}{\Delta y}\Big]\mathbf{u}_j \tag{1.42}$$

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

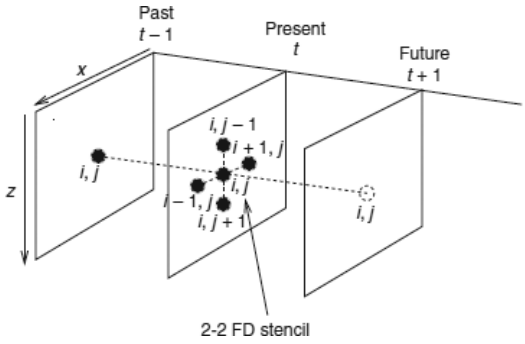
$$\nu \nabla^2 \mathbf{u} = \nu\Big(\frac{u_{x+\Delta x,y} - 2u_{x,y} + u_{x-\Delta x,y}}{\Delta x^2} + \frac{u_{x,y+\Delta y} - 2u_{x,y} + u_{x,y-\Delta y}}{\Delta y^2}\Big) \tag{1.43}$$

While for the pressure gradient a second order scheme is used. For further details see the section 3.1

$$\nabla p = \frac{p_{j+1} - p_{j-1}}{2\Delta x_i} \tag{1.44}$$



(a) Five points stencil, 2D case.



(b) 5 points stencil in two dimensional case.

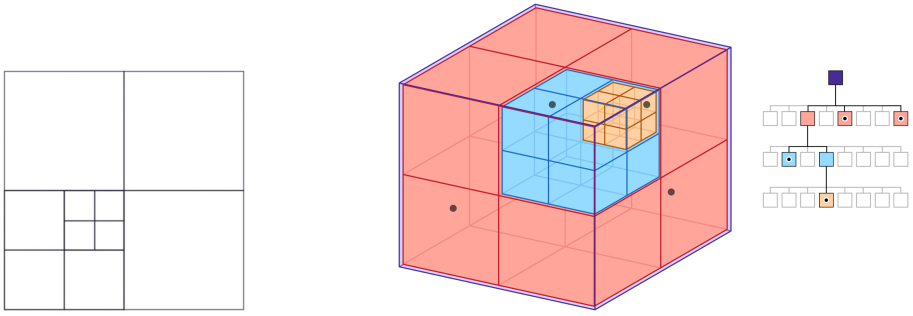
Chapter 2

Numerical domain

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 approach allows the user to solve problems that are intractable on an uniform grid. The approach we use is presented in [5] 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 2.1a).
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,



(a) A 2D quadtree subdivision of the grid (b) A 3D quadtree subdivision of the grid

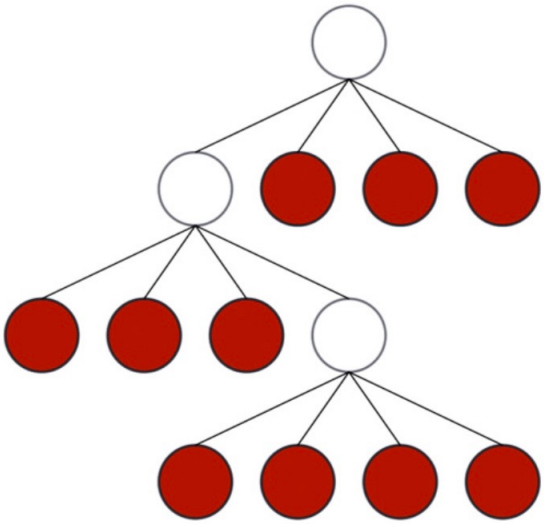


Figure 2.2: The decomposition of the grid represented as a quadtree

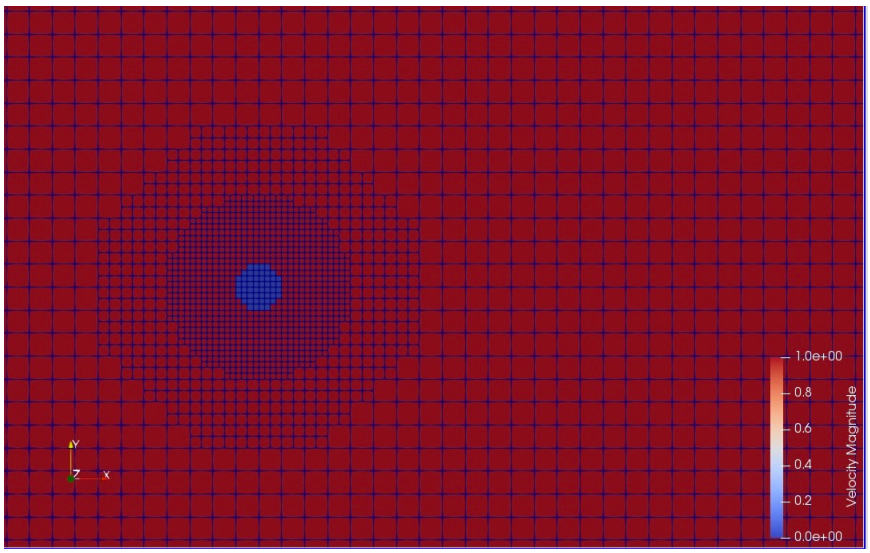


Figure 2.3: A solid cylinder in a fluid domain discretized with a cartesian hierarchical quadtree mesh

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. Neighbours to the cells can be defined through the surfaces or through the corners.

If each cell has all the neighbours 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 guaranted by the use of the library PABLO [6].

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 neighbours identification.

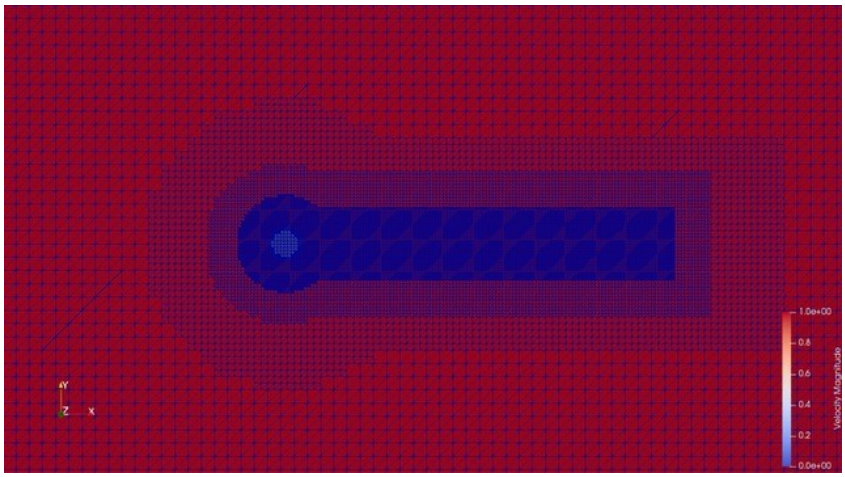
2.1.1 Cells' refinement criteria

The initial setting of the code used for simulation predicted a dynamic mesh refinement based on the value computed of the velocity gradient. The number of iteration between two grid refinement is an input given from the user. Such mechanism compute the gradient in each cell and evaluates, using a criteria, if it's necessary a cell subdivision. The criteria used for the refinement algorithm needs a minimum value and if the gradient computed overcomes such values the octant will be refined. Obviously, the dynamic refinement results in large computational costs each time the grid is computed. For our simulation this algorithm was ignored imposing a manual refinement where eddies and strong gradient occurs. In order to simulate the classical cylinder three refinement configuration has been adopted

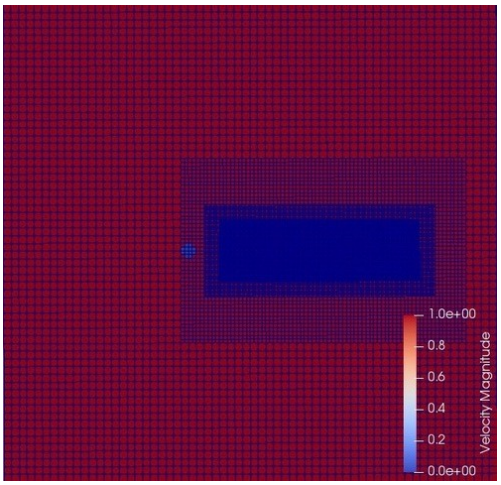
- Circular refinement: this approach leads a low computational costs since only the areas around the cylinder is refined, on the other hand the eddies that occurs in the wake are not represented in the best way since there is no refinement provided.

- 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.

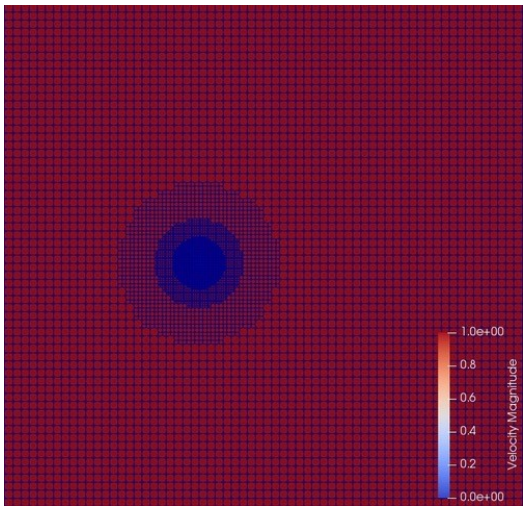
- Mixed refinement: such approach is a good balanced mix between the two previous method. A circular refinement on the cylinder wall is achieved while a rectangular ones provides the high resolution required in the wake to catch all the eddies generated after the transient.



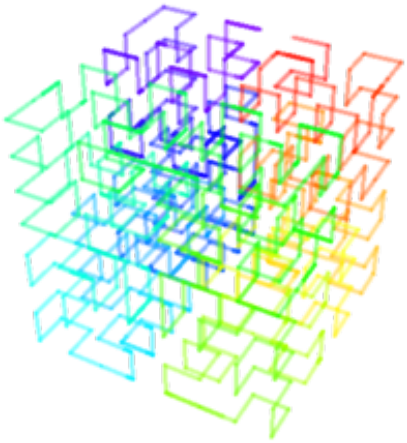
(a) Mixed refinement, 7 level with 3 refinement



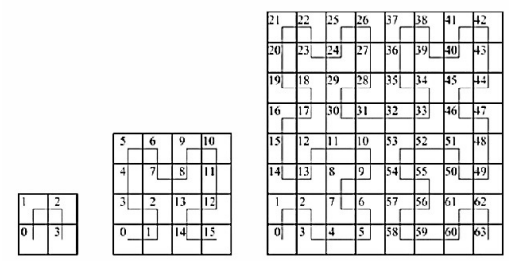
(b) Rectangular refinement, 7 level with 3 refinement



(c) Circular refinement, 7 level with 3 refinement



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



(b) A representation of the Hilbert curve ordering at different levels of 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 [9].

Hilbert curve

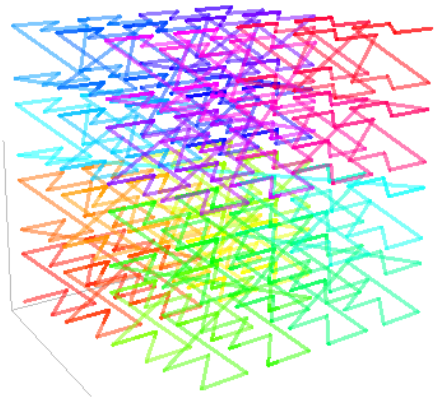
This curves was introduced by Hilbert in 19th century. Using this pattern each cell of a two-dimensional space can be represented with an index, with order shown in figure 2.5a . So it makes possible to map 2D spaces into 1D. To find Hilbert order many algorithms can be used that generates, and handle it.

Z-Order

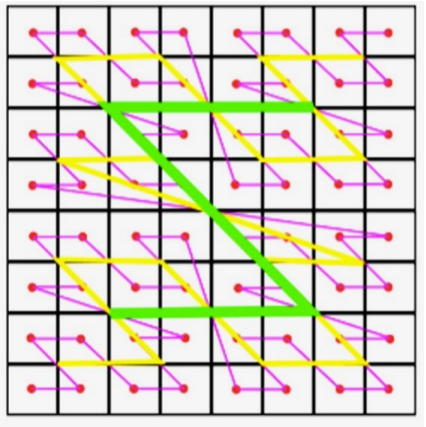
The Z-ordering was first introduced by Morten in 1966 [9]. It can be used to generate an ordered numeration of the cells while using an quadtree grid. The basic idea is to sort the cell index following a space filling line with an inverse Z shape. At each subdivision of a cell into four children to each 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.
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.

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



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



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

Table 2.1: Geometric parameters

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

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. 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. [8].

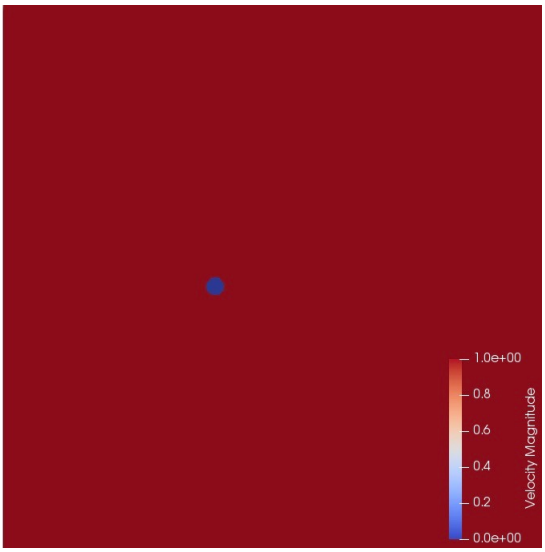


Figure 2.7: The circular cylinder in the fluid domain at t=0

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

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

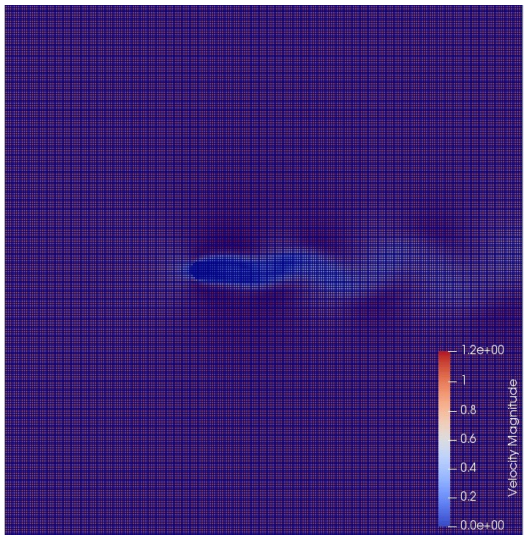
Flow initial condition	
ρ	1
U	1
T	1
p	1
Re	100

Table 2.3: Boundary condition used for this analysis

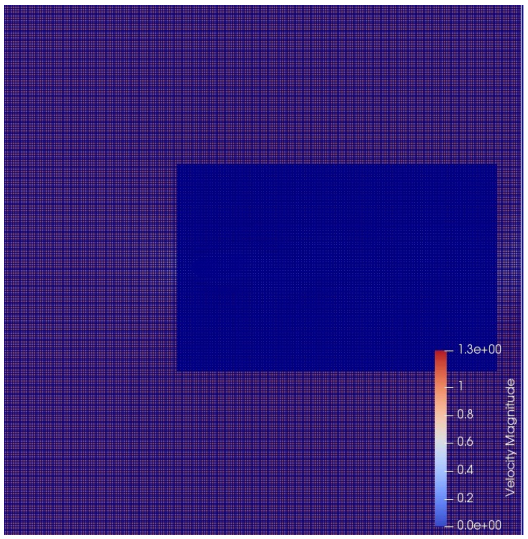
Boundary Condition		
Inflow	U=1	$\nabla p = 0$
Ouflow	Freeflow	
upper&lower boundaries	U=1	$\nabla p = 0$

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 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. AS we know the drag, after the transient period has an oscillatory behaviour that requires an average approximation to define an unique drag coefficient value. According with the above explained the grids used will be:

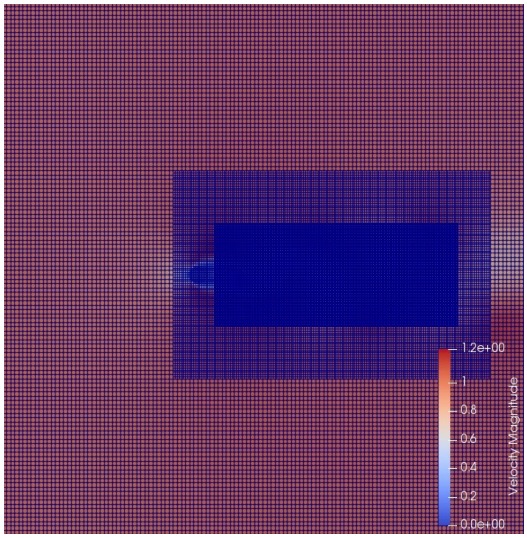
1. Level 10 with 0 refinements, figure 2.8a
2. Level 9 with 1 refinements, figure 2.8b
3. Level 8 with 2 refinements, figure 2.8c
4. Level 7 with 3 refinements, figure 2.8d



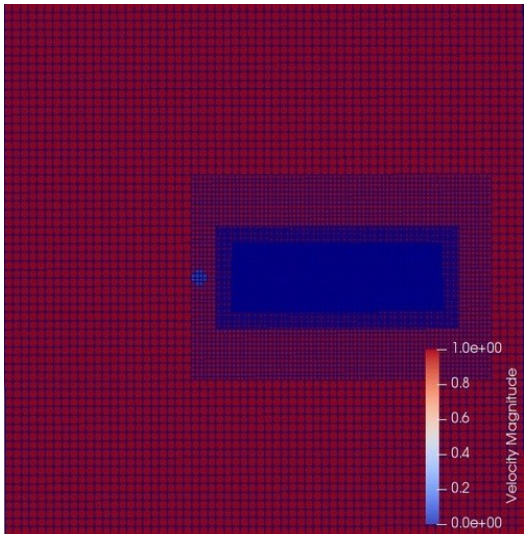
(a) Cartesian mesh, Level 10 with no refinement



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



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

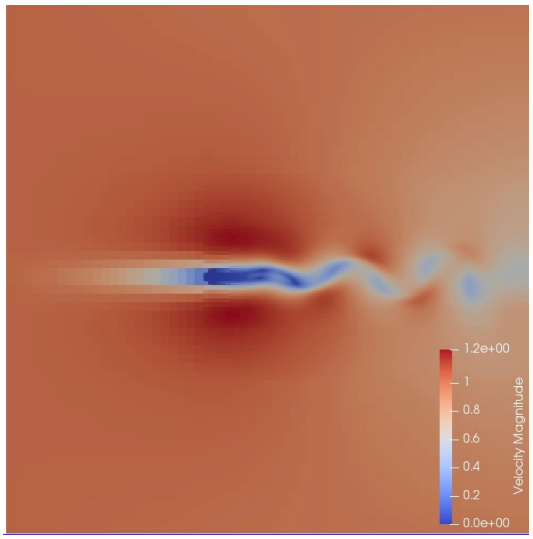


(d) Initial Level 7 with three refinement on the wake

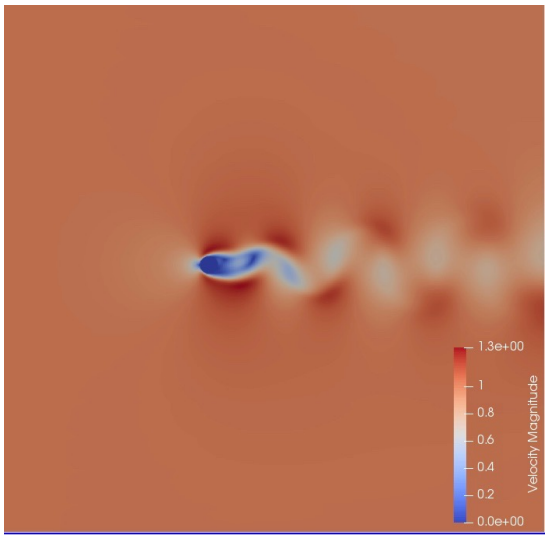
Figure 2.8: the grids used to discretize the domain

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 [s]	Tcpu [h]	reduction	#cell	#cell red
L10-0	$1,2534$	$6,13\%$	45.733	12 : 42 : 13	$0,00\%$	1048576	$0,00\%$
L9-1	$1,2483$	$6,52\%$	18.522	05 : 08 : 42	$59,50\%$	441856	$57,86\%$
L8-2	$1,2756$	$4,47\%$	9.609	02 : 40 : 09	$78,99\%$	179584	$82,87\%$
L7-3	$1,2620$	$5,49\%$	6.107	01 : 41 : 47	$86,65\%$	84832	$91,91\%$



(a) Velocity field after 500 s of physical time using the grid level 7 with three refinements



(b) Velocity field after 500 s of physical time using the cartesian grid

The result are presented in table 2.4. the computational time is significantly reduced where using the octree mesh 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 the graph 2.10 shows the time reduction versus the error rate referred to the drag coefficient value calculated by [8] and 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 error obtained with a significant time reduction.

To have a larger view of the error and number of cells behaviour 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

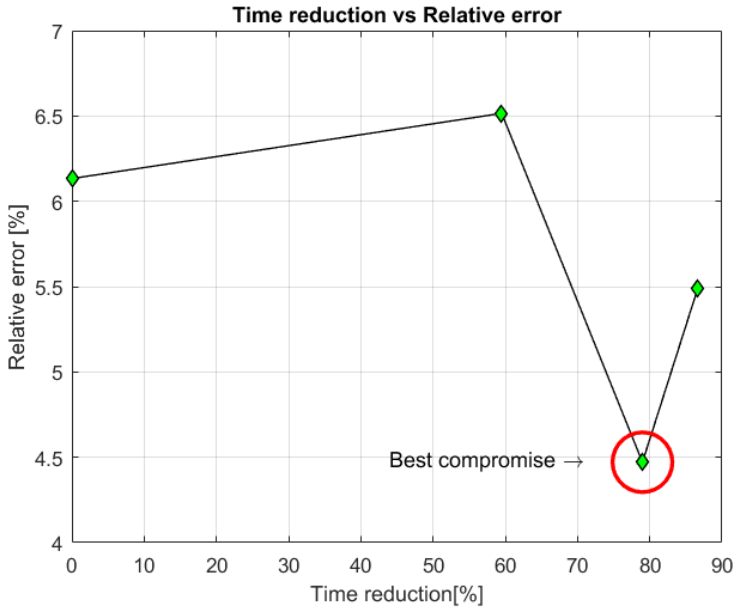


Figure 2.10: Best compromise evaluating the relative error and the time reduction of each grid. Old evaluation method.

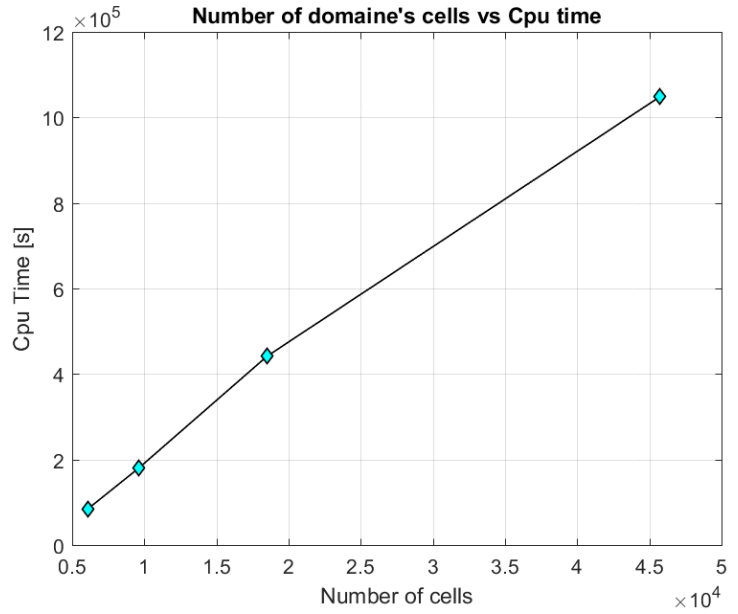


Figure 2.11: Quasi-linear relation between the number of cells and CPU time

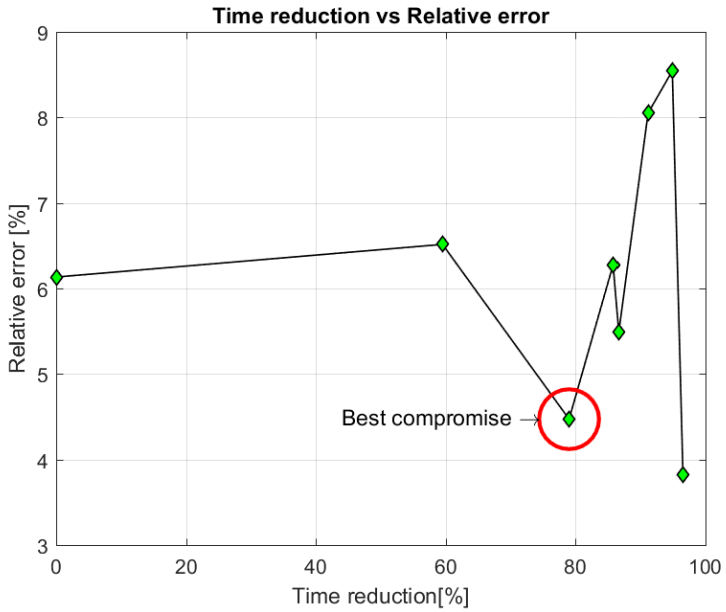


Figure 2.12: Global best compromise considering all the grid configuration

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: Result comparison between different grid configuration

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.

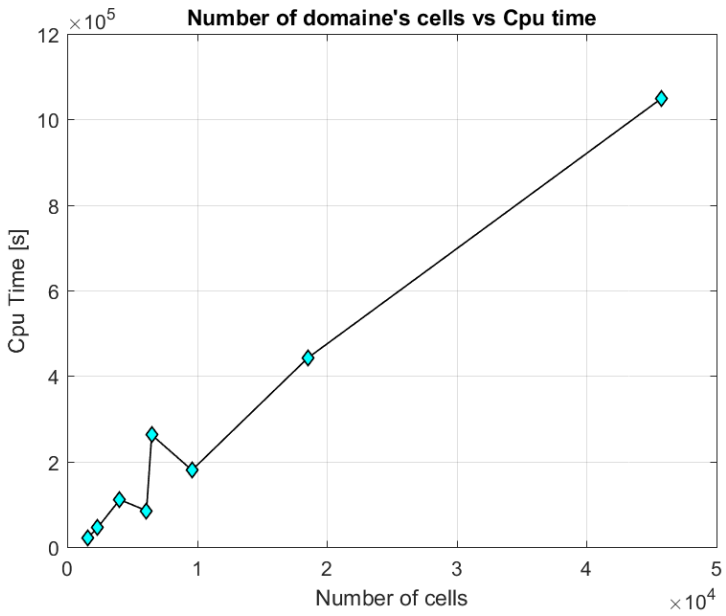


Figure 2.13: Global time-Number of cells relationship considering all the grid configuration

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 SMeCH code. This phase of this thesis project has been useful to better understanding how the code works and represent the first change provided in order to improve accuracy without losing convergence order.

Starting from the momentum balance equation, and neglecting the gravity contribute we obtain a general formulas for continuous field. Fixing the steady cylinder the equation that describes the total force acting on it is the following.

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

Discretizing the domain, the previous expression can be written as

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

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

It's required to evaluate the pressure and the velocity gradient for each control points to evaluate the force acting on it and sum the values following the previous formulas.

3.0.1 Control points

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 3.3 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 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 is difficult. To over come this problem we use a Lagrangian marker

approach. This consist in considering the points of the geometry as Lagrangian marker, 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.

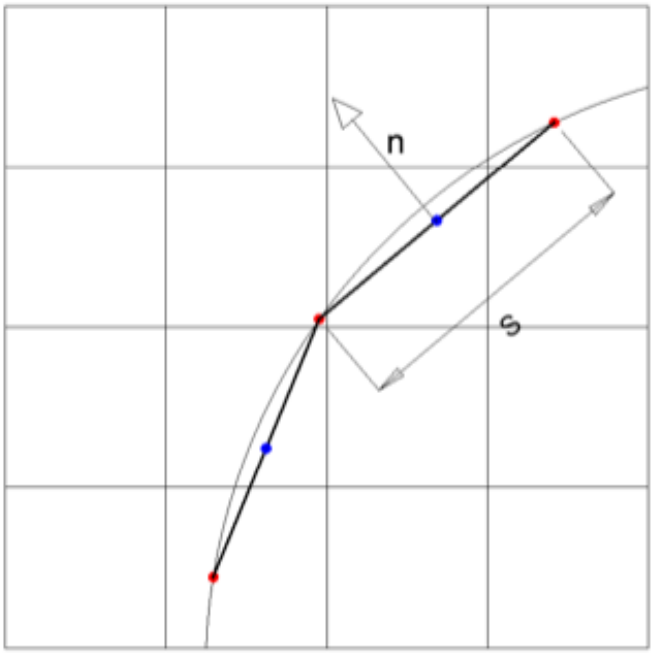


Figure 3.1: The points used to define the geometry of the solid body are used as lagrangian marker. The forces are computed on the middle point of the segment connecting two consecutive marker, where it is easy to calculate normal vector and surface of the cell, this control point are represented in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

3.1 Old interpolation method

From equation 3.3, 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 3.3) 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 criteria to chose the neighbours to use is required. Referring to figure 3.3 , from the i-th control point we move to the closest vertex (yellow arrow in fig. 3.3) and we chose as neighbours the four cells insisting on the vertex. The pressure is defined both in solid and fluid cells, while the gradient of velocity is defined only in fluid cells as the velocity is null inside the body since it has been defined as steady. So to interpolate the pressure all the cells $(1, 2, 3, 4)$ can be take into account, while for the velocity gradient only the fluid ones (the light-blue in 3.3 right side) will be used. The strategy is to evaluate $p_s + \nabla \mathbf{u}_s$ for each fluid neighbour . The value of the pressure is known in the cell centres of the neighbours cell while the gradient in each s-th point can be approximate using a second order finite difference scheme. Using Taylor series expansion to approximate the value of the velocity on the $(i+1)^{th}$ poits of our discretized domain (1D for semplicity) we can write:

$$u(x_{i+1}) = u(x_i) + (x_{i+1} - x_i)\Big(\frac{\partial u}{\partial x}\Big)_{x_i} + \frac{(x_{i+1} - x_i)^2}{2!}\Big(\frac{\partial^2 u}{\partial x^2}\Big)_{x_i} + \frac{(x_{i+1} - x_i)^3}{3!}\Big(\frac{\partial^3 u}{\partial x^3}\Big)_{x_i} + O(x_{i+1} - x_i)^4 \tag{3.5}$$

Now, we want to approximate the value of the velocity on the $(i+1)^{th}$ points, thus we can write

$$u(x_{i-1}) = u(x_i)(x_i - x_{i-1})\Big(\frac{\partial u}{\partial x}\Big)_{x_i} + \frac{(x_{i-1} - x_i)^2}{2!}\Big(\frac{\partial^2 u}{\partial x^2}\Big)_{x_i} - \frac{(x_i - x_{i-1})^3}{3!}\Big(\frac{\partial^3 u}{\partial x^3}\Big)_{x_i} + O(x_{i-1} - x_i)^4 \tag{3.6}$$

Taking the difference from the two expression we lead to

$$\Big(\frac{\partial u}{\partial x}\Big)_{x_i} = \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} - \frac{(x_{i+1} - x_i)^2 - ((x_i - x_{i-1})^2}{2(x_{i+1} - x_{i-1})}\Big(\frac{\partial^2 u}{\partial x^2}\Big)_{x_i} - \\ - \frac{(x_{i+1} - x_i)^3 - (x_i - x_{i-1})^3}{6(x_{i+1} - x_{i-1})}\Big(\frac{\partial^3 u}{\partial x^3}\Big)_{x_i} + H$$

The terms that were deleted at RHS are called the truncation error. The latter measure the accuracy of the approximation and determine the rate at which the error decreases. In the case of Δx constant the last equation can be wrote as

$$\Big(\frac{\partial u}{\partial x}\Big)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{3.7}$$

This is a *second order approximation of the first derivative*, for uniform Δx. Obviously there are different way to achieve this result, for instance fitting the function with a second order polynomial $(u(x) = a + bx + cx^2)$ over the same three point of the same stencil we considered before. The polynomial function wrote for the three points, and setting the origin of the coordinate system in x_i, we obtain three equation that defines the three coefficient of the function(a, b, c). Evaluating the first order derivatives in the origin of the coordinate system we get the same previous result. The gradient matrix can be computed with that expression, and the force on the control point is

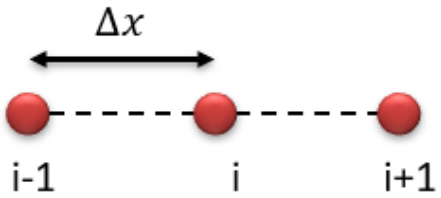


Figure 3.2: 1D Stencil for first derivative approxmation.

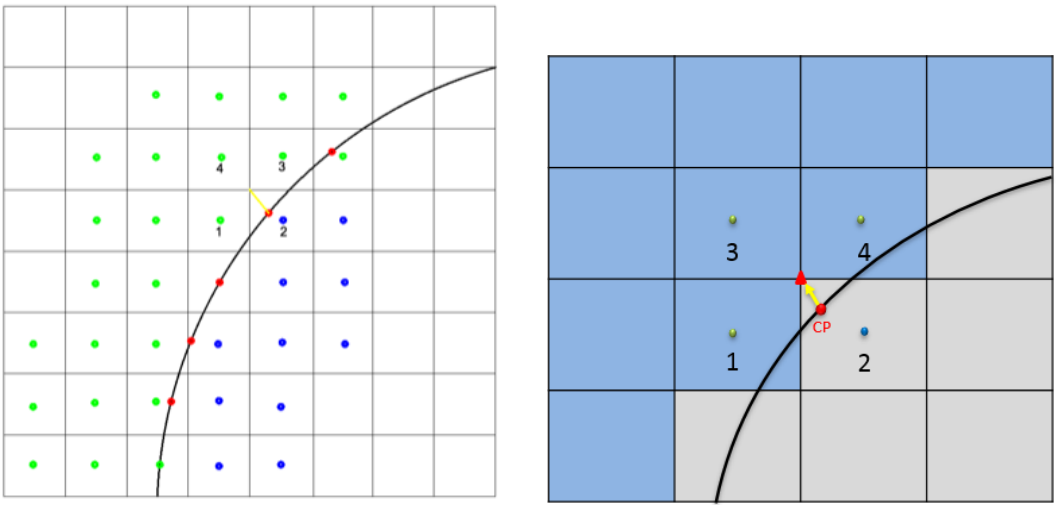


Figure 3.3: The figure represents the first method used to find neighbours. With light-blue color are outlined the fluid cells while with grey the solid ones

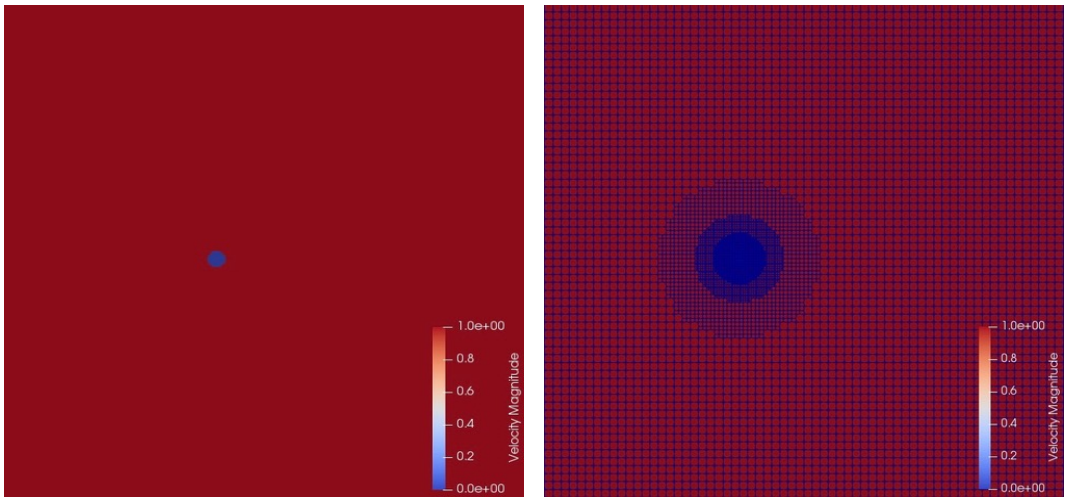
$$\mathbf{F}_i = \frac{1}{N} \sum_s \Big(p_s + \nabla \mathbf{u}_s \Big) \cdot S_i \cdot \mathbf{n}_i$$

As possible to appreciate, the previous method, used to do a simple average between the neighbour and assign that value to the control point. Obviously, this method shows few accuracy, so, a new algorithm was required in order to improve interpolation accuracy without further computational costs time.

3.1.1 Results

With the method previously presented we obtain the three 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 [8] 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.

$$C_D = \frac{F_x}{1/2 \rho u^2 S} \tag{3.8}$$



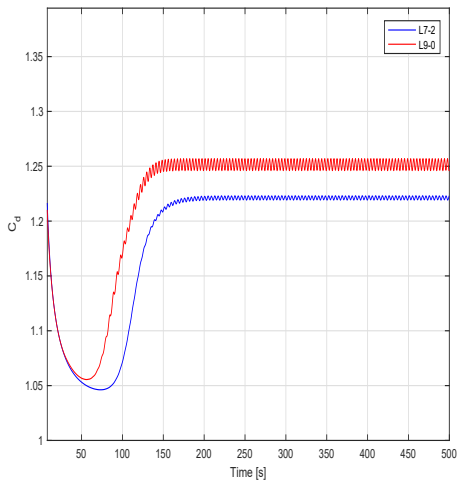
(a) (b)

Figure 3.4: The fluid domain used for the test, in 3.4a the velocity field at the initial condition, in 3.4b the quadtree grid with initial level 7 and three circular refinement around the surface of the cylinder.

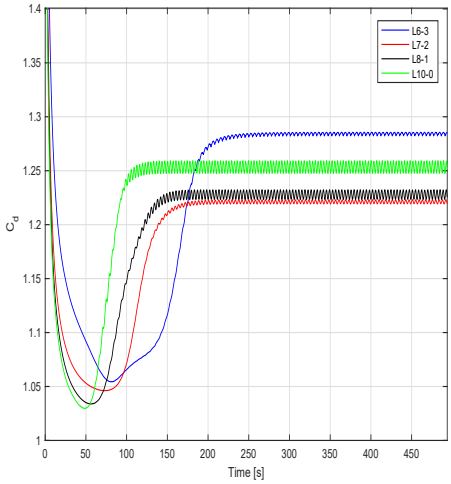
This results are the same already used to compare the quadtree grid to the uniform cartesian grid presented in the previous chapter.

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 3.5: 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.

3.2 New interpolation method

3.2.1 Interpolation of the gradient with the least square interpolation

In order to improve the accuracy of the method used to evaluate $p + \nabla \mathbf{u}$, we will focus on how the 2 terms are individually obtained. 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

$$\mathbf{u}_{(x_i)} = \mathbf{u}_{(x_0)} + (\mathbf{X_i} - \mathbf{X_0})\nabla \mathbf{u}_{(x_0)} \tag{3.9}$$

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 3.9 we can get the sum of the distances.

$$I = \sum_i \frac{1}{2}\Big[\mathbf{u_{x_i}} - \mathbf{u_{x_0}} - \nabla \mathbf{U}\|_{x_0}(\mathbf{X_i} - \mathbf{X})\Big]^2 \tag{3.10}$$

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.

$$\nabla \mathbf{u} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = \left\{ \begin{array}{c} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{array} \right\} \cup \left\{ \begin{array}{c} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{array} \right\} \tag{3.11}$$

$$\nabla \mathbf{U}\|_{x_0}(\mathbf{X_i} - \mathbf{X}) = \sum_{d=1}^{2} (\mathbf{X_i^d} - \mathbf{X^d}) \tag{3.12}$$

Where d is an index that represent the two vectors in which the gradient was divided. According to that and expanding the square, the 3.10 can be rewritten.

$$I = \sum_i \frac{1}{2}\Big\{\Delta\mathbf{u}^2 - 2\Delta\mathbf{u}_i\nabla\mathbf{u}\|_{x_0}(\mathbf{X_i} - \mathbf{X}) + \Big[\sum_{d=1}^{2}(\mathbf{X_i^d} - \mathbf{X^d})\Big]^2\Big\} \tag{3.13}$$

Where for convenience $\Delta\mathbf{u} = \mathbf{u_{x_i}} - \mathbf{u_{x_0}}$. Now we take the derivative of each component of the 3.13 with respect to the two vectors in which the gradient was divided. and we obtain two linear sistems.

$$\begin{cases} \dfrac{\partial I_u}{\partial \nabla_x u} = 0 \\ \dfrac{\partial I_u}{\partial \nabla_y u} = 0 \end{cases} \tag{3.14}$$

$$\begin{cases} \dfrac{\partial I_v}{\partial \nabla_x v} = 0 \\ \dfrac{\partial I_v}{\partial \nabla_y v} = 0 \end{cases} \tag{3.15}$$

Where $\nabla_x = \frac{\partial}{\partial x}$, $\nabla_y = \frac{\partial}{\partial y}$ and $I = \{I_u\ I_v\}$.

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

$$\begin{cases} \dfrac{\partial I_u}{\partial \nabla_x u} = \displaystyle\sum_i \frac{1}{2}\Big[0 - 2\Delta\mathbf{u}_i(x_i - x) + 2\Big((x_i - x)\frac{\partial u}{\partial x} + (y_i - y)\frac{\partial u}{\partial y}\Big)(x_i - x)\Big] = 0 \\ \dfrac{\partial I_u}{\partial \nabla_y u} = \displaystyle\sum_i \frac{1}{2}\Big[0 - 2\Delta\mathbf{u}_i(y_i - y) + 2\Big((x_i - x)\frac{\partial u}{\partial x} + (y_i - y)\frac{\partial u}{\partial y}\Big)(y_i - y)\Big] = 0 \end{cases} \tag{3.16}$$

$$\begin{cases} \displaystyle\sum_i \nabla_x u(x_i - x)^2 + \sum_i \nabla_y u(x_i - x)(y_i - y) = \sum_i \Delta u_i(x_i - x) \\ \displaystyle\sum_i \nabla_x u(y_i - y)^2 + \sum_i \nabla_y u(x_i - x)(y_i - y) = \sum_i \Delta u_i(y_i - y) \end{cases} \tag{3.17}$$

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

$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 \end{bmatrix} \begin{Bmatrix} \dfrac{\partial u}{\partial x} \\ \dfrac{\partial u}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \sum_i (x_i - x)\Delta u_i \\ \sum_i (y_i - y)\Delta u_i \end{Bmatrix} \tag{3.18}$$

The system 3.15 is obtained deriving the second component of I and imposing it equal to zero we can evaluate the last two components of the gradient.

$$\begin{cases} \dfrac{\partial I_v}{\partial \nabla_x v} = \displaystyle\sum_i \frac{1}{2}\Big[0 - 2\Delta\mathbf{u}_i(x_i - x) + 2\Big((x_i - x)\frac{\partial v}{\partial x} + (y_i - y)\frac{\partial v}{\partial y}\Big)(x_i - x)\Big] = 0 \\ \dfrac{\partial I_v}{\partial \nabla_y v} = \displaystyle\sum_i \frac{1}{2}\Big[0 - 2\Delta\mathbf{u}_i(y_i - y) + 2\Big((x_i - x)\frac{\partial v}{\partial x} + (y_i - y)\frac{\partial v}{\partial y}\Big)(y_i - y)\Big] = 0 \end{cases} \tag{3.19}$$

$$\begin{cases} \sum_i \nabla_x v(x_i - x)^2 + \sum_i \nabla_y v(x_i - x)(y_i - y) = \sum_i \Delta v_i (x_i - x) \\ \sum_i \nabla_x v(y_i - y)^2 + \sum_i \nabla_y v(x_i - x)(y_i - y) = \sum_i \Delta v_i (y_i - y) \end{cases} \tag{3.20}$$

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

$$\begin{bmatrix} \sum_i (x_i - x)^2 & \sum_i (x_i - x)(y_i - y) \\ \sum_i (x_i - x)(y_i - y) & \sum_i (y_i - y)^2 \end{bmatrix} \begin{Bmatrix} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \sum_i (x_i - x)\Delta v_i \\ \sum_i (y_i - y)\Delta v_i \end{Bmatrix} \tag{3.21}$$

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.

3.2.2 Pressure interpolation 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

$$\phi(\mathbf{x}, \mathbf{c})) = \phi(||\mathbf{x} - \mathbf{c}||) \tag{3.22}$$

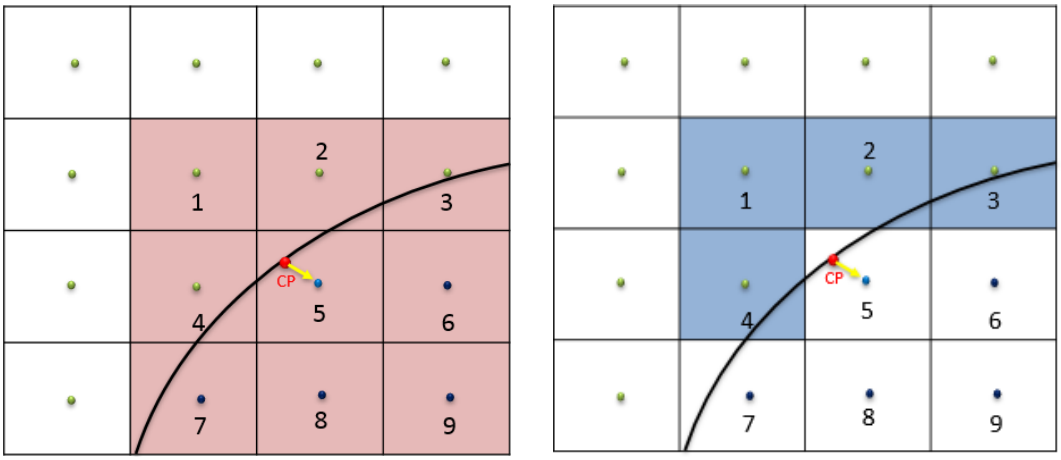
Such functions, as wrote, depend only from the radial distance from a fixed point. The norm usually is the Euclidean distance.Sums of radial basis functions are typically used to approximate given functions. RBFs functions are typically used to build up function approximations of the form

$$y(x) = \sum_{i=1}^{N} w_i \phi(||\mathbf{x} - \mathbf{x_i}||) \tag{3.23}$$

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.

3.2.3 Neighbours selection

The preceding method didn't have many restrictions in terms of neighbours selection, the method could be used with any number of fluid neighbours 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 neighbours, as in the case brought in figure 3.6. In the case showed below cells from 1 to 9 ahve been selected for interpolation. The fluid cells are highlighted with green cell center while the non-fluid ones with blue cell center. So the choice of the neighbours used to interpolate the gradient and the pressure 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 neighbours, the interpolation is made using only the fluid cell among the nine taken into account. On the other hand all the nine neighbours can be used to interpolate pressure as it's defined also in the fluid cells..



(a) Neighbours selected for pressure interpolation.

(b) Neighbours selected for ∇V interpolation.

Figure 3.6: 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.

3.3 Rate of convergence

In this section we will discuss the main results of this chapter. As seen in the previous pages the ∇V computation, to evaluate the force acting on the body considered, has been improved. Overall, the new method doesn't allow to have a fastest rate of convergence but highly accuracy has been achieved without losing speed. One can write the analytical solution as the numerical ones plus an error coming from the discretization process. Writing as ϕ_{an} the analytical solution and as ϕ_{num} we obtain, using the O notation

$$\phi_{an} = \phi_{num} + O(\Delta x)^m \tag{3.24}$$

(Δx) represent the space discretization of the generic grid. As we can see, the discretization error obtained subtracting the analytical value and the exact solution in a function of the space discretization. Taking the norm of the difference we obtain

$$||\phi_{an} - \phi_{num}|| = +O(\Delta x)^m \tag{3.25}$$

Using the logarithm for both equation sides

$$log(||\phi_{an} - \phi_{num}||) = m \cdot log(\Delta x) \tag{3.26}$$

At this point, following a graphical approach, the value of n represent the slope of the interpolated value $log(||\phi_{an} - \phi_{num}||)_i$ for each $log(\Delta x)_i$ taking care that

$$(\Delta x))_{s+1} = \frac{(\Delta x)_s}{2} \tag{3.27}$$

Where $(\Delta x)_s$ is the space discretization of the s^{th} simulation. It's necessary interpolate with a linear function the data obtained and find the coefficient related to the unknown value of the classical formula $y = mx + q$, in this case m.
Using a different approach, a log-log plot the slope is obtained selecting two point as shown in 3.7. Choosing two point of the line

$$log(Err_1(\Delta x_1)) = m \cdot log(\Delta x_1)$$
$$log(Err_2(\Delta x_2)) = m \cdot log(\Delta x_2)$$

The slope m is found taking the difference between the two previous expression:

$$m = \frac{log(Err_2) - log(Err_1)}{log(\Delta x_2) - log(\Delta x_1)} \tag{3.28}$$

$$m = \frac{log\frac{(Err_2)}{(Err_1)}}{log\frac{(\Delta x_2)}{(\Delta x_1)}} \tag{3.29}$$

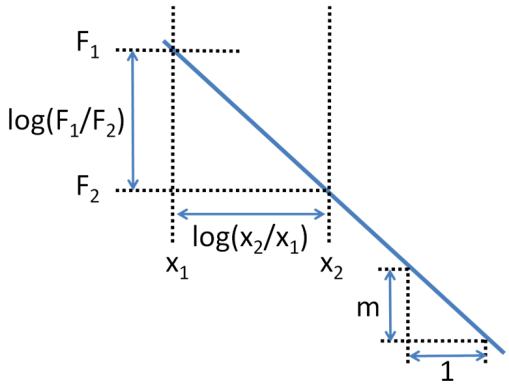


Figure 3.7: Finding the slope of a log-log plot using the ratios from [14]

Table 3.2: Coordinate of the first control point

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

3.3.1 Single point convergence Order

To evaluate the convergence order on our particular test case, for first we considered a single *control point* of the geometry and after the whole one. We used to impose an analytical field for velocity and pressure, so, the value of the velocity and pressure was known in each point as imposed. When the pressure and velocity field is imposed the interpolated value has been extract and stored in a table. Such data were used to evaluate the rate of convergence.

Such point, as shown by the coordinates value, is the first one created by the code and located almost at the . As analytical field we use the data reported in table 3.3. The function $sin^2(x)$ for the pressure ensure an always positive value.

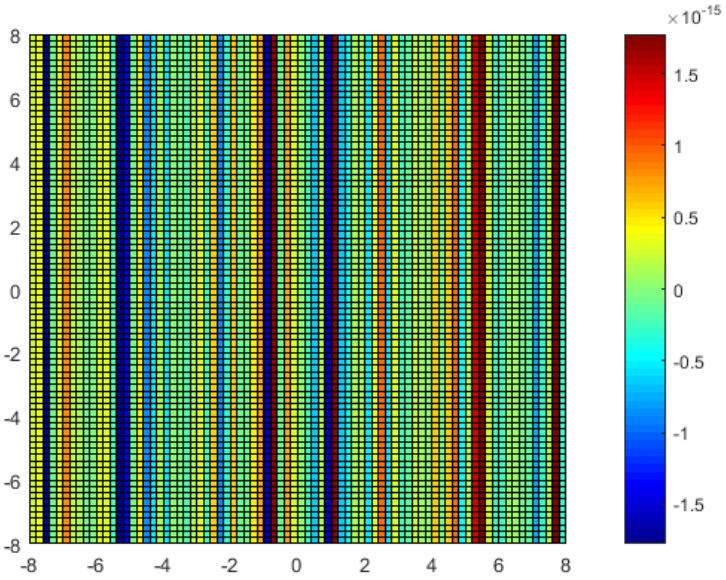
Table 3.3: Imposed Flow field

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

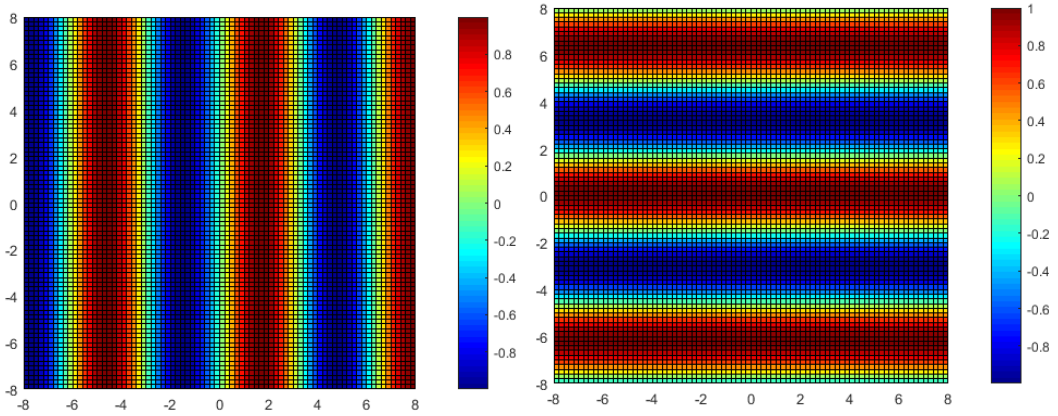
The value of the gradient of velocity and the pressure, both evaluated in the control point wrote before are shown in table 3.4.

∇V convergence order

Now the error of the numerical field has been evaluated respect the exact value showed in 3.4 for the different grid configuration as reported in the previous



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



(b) $u = \sin x$ (c) $v = \cos y$

chapter for the grid analysis. Obviously to ensure a spacial step size always divided by two the grid level has been increased each simulation. The octant dimension has been reported in the last column of the table 3.5

Concerning the gradient, only the component du/dx has been token into account foor the error evaluation, since the components dv/dx and du/dy values are about the same order of the machine epsilon so, the error operation might lead to a *Loss of significance*. By the difference of the numerical values in 3.5 and the exact ones in the table 3.4 we obtain the table of the errors .

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	$-0,03141$	$0,019778174$

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	$-0,989749$	$5,83438 \cdot 10^{-}17$	$-0,00303$	$-0,03515$	$0,007813$

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 the logarithmic value of the error and the Δx is required and the plot is in a log-log axis enumeration, such as the slope represent the convergence rate. As we can see from the figure the slope is almost one, that means that the error will be divided by two if the grid dimension will be divided by 2. As told before this way to evaluate the gradient does not increase the rate of convergence to the solution, but the accuracy obtained is higher than the previous method. A sharply decrease of the error will be shown after focusing on the error of the drag coefficient on the whole cylinder.

Pressure convergence order

The same approach could be done for the pressure. An analytical field for the pressure is imposed, so the value is exactly known everywhere in the whole domain. With the *RBF* interpolation we can evaluate the pressure in the control point chosen outlined in table 3.2 and compare the results with the analytical ones.

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	$0,01979160$	$-11,21829058$	$-4,85203$

Again, to evaluate the rate of convergence, the logarithmic value of the error and the Δx is required and the plot is in a log-log axis enumeration, such as the slope represent the convergence rate. For this convergence order upgrading an RBF interpolation scheme has been used. Such method has been written by Ing. Bernard and written in the code, in the file *interpolator.tpp* .

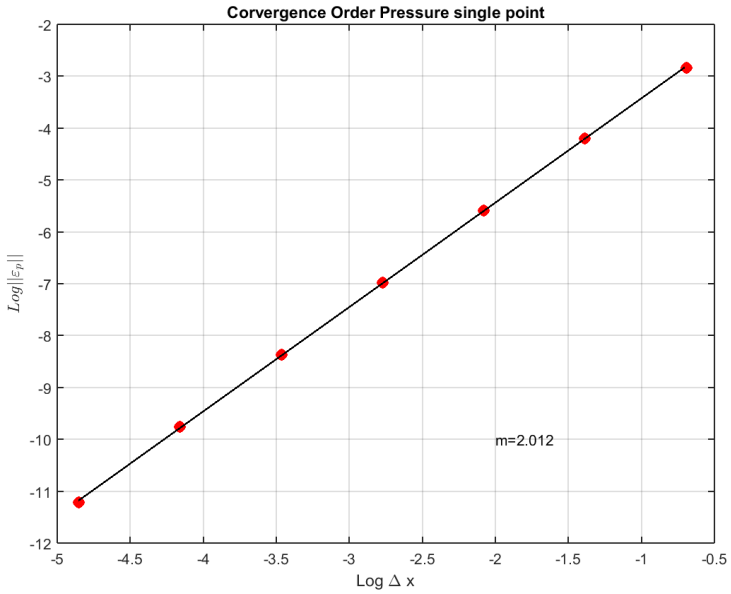


Figure 3.9: Pressure Convergence rate over a single point

3.3.2 All points convergence Order

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 such convergence using more control points of the geometry. As done in the previous section the same analytical field both for pressure and gradient have been imposed. Both interpolation methods have been adequately explained before, so using this approach we used to sum all the interpolated value of each control

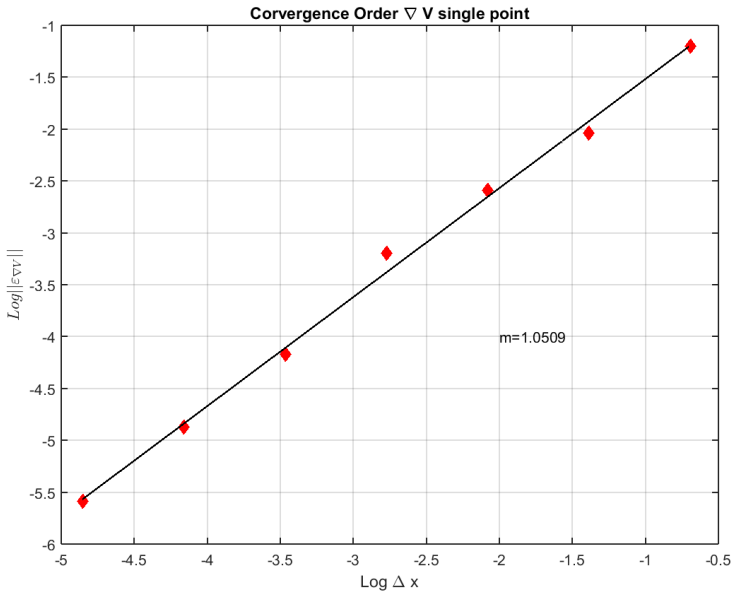


Figure 3.10: ∇V Convergence rate over a single point

point, both for pressure and ∇V. As reported, only the component du/dx has been considered to avoid *Loss of significance* for the error calculation.

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	$0,010489$	$6,08E-05$	$-4,557466326$	$-9,7071201$	$-4,15888$

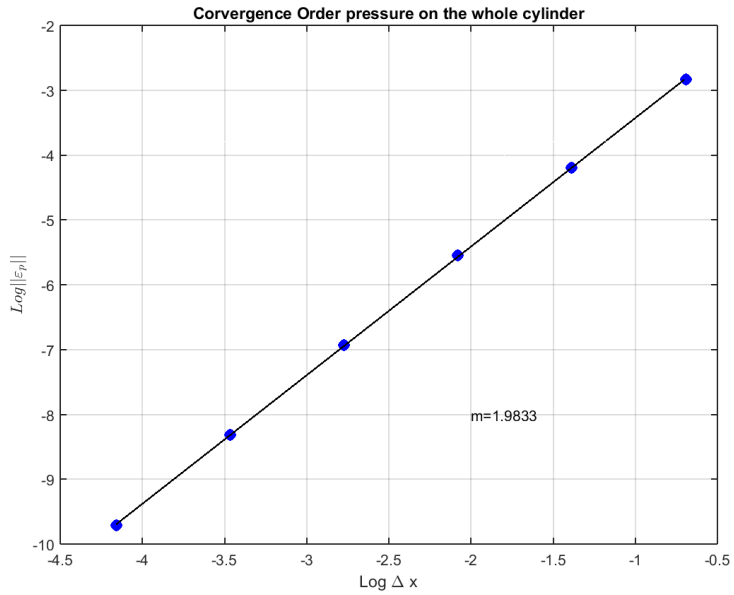


Figure 3.11: Pressure Convergence rate, all control points

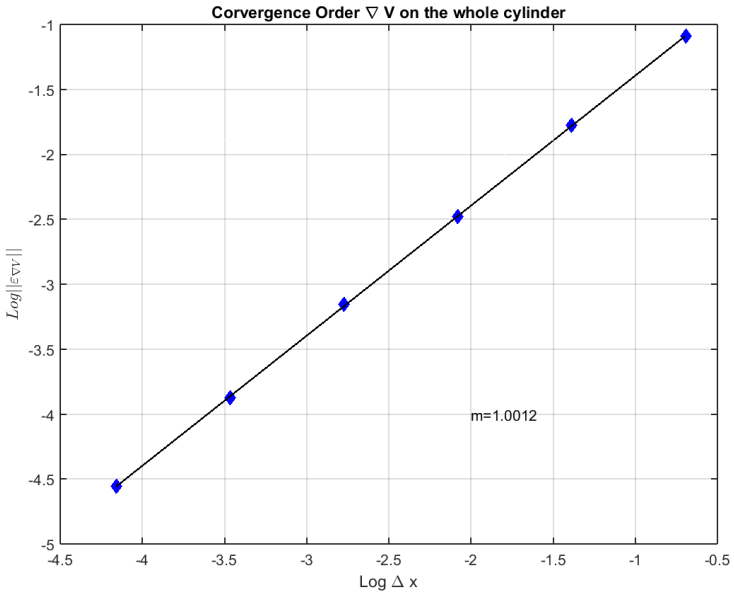


Figure 3.12: ∇V Convergence rate, all control points

The results obtained in this section is almost the same of the previous ones, with a first order rate of convergence on the ∇V and a second order for the pressure.

3.4 Drag coefficient 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.

Table 3.9: Results obtained with the optimized approach to the Lagrangian marker method.

Grid	Cd_{av}	$\varepsilon\%$	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 coefficient has been evaluated and the results are then compared to 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 [8] for a flow past a cylinder at reynolds 100.

$$C_{D_{REF}} = 1,3353 \tag{3.30}$$

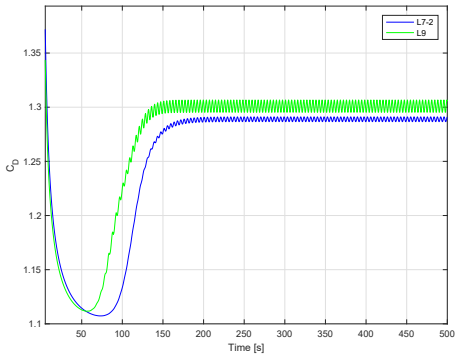
The relative error is evaluated as:

$$\varepsilon\% = \frac{C_D - C_{D_{REF}}}{C_{D_{REF}}} \times 100 \tag{3.31}$$

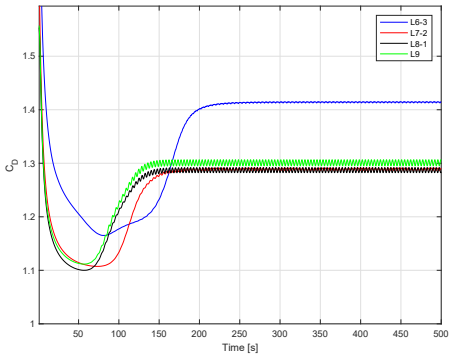
An the results are the following.
In Figure 3.14 are shown different flow field configuration of the same simulation evaluate for different physical time. Until $40s$ the field looks like homogeneous with no oscillation phenomena in the rear part of the cylinder. Vortex structure appears nearly $100s$ with the classical alternate and constant frequencies eddies generation according to the literature and experimental data. Firstly the graphs showed in 3.16 want to summarize the results obtained. The first one shows a *quasi-linear* behaviour of the code since the CPU time is almost proportional to the number of the cells. The same result has been obtained in the paragraph 2.3 where have been reported the results obtained using Octree domain subdivision.

3.4.1 Comparison

In this section the main results obtained from the implementation of the least square method and the use of the RBF interpolation will be shown. Regarding the table the relative error has been significantly reduced for every grid configuration used. In addition the predicted best grid as 7^{th} with 3 refinements obtaining an error less than 1%.As it's possible to see 3.16 the drag coefficient value was underestimated before the new interpolation method.



(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.

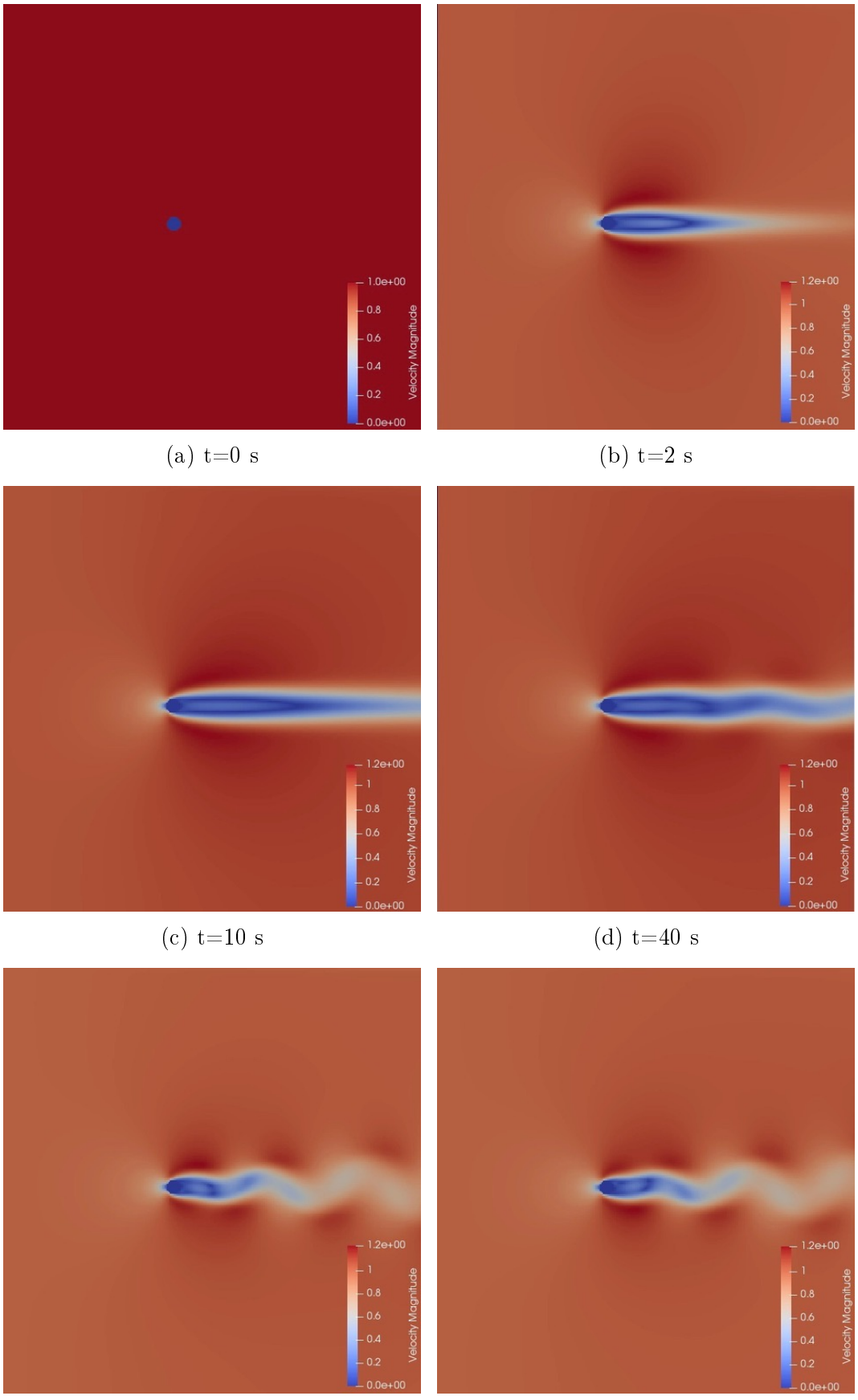


(b) The results compared with the C_D obtained using several grids with differen discretizedation.

Figure 3.13: 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.

Table 3.10: Drag coefficient comparison between the average method and the LS and RBF interpolation.

Grid	Averaged Cd	abs(ε%)	LS and RBF Cd.	ε%
$L10$	$1,2534$	$6,1335$	$1,318$	$1,30$
$L9_1$	$1,2483$	$6,5154$	$1,3599$	$1,84$
$L8_2$	$1,2756$	$4,4709$	$1,3158$	$1,46$
$L9$	$1,2515$	$6,2757$	$1,3009$	$2,58$
$L7_3$	$1,262$	$5,4894$	$1,3269$	$0,63$
$L8_1$	$1,2278$	$8,0506$	$1,2872$	$3,60$
$L7_2$	$1,2212$	$8,5449$	$1,2888$	$3,48$
$L6_3$	$1,2843$	$3,8194$	$1,4142$	$5,91$



(a) t=0 s

(b) t=2 s

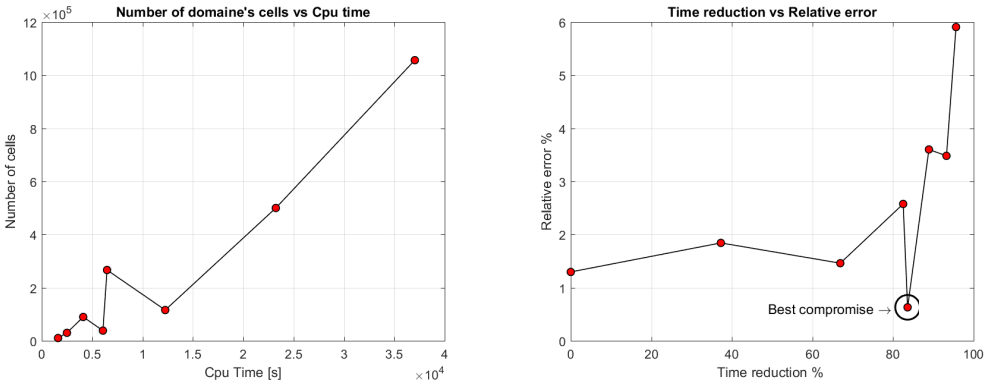
(c) t=10 s

(d) t=40 s

(e) t=100 s

(f) t=500 s

Figure 3.14: The velocity field of the flow past a unit diameter cylinder at $Re = 100$. In 3.4a are reported the initial conditions t=0s, in 3.14fthe field after the end of the transient, the flow is completely developed at t=500s



(a) The relation between CPU time and the number of cells of the domain, varying the discretization level

(b) Best compromise between the time reduction and the relative error referred to the literature value 1.3353

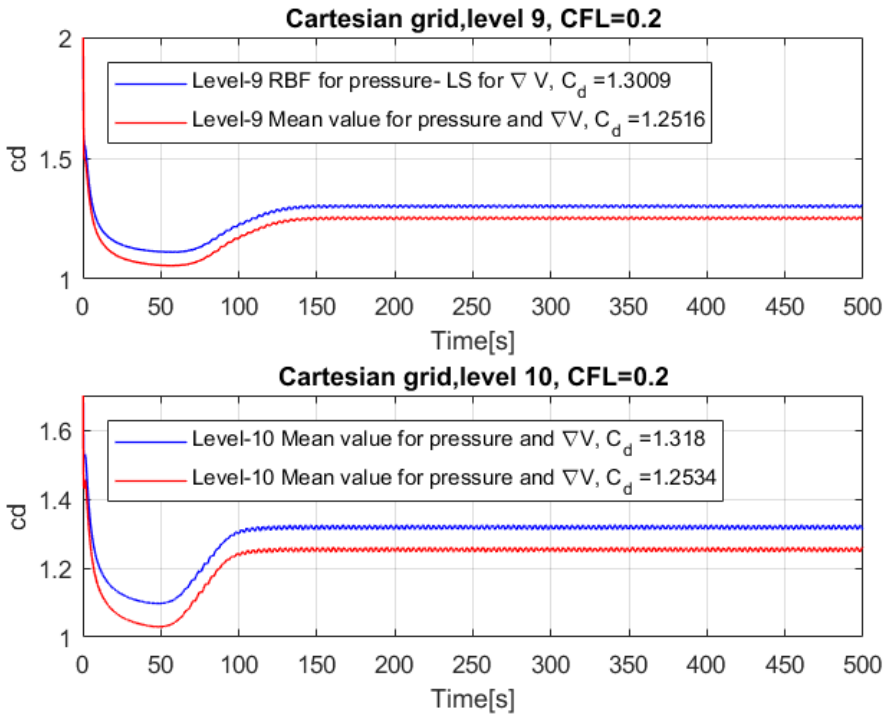


Figure 3.16: Drag coefficient comparison for the mean interpolation method and the LS with RBF interpolation for pressure values. The values are available in the legend.

Chapter 4

Windkessel model

According to the physiological literature, the aorta is the largest artery, originating from the heart and extending down to the abdomen, where it is divided in several smaller arteries.
The blood from the aorta flows to the visceral organs and to the peripheral regions in the systemic circulation. The aorta has a complex, three-dimensional curved geometry with multiplanar curvature. The aorta begins at the aortic valve and makes its first bend, past the pulmonary vessels and bronchi. A second curvature allows it to bypass the esophagus and the trachea. In the end the last plane of curvature allows it to bend around to the left atrium.
The Aorta can be divided in three main sectors

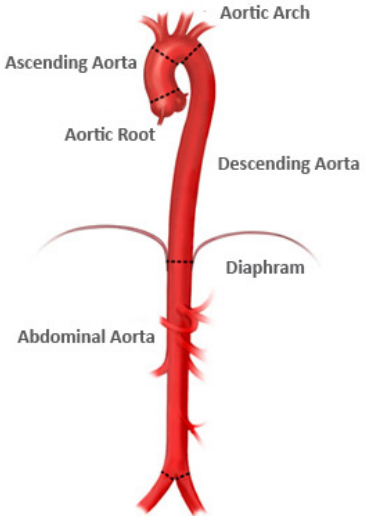


Figure 4.1: Ascending aorta from[11].

- **The Ascending Aorta** is the initial portion of the aorta. It has its origin at the level of the aortic valve. It runs obliquely, upward and to the right, until it reaches the edge of the second costal cartilage, where it ends by continuing in the aortic arch.

- **The Aortic arch** follows the ascending aorta at the level of the second costal cartilage. It moves back, to the left, to reach the left margin of the body of the fourth thoracic vertebra where it continuing with the descending aorta. From aortic arch originate the brachiocephalic trunk (innominate artery), which divides into the right subclavian and right carotid artery, the left carotid and the left subclavian artery, which carry blood to the head, neck and shoulders.

- **The descending aorta** is the last section of the arta an it's composed by two sections: thoracic aorta and abdominal aorta

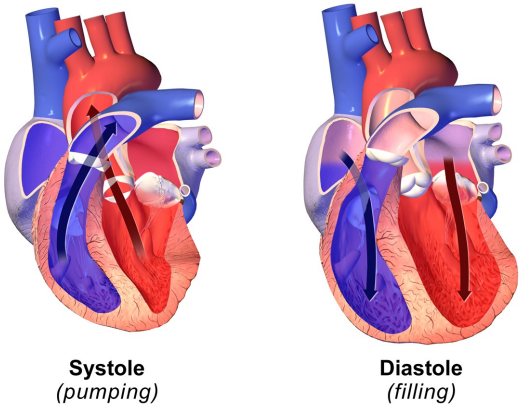


Figure 4.2: Systole and diastole from [12]

The cardiac Cycle is a closed-loop where the heart works as a pump throughout the systemic circulation. The first phase of the cardiac cycle is the **ventricular diastole**, when the ventricles are relaxed and the oxygenated blow flows in from the atria. That phase is followed by a **Systole**, where the ventricles contract and eject blood through the aorta. The pressure, inside the aorta, rises and the maximum is called the *Systolic pressure*. At the end of this pumping phase the pressure decrease to a minimum value, and it's known as *diastolic pressure*.

The **Windkessel Model** was designed by the german physiologist Otto Frank. He described the previous cardiac cycle as a closed hydraulic circuit. These models are commonly used the load undertaken by earth during the cardiac cycle and relates the blood pressure with the flow. The model characterize the *arterial compliance*, *peripheral resistance* due the valves and the inertia of the blood flow. A generic Windkessel model takes into account the following parameters while modelling the cardiac cycle.

- **Arterial compliance** related to the elasticity of the major artery in the entire cardiac cycle.
- **Peripheral resistance** modelize the resistance that the blood meets flowing in the aorta.
- **Inertia** simulates the inertia of the blood.

In conclusion, this model, allows us to find a relationship between blood pressure and blood flow in the aorta taking into account resistance and compliance due the vessels and the valves. The most classical approach is to compute the exponential pressure curve determined by the systolic and diastolic phases. Increasing the number of the elements the model is composed of, new physiological factor are accounted that leads more accurate results when related to the original curve.
The dimpliest Winkessel models is the *2-Element* one.

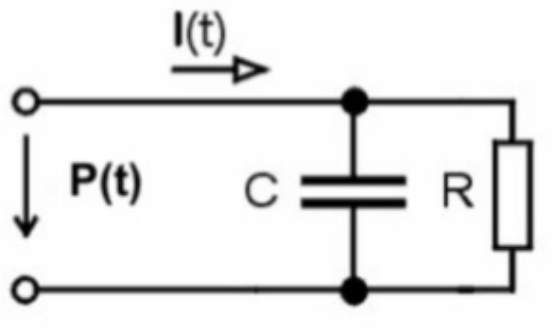


Figure 4.3: Analog: the 2 elements electrical model

4.1 The 2-Element Windkessel Model

The simplest way to demonstrate the hemodynamic state is the *2-Element model* as described in [10]. The model uses the electrical analogy. The two elements that the model is composed of, simulates the arterial compliance represented as a capacitor (C in $cm^3/mmHg$) with electric charge storage properties, and the resistance due the arterial system (R in $mmHg \cdot s/cm^3$) represented by a classical resistor. The blood flow can be seen as the current flowing in the circuit (I(t) in cm^3/s) and the pressure can be seen as a time-varying electric potential (P(t) in $mmHg$). The electric model as seen in fig. 4.3 is given as :

$$I(t) = \frac{P(t)}{R} + C\frac{dP(t)}{dt} \tag{4.1}$$

The flow blood has, as pressure, a time-variable law. In particular the diastolic phase, when the ventricles are relaxed, is characterized by $I(t) = 0$. However, ventricular contraction define the systolic phase while $I(t) \neq 0$, and it's modelled as a sine wave with amplitude I_0. Blood is ejected in the aorta and can be modelled as a sinusoidal wave, as done in

$$I(t) = I_0 \sin\Big(\pi \cdot \frac{mod(t, T_c)}{T_s}\Big) \tag{4.2}$$

Where:

- t is the rime in seconds
- T_c is the period of the cardiac cycle in second, $T_c = 60/72 = 0.8333\overline{3}$
- T_s is the systole's period in seconds, about $0.333 \cdot T_c$
- $mod(t, T_c)$ is the remainder of $t$divided by T_c
- I_0 is the maximum amplitude of the blood flow during a systole, obtained using a literature data about the full blood flow in one cardiac cycle, as $90cm^3$.

  $$90 = \int_0^{T_c} I_0 \sin\Big(\pi \cdot \frac{mod(t, T_c)}{T_s}\Big) dt \tag{4.3}$$

  From the equation 4.3 we obtain $I_0 = 424.1mL$

Further elements could be added to this simple model obtaining a more realistic scenario, for the three and four elements Windkessel the report

4.2 Analytical solution of the *2-elements* model

As described before the whole cardiac cycle can be subdivided in two different phases, the systolic and the diastolic ones. The first one leads to an non-homogeneous differential equation while the latter to an homogeneous solution considering as zero the blood flow inside the aorta.

4.2.1 Systolic phase

The systolic phase is characterized by the ventricular contraction that gives motion to fluid. The inhomogeneous solution can be written as

$$I(t) = \frac{P(t)}{R} + C\frac{dP(t)}{dt} \qquad I(t) \neq 0 \tag{4.4}$$

as done in [10] we use an integrating factor $u(t)$ as

$$u(t) = \int \frac{1}{RC} dt = e^{\frac{t}{RC}} \tag{4.5}$$

Writing the inhomogeneous form with $I(t) = I_0 \sin\left(\frac{\pi t}{T_s}\right)$ we can write

$$\frac{P(t)}{CR} e^{\frac{t}{RC}} + \frac{dP(t)}{dt} e^{\frac{t}{RC}} = \frac{I_0}{C} \sin\left(\frac{\pi t}{T_s}\right) e^{\frac{t}{RC}} \tag{4.6}$$

It's obvious that

$$\frac{P(t)}{CR} e^{\frac{t}{RC}} + \frac{dP(t)}{dt} e^{\frac{t}{RC}} = \frac{d}{dt}\left(e^{\frac{t}{RC}} P(t)\right) \tag{4.7}$$

Now we can integrate, in time, both sides writing the two terms of the equation 4.6 the as described by the latter equation.

$$\int d(e^{\frac{t}{RC}} P(t)\Big) = \int \frac{I_0}{C} \sin\left(\frac{\pi t}{T_s}\right) e^{\frac{t}{RC}} dt \tag{4.8}$$

and the solution is :

$$\boxed{y(t) = c_1 e^{\frac{-t}{RC}} + \frac{-e^{\frac{t}{RC}} T_s I_0 R\Big(C\pi R \cos(\frac{\pi t}{T_s}) - T_s \sin(\frac{\pi t}{T_s})\Big)}{T_s^2 + (c\pi R)^2}} \tag{4.9}$$

to evaluate the constant c_1 we impose the initial condition, considering that the pressure at the start of the systolic (T_{ss}) cycle is the pressure at the end of the diastolic one. So, for $t = 0$ the pressure $P(0) = P_{ss} = 80$.

$$c_1 = P_{ss} + \frac{T_s I_0 R\Big[C\pi R \cos(\frac{\pi(t-t_{ss})}{T_s}) - T_s \sin(\frac{\pi(t-t_{ss})}{T_s})\Big]}{T_s^2 + (c\pi R)^2} e^{\frac{(t-t_{ss})}{RC}} \tag{4.10}$$

which give us

$$c_1 = P_{ss} + \frac{T_s I_0 R\Big[C\pi R\Big]}{T_s^2 + (c\pi R)^2} \tag{4.11}$$

4.2.2 Diastolic phase

The omogeneous equation for the dyastolic phase is

$$\frac{P(t)}{R} + C\frac{dP(t)}{dt} = 0 \tag{4.12}$$

and the solution is

$$P(t) = ce^{\frac{-t}{RC}} \tag{4.13}$$

As done for the previous section the constant c can be computed imposing the pressure at the start of the diastolic cycle, as initial condition, the last value of the preceding systolic one. At time t_{sd} (start of diastolic cycle), $P(t)$ equals the diastolic pressure (P_{sd}).

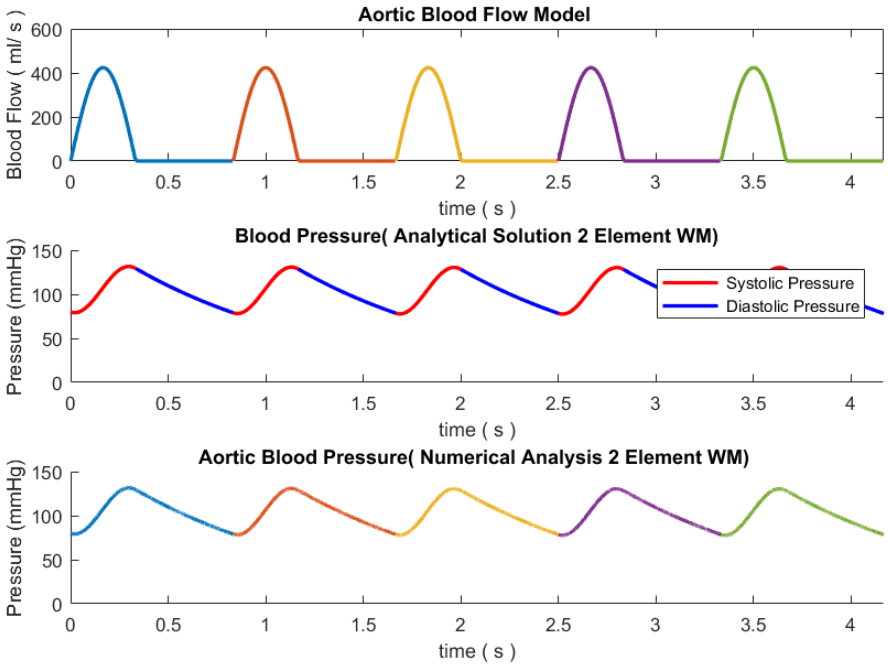


Figure 4.4: Solution from [10]

In figure 4.4 the graph outlined have been obtained via MatLab using the code provided by [10]. The analytical solution was, instead, obtained with ODE23 package from MatLab standard functions. The analytical solution of the 2 elements WM has been also implemented in the code obtaining the solution showed in 4.5.
Figure 4.5 shows the aortic blood pressure for the numerical solution. Note that, as expected, the blood pressure varies between the range of 80-120mmHg. An other test that has been done is to compute the same siulation varying the initial condition. As mentioned in [10], as time progresses, the pressure values reach equilibrium point and converge to form a single curve. Relating this physiologicaly, we can infer whatever perturbation the heart is subjected to (which reflect in high or low pressure fluctuation), it reaches steady state value after a period of time. This feature of the system has been verified as shown

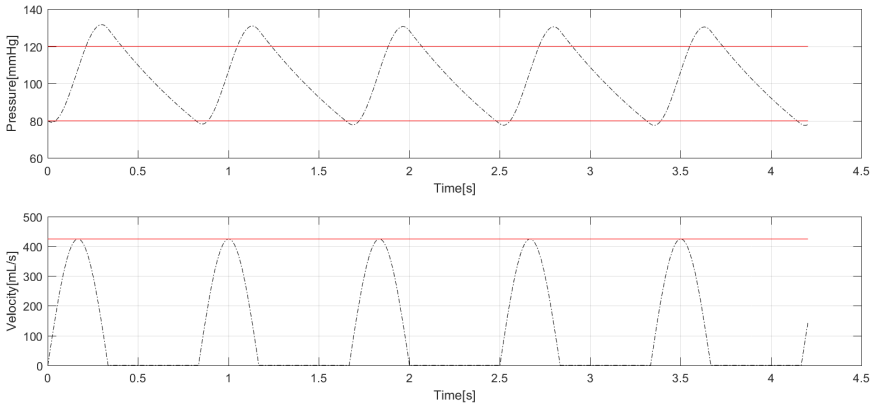


Figure 4.5: Analytical model implemented with the code.

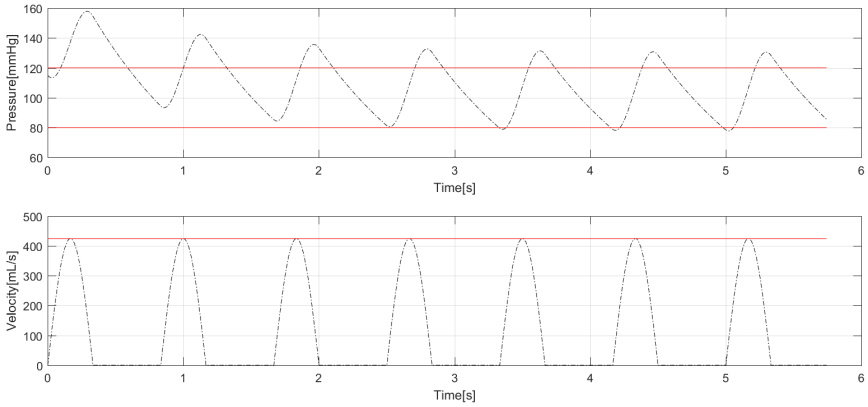


Figure 4.6: Analytical model with a different initial condition $115mmHg$.

in 4.6 where an initial condition equal to $115mmHg$ has been imposed. The model is capable of absorbing the fluctuation in the blood dynamics during the cardiac cycle, and the same results can be seen in [10].

4.3 Numerical Solution

The laws that regulate the phenomena have been described before as analytical form, in particular, the blood flow into the aorta will be always treated as analytically known. To discretize the differential equation we will use, firstly Euler first order discretization for the pressure, and after a Runge kutta method to achieve the second order.

4.3.1 Euler method

The dimpliest method for approximating the solution of equation 4.6 is called Euler method. The basic idea is to obtain an equation that approximate the continuous function at certain point t_i while an initial condition is fixed in time. the variable *time* is taken into account for this approach to the numerical method since our solution will be time-dependent, but this methods and

all the following one can be used also for space discretization equation. For example, starting from an initial condition $y_0 = y(t_0)$ the method will give the value y_1 at time $t_1 = t_0 + \Delta t$. Once the first time step is done, we use the same procedures for the second step and to know the value of the approximated function in t_2.

It's possible to lead to an algorithm formulation for the Euler method starting from different point of view. From calculus we can write the definition of first derivative of a function $y(t)$ at a point $t = a$ with a time step Δt

$$y'(a) = \lim_{\Delta t \to 0} \frac{y(a + \Delta t) - y(a)}{\Delta t} \tag{4.14}$$

So if we compute the the above equation for an enough small time step we have an approximation of $y'(a)$. We know that if the limit exist then both the limit sh $\Delta t \to 0$ from the right and the left must agree. Fixing $\Delta t > 0$, $a = t_0$ and let $t_1 = t_0 + \Delta t$ we obtain

$$y'(t_0) \approx \frac{y(t_1) - y(t_0)}{\Delta t} = \frac{Y_1 - Y_0}{\Delta t} \tag{4.15}$$

Writing our differential equation as $y'(t) = f(t_0, Y_0)$ since it is function of the parameters at time $''0''$. The solution in Y_0 is given as our initial condition, so we can write

$$Y_1 = Y_0 + \Delta t f(t_0, Y_0) \tag{4.16}$$

Once Y_1 is obtained the process can be repeated to obtain the same equation for Y_2. In a more general form we can write

$$Y_{i+1} = Y_i + \Delta t f(t_i, Y_i) \tag{4.17}$$

This form is known as **forward Euler method**, the term forward is used in the name because we wrote the equation at point $t_i$and difference forward in time to t_{i+1}. Now we want to see if we get the same difference equation when we let $h \to 0$ through value less than 0 in the general limit wrote as equation 4.14. In this case $a + h$ lies to the left of a, so we will use a secant line passing through $(a, y(a))$ and a points to its left to approximate $y'(a)$.

$$y'(t_1) \approx \frac{y(t_0) - y(t_1)}{-\Delta t} \tag{4.18}$$

which lead to

$$\frac{Y_1 - Y_0}{\Delta t} = f(t_1, Y_1) \tag{4.19}$$

Where we have used the fact that $t_0 - t_1 < 0$, so in a more general way we can write

$$y_{y+1} = Y_i + \Delta t f(t_{i+1}, Y_{i+1}) \tag{4.20}$$

This method is known as **backward Euler method**,that is because we are writing the equation at t_{i+1} and the difference is made backward in time. The forward Euler scheme is an **explicit** scheme because we can write the unknown value as a function ok known values whereas the backward scheme is called an

implicit scheme because the unknown value is written implicity in terms on known values and itself.

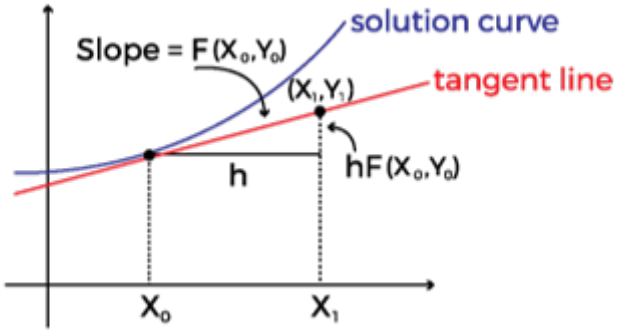


Figure 4.7: Euler graphical method

However, in practice we will use explicit method for the convergence test and for all the further application. Let's write the Eulerian discretization for our problem. Starting from the differential equation we wrote as 4.1 in the most general

$$\frac{dP(t)}{dt} = \frac{I(t)}{C} - \frac{P(t)}{RC} \tag{4.21}$$

In this particular case

$$P(t^{k+1}) = P(t^k) + \Delta t \cdot F(t^k, P(t^k)) \tag{4.22}$$

where

$$F(t^k, P(t^k))) = \frac{I(t^k)}{C} - \frac{P(t^k)}{RC} \tag{4.23}$$

and ordering the last equation we obtain the following expression as

$$\boxed{P(t^{k+1}) = P(t^k)\Big(1 - \frac{\Delta t}{RC}\Big) + \frac{\Delta t}{C} I(t^k)} \tag{4.24}$$

Results Euler method

As view before in section 3.3 the convergence order has been verified since the analytical solution is known. The resistence and the capacitor characteristics are the same outlined before in the paragraph 4.1 has been resumed in table 4.1

Three time steps has been simulated in order to calculate the convergence order of the method and only one cardiac cycle results enough for convergence order study. The difference between the two solution, the error value has been computed and it's plotted in 4.9b. For first we can appreciate the same behaviour for both three curves, obviously such error is reduced each time steps refinement. In particular, as discussed in paragraph 3.3.1, the error results divided by two if the time step is half of the previous one. Such behaviour advice, as expected, a first order convergence as possible to see in 4.9a

Table 4.1: Features of the 2 Elements Windkessell model

$R =$	$0,95$
$C =$	$1,0666$
$T =$	$0,8333$
$T_s =$	$0,33332$
$T_d =$	$0,49998$
$I_0 =$	$424,1$

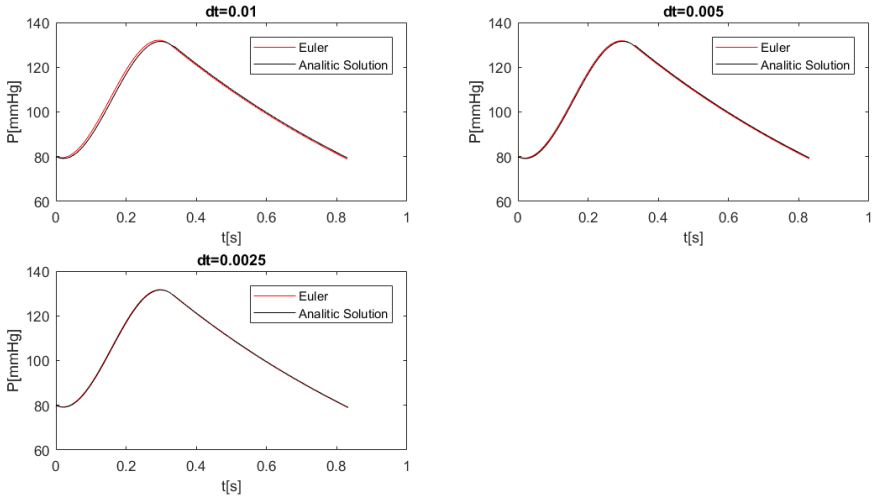
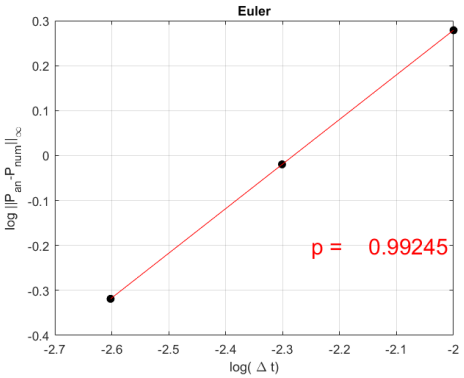
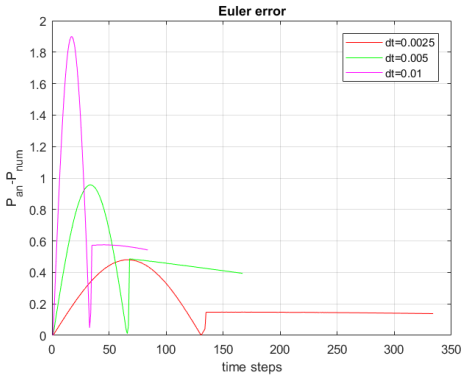


Figure 4.8: Different time steps compared with the respective analytical solution.



(a) Rate of convergence, Euler method using three time steps.



(b) Errors in the three time steps configuration.

4.3.2 Runge-Kutta II

The Euler method, as we saw before is simple and intuitive and gives an approximation that is as close to the analytical solution as desiderata. However, the accuracy improves only *linearly* with the time step, that means that it takes 10 times as many steps to achieve an approximation 10 times as accurate. With some little adjustment the method could be more efficient giving an accuracy that improves *quadratically*, in other words we are looking for a second order method. Let's use an integral point of view to solve a differential equation. If ϕ is a smooth function on an interval that contains the point t_i and t_{i+1} then the Fundamental Theorem allows us to write

$$\phi(t_{i+1}) = \phi(t_i) + \int_{t_i}^{t_{i+}} \phi'(u)du \tag{4.25}$$

In other words we can approximate the function ϕ at the point t_{i+1} having just information about ϕ at the point t_i and approximating the integral. For example, we can approximate ϕ' on the interval we used before, using the value at the left endpoint. This is known as Euler Method.
On the other hand the Heun's Method uses the trapezoids instead of rectangle as shown in fig. 4.10. In this case the mathematical expression to predict the value of the ϕ' at the next time step is

$$\phi(t_{i+1}) \approx \phi(t_i) + \frac{h}{2}(f(t, \phi(t_i)) + f(t, \phi(t_{i+1})) \tag{4.26}$$

The problem is that the value we want to approximate , appears on the right-hand side. We use the Euler Method to approximate the value $\phi(t_{i+1})$,

$$\tilde{y}_{i+1} = y_i + hf(t_i, y_i) \tag{4.27}$$

after, we use this value in the right-hand side of the equation 4.26

$$y_{i+1} = y_i + \frac{h}{2}(f(t, y(t_i)) + f(t, \tilde{y}_{i+1})) \tag{4.28}$$

The method shown how to predict the value of the function at the next time step using a graphical approach. As seen this method needs to two different

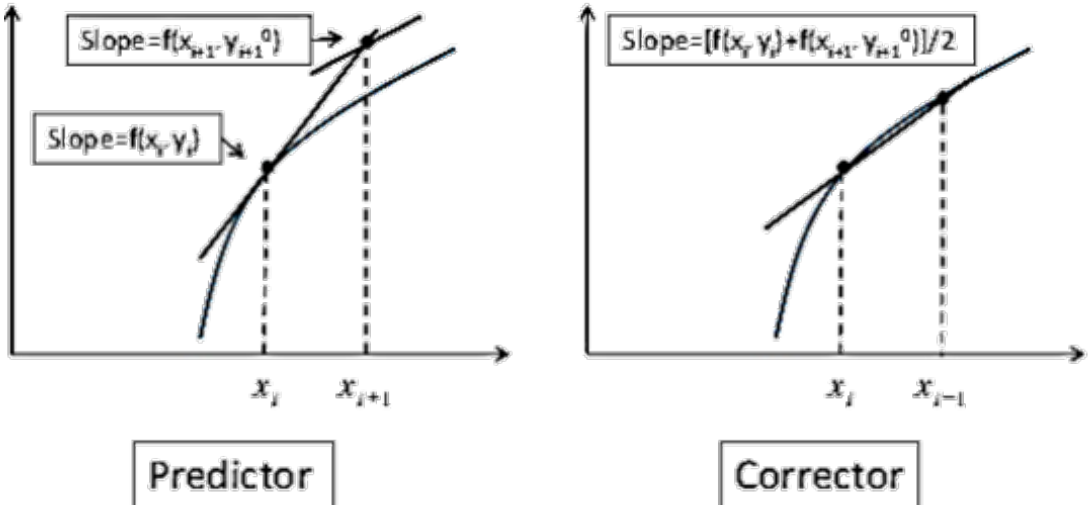


Figure 4.10: Heun graphical method

stage to get the value. That's the reason why such method is also called two-stage second-order Runge-Kutta methods. Once described the general numerical method we are ready to adapt it to our ODE. Starting from the expression of the 4.21 we can write the following system

$$p^{(1)} = p^k + \Delta t \cdot F(t^k, p^k) \tag{4.29}$$

$$p^{k+1} = \frac{1}{2}p^k + \frac{1}{2}p^{(1)} + \Delta t \cdot F(t^{k+1}, p^1) \tag{4.30}$$

Using the first equation in the second one,

$$p^{k+1} = p^k + \frac{\Delta t}{2}\left[F(t^k, p^k) + F(t^{k+1}, p^k + \Delta t \cdot F(t^k, p^k))\right] \tag{4.31}$$

$$p^{k+1} = p^k + \frac{\Delta t}{2}\left[\frac{I(t^k)}{C} - \frac{P(t^k)}{RC} + \frac{I(t^{k+1})}{C} - \frac{p^k + \Delta t(\frac{I(t^k)}{C} - \frac{p^k}{RC})}{RC}\right] \tag{4.32}$$

Organizing the different terms we finally obtain:

$$\boxed{p^{k+1} = p^k \cdot \left[1 - \frac{\Delta t}{RC} - \frac{1}{2}\left(\frac{\Delta t}{RC}\right)^2\right] + \frac{\Delta t}{2C} \cdot \left[I(t^k) + I(t^{k+1})\right] - \frac{\Delta t^2}{2RC^2} I(t^k)} \tag{4.33}$$

Results Runge-Kutta II method

Using the data coming from the analytical solution the Runge-Kutta method convergence order has been computed. For this rate of convergence evaluation a constant for the non-homogeneous therm. Firstly the constant use has been zero. The results obtained are outlined below.

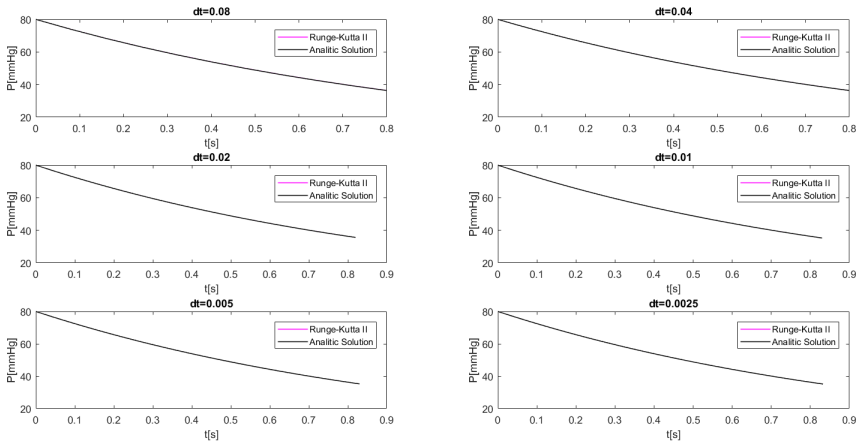
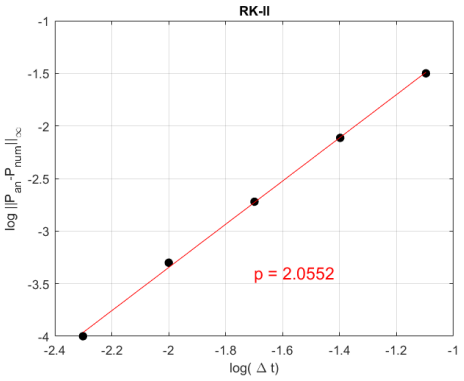
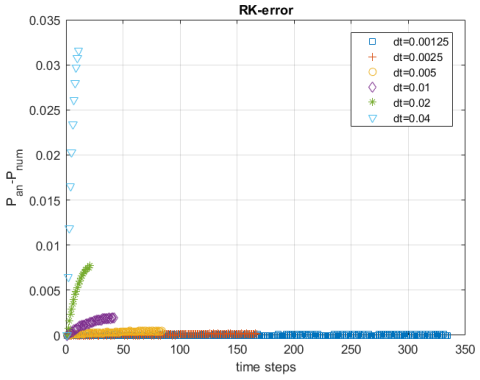


Figure 4.12: Different time steps of the Runge Kutta II, $I = 0$, compared with the respective analytical solution.



(a) Rate of convergence, RK-II method using three time steps with $I = 0$.



(b) Errors in the different time steps configuration for the Runge Kutta II method with $I = 0$.

As possible to see from 4.11a the order achieved is 2 according to the mathematical model. The graph outlined in Figure 4.11b shows a constant decreasing error. Dividing by two the time step used the error results divided by 4. A further test, using a non-zero constant for the non-homogeneous term has been performed achieving a second order using the same time step of the previous test.

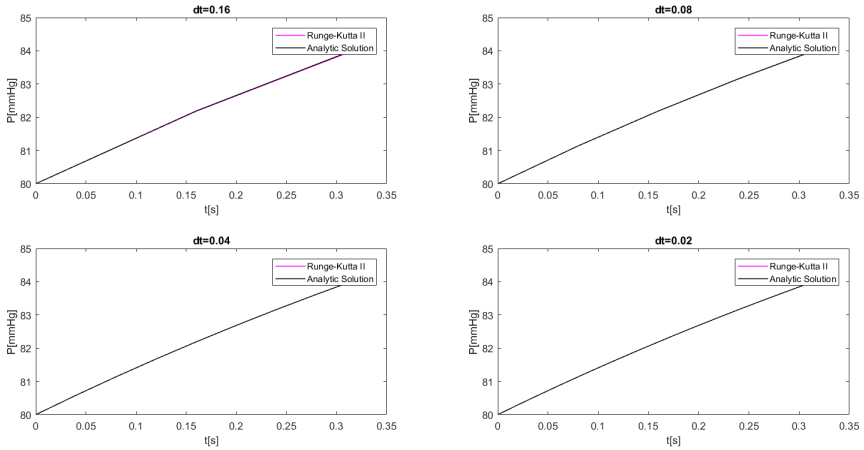
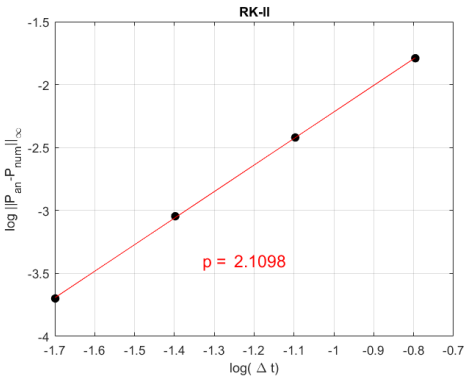
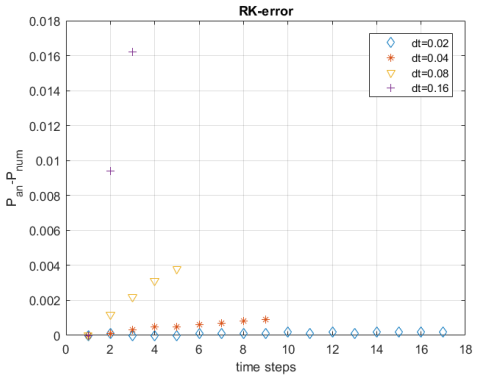


Figure 4.14: Different time steps compared with the respective analytical solution.



(a) Rate of convergence, RK-II method using three time steps with $I = 100$.



(b) Errors in the different time steps configuration for the Runge Kutta II method with $I = 100$.

Chapter 5

2D Aorta Simulation

In this chapter will be presented simulations and tests of the developed code including the Windkessel model applied to the boundaries. The work done is mainly over simple geometry without involving particular structures or moving body. Moreover, two-dimensional Aorta was not available as default geometry like the cylinder we discussed before and for preliminary tests has been retained to validate the model over simple geometry since, a real one needs , to be performed, greater computational cost. Firstrly a T shaped duct has been taken in account as shown in 5.4a, this geometry does not requires level set function to define a new geometry. An Y shaped aorta was requested as more realistic test case, so the horizontal duct created before for the T shaped geometry has not been modified while the right half zone of the domain is obtained by overlapping several spacial disequation by-passing the input file, the result obtained is shown in 5.2. That geometry has been built using two square imposed from the input file and a geometry defined by a level set. This kind of simple aorta's geometry has been adopted to test the model to predict the pressure on the boundary and to show some initial results. In figure 5.1 can be observed that the fluid domain is represented as the maximum value of a function that only measure the distance from the generic cells to the nearest wall. Obviously such function allows us also to define the fluid zone or the solid ones giving appropriate penalizzation values to each cell , positive if the cell belongs to the fluid part, and viceversa for the solid ones.
Moreover, since the geometry used to define the smaller vessel are not defined in the code a priori but implemented as a level set, unfortunately it's not provided a de-refinement process inside the solid zone. As showed in 5.3 a function called "*CleanInsolid*" provides a local de-refinement paying attention to avoid a double difference level between two consecutive cells.

5.0.1 Inflow boundary condition

Before having the vessel geometry showed in fig. 5.2 the simple T shaped Aorta showed in fig. 5.4a has been simulated in order to verify the correct and consistent work of the Navier-Stokes code. A first test has been the Flow rate in the three different duct that the geometry is composed by. The results obtained are shown in 5.4b.
Moreover, for the Inflow boundary condition, to give as most realistic as possible behaviour of the fluid inside the aorta, a Poseuille flow has ben imposed.

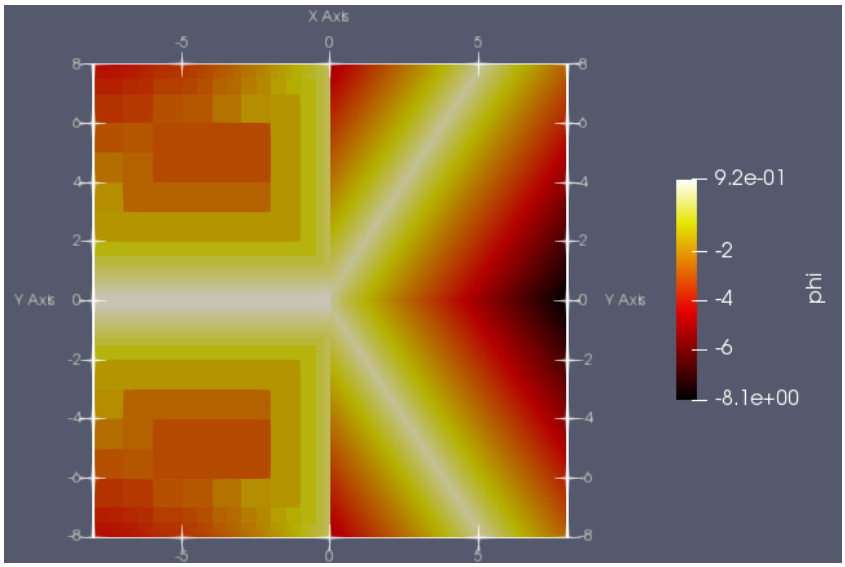


Figure 5.1: Level set that allows the presence of the wall for the Y shaped Aorta.

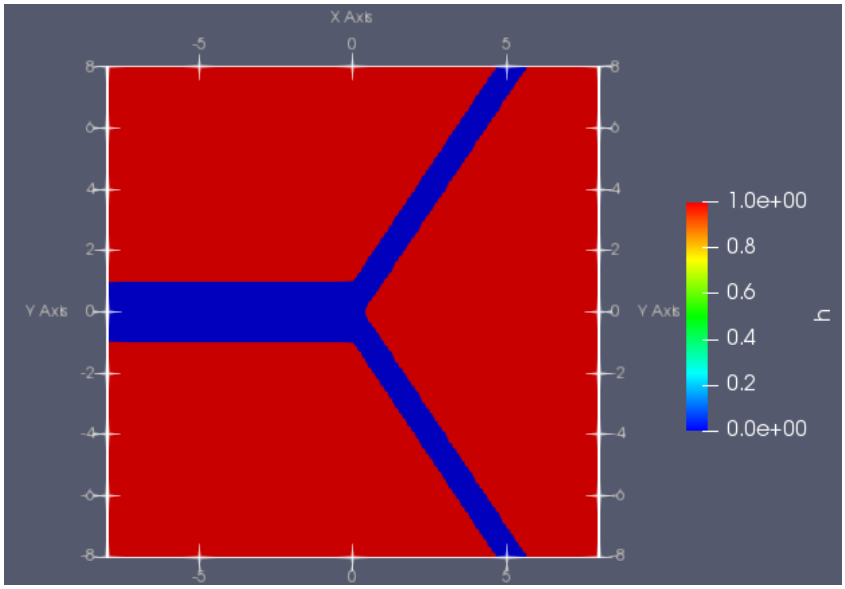


Figure 5.2: Penalizzation cells value used to define solid and fluid cells.

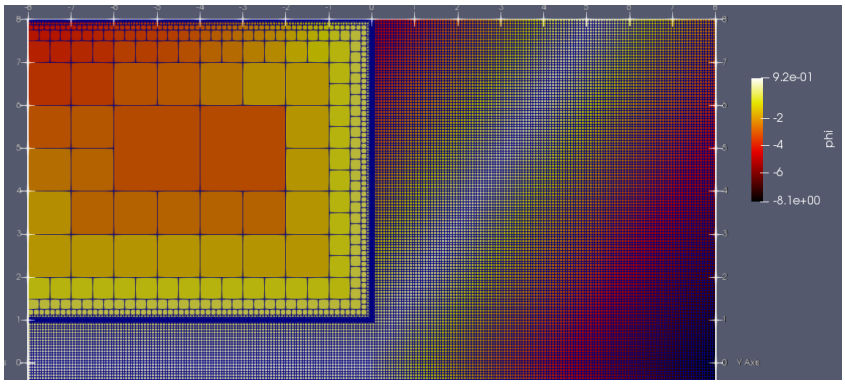
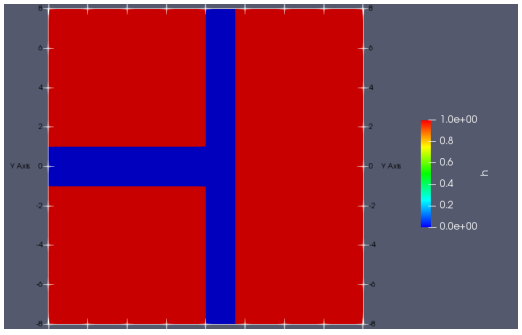
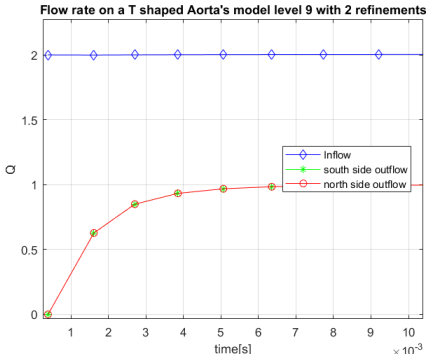


Figure 5.3: Half Aorta's grid with a de-refinement in the solid zone identified as a square given as input



(a) Level set on the domain of the T shaped Aorta.



(b) Flow rate evaluated in the three vessel of T-shaped aorta's model

Such velocity profile needs to be time dependent to simulate the oscillating velocity profile outlined in [10]. For that kind of simulation there aren't any indication about the shape of the inflow since it's an 1D simulation so, a parabolic profile has been created as shown in fig. 5.5. The profile has been crated starting from the coordinate of the fluid cell on the left boundaries following the law

$$f(y) = 1 - y^2 \tag{5.1}$$

The duct width goes from -1 to 1, and the profile velocity is equal to 0 at the wall of the first horizontal part of the Aorta. The timing of the pulsatile inflow respects the periods given by the [10].

5.0.2 Outflow flow rate

The flow rate in correspondence of the boundary of the domain has been computed to have information about the flow behaviour in the little vessel since the left boundary is characterized by an oscillating boundary condition. For first the flow rate has been applied to a Y shaped aorta's model to observe the velocity field fluctuation. Also if the velocity on the left side is imposed equal to zero during the systolic phase, in the little vessel the flow has a proper non-zero velocity.

5.0.3 Windkessel code integration

Evaluated the method to discretize and implemented the Windkessel model such method as been adapted and interfaced with the code. The U^* velocity is the velocity predicted after the prediction step and such value has been an input to the adopted model. When the pressure is evaluated the relative pressure map is updated with the new value. Once the P^* has been collected, the corrected pressure as output from the Windkessel model this value is inserted as right Hand side of the projection step. In figure 5.7 is shown all the work flow and the integration inside the NS equations.

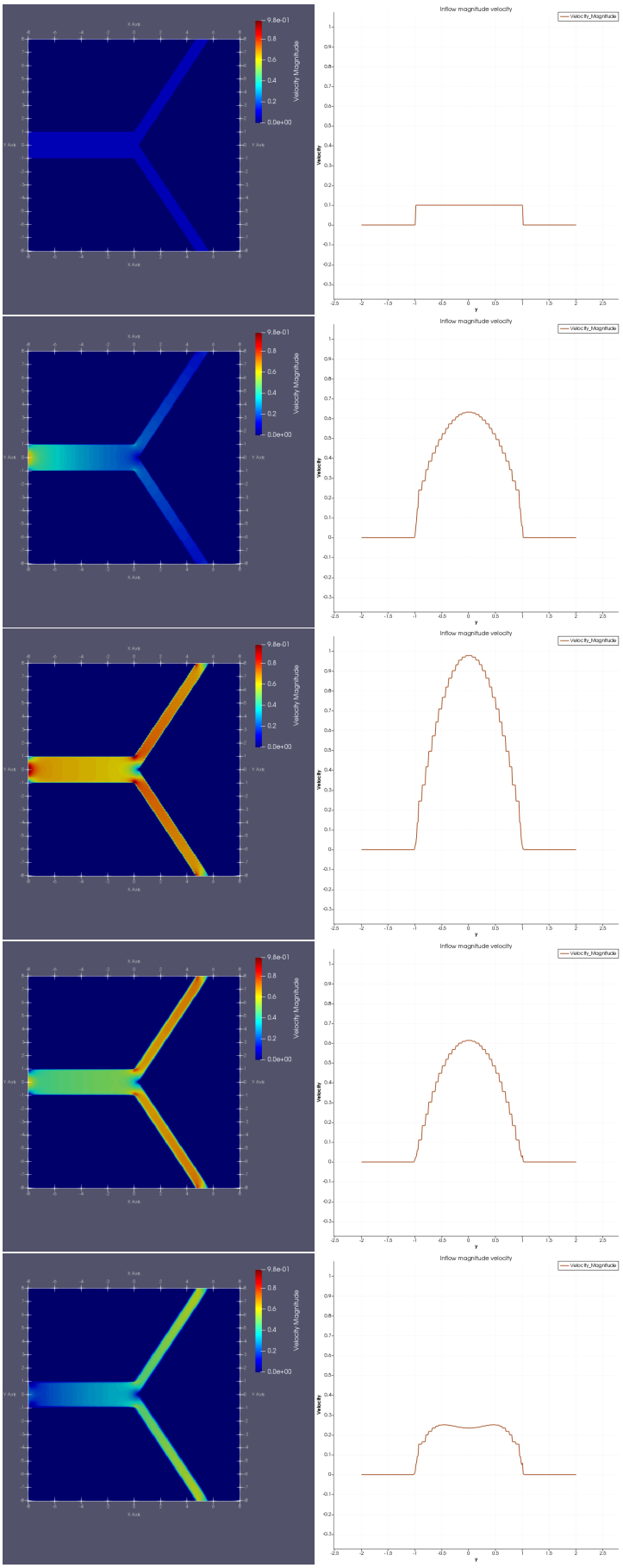


Figure 5.5: Velocity Inflow profile for 4 different time step

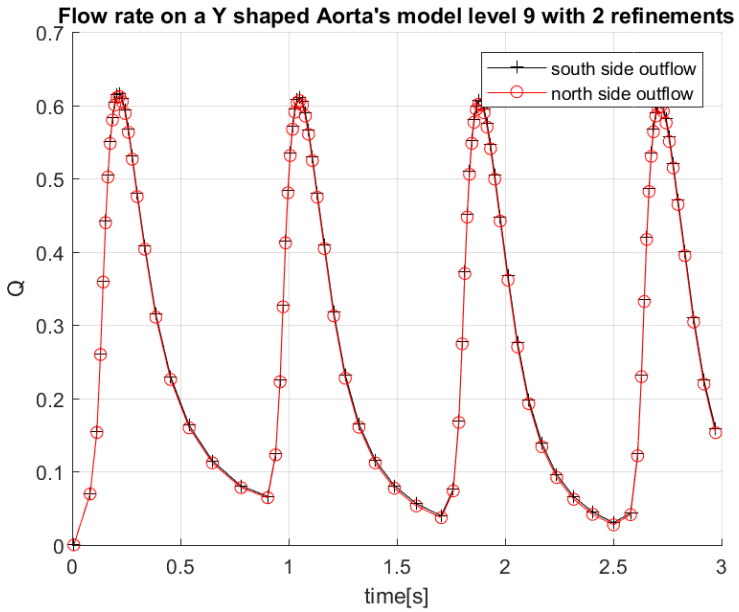


Figure 5.6: Outflow flow rate

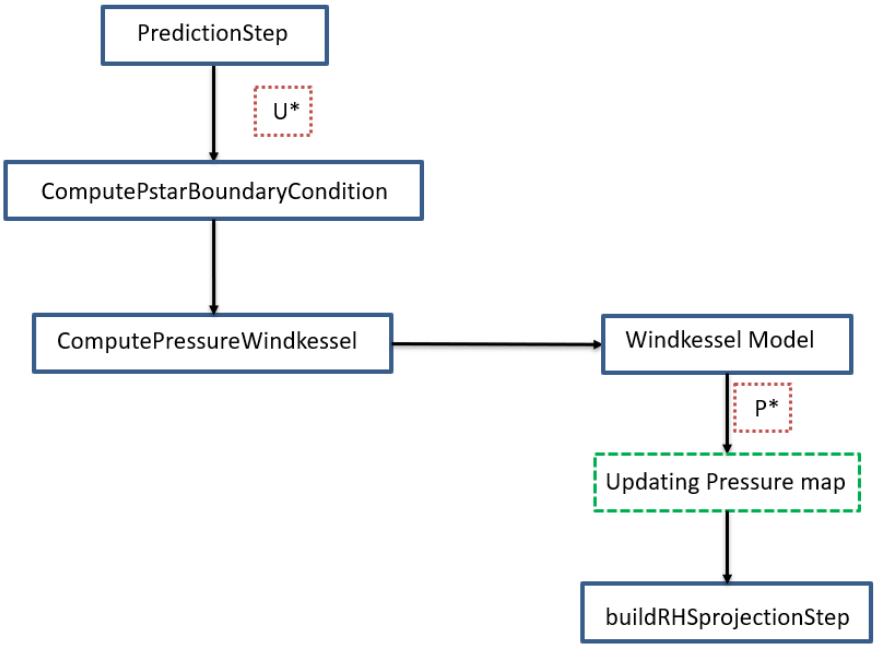


Figure 5.7: Flowchart of the function used to integrate the algorithm in the code.

Chapter 6

Future developments

In the following lines will be outlined some improvement and changes that can be done in order to increase the stability and the precision of the solution. A trade off considering time costs will be necessary to find the optimum solution.

6.0.1 Advection term discretizzation

Upwind schemes use an adaptive or solution-sensitive finite difference stencil to numerically simulate the direction of propagation of information in a flow field. The code used during this thesis project uses, to discretize Navier-stokes advection term, an upwind discretizzation of first order accuracy in space and time. This kind of scheme introduces severe numerical diffusion in the solution where larger gradient accurs.Considering one-dimensional linear advection equation the term a represent the propagation speed of the signal. The upwind scheme is stable if the CFL condition is satisfied.

$$c = |\frac{a\Delta t}{\Delta x}| \leq 1 \tag{6.1}$$

The first improvement that could be done is an implementation of a method with more spacial accuracy.

6.0.2 Windkessel model

In order to have a better understanding of the fluid behaviour in a more realistic case it's necessary introduce a model that takes into account several vessel behaviour. In order to do that it's required a larger in number Windkessel model where each elements represents the vessel characteristics. In figure 6.1 are shown two different electrical circuit in analogy to aortic flow with more than 2 elements.

Once implemented a more realistic mode that could be applied on a three dimensional duct or, as a final results, on an realistic Aorta's model. The code allows users to utilize as geometry STL files given as input. Using a scan technique can be used real patients Aorta as geometry where the blood must flows inside as reported in 6.2.

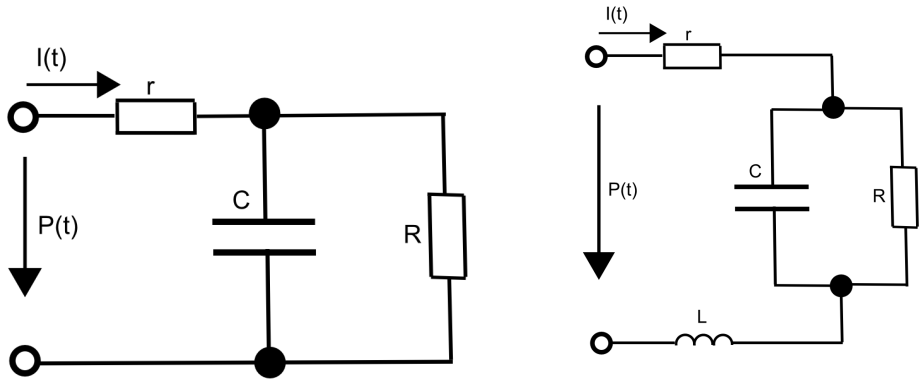


Figure 6.1: Windkessel model with 3 or 4 elements

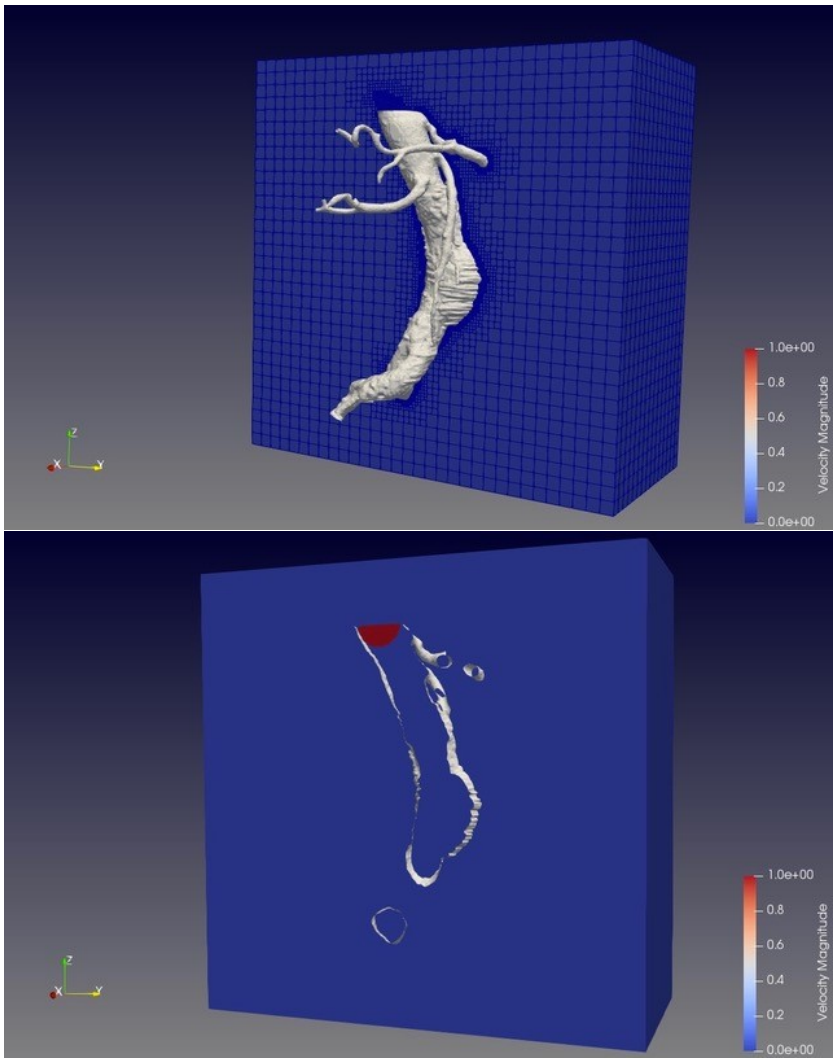


Figure 6.2: Domain section of a three dimensional aorta's recostruction using CAT technique

6.0.3 Least Square method

Actually the improvement given to the code gives more precision to forces evaluation over a generic body. As saw in the chapter of the forces the gradient, even if more accurate, gives a first order accuracy in space. The next step could be an extension of the interpolation method to a second order accuracy for the gradient variable.

Acknowledgments

Il presente progetto di tesi è stato stato svolto in collaborazione con il centro di ricerca INRIA di Bordeaux, dunque ci tengo a ringraziare in primis coloro che hanno reso possibile questa collaborazione, il mio relatore, Professor Domenic D'ambrosio e le persone che mi hanno seguito durante l'esperienza all'estero, il mio correlatore ing. Florian Bernard e il professore a capo dell' equipe MEMPHIS, Angelo Iollo.

Inoltre, è doveroso effettuare un grande ringraziamento al mio collega, nonchè compagno d'avventura, Marco, per il suo sostengo e per avermi trasmesso le sue conoscenze e le sue passioni nel corso della nostra convivenza a Bordeaux.

Ringrazio la mia famiglia, per la quale un encomio non basterebbe per potermi sdebbitare del sostegno e dell'incoraggiamento ricevuto nei momenti di stress che contornano la vita di ogni studente universitario. In particolare, i miei fratelli, Francesco e Maria Chiara che possano sempre giorire ed essere fieri di me e dei miei traguardi, spero di poter ripagare ripagare il loro affetto con la mia esperienza. A mio padre, il mio consigliere personale, nonchè affettuosa e premurosa presenza nella mia vita che mai è stata priva di sue attenzioni. A mia madre, il mio faro di speranza e la mia ancora di salvezza da sempre, che tu possa sempre avere la forza di gioire dei miei successi che sono anche i tuoi.

Ringrazio i miei colleghi nonché amici e fratelli scelti, per avermi fatto sentire a casa anche se non lo ero durante questi anni da fuorisede.

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

Bibliography

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

[2] 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.

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

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

[5] 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)

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

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

[8] 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.

[9] 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).

[10] M. Catanho, M. Sinha and V. Vijaya, Model of Aortic Blood Flow Using the Windkessel Effect. BENG 221 - Mathematical Methods in Bioengineering. Report, October 25,2012.

[11] https://vintilablog.wordpress.com/2015/02/27/despre-anevrismul-aortei-abdominale/

[12] https://commons.wikimedia.org/wiki/File:SystolevsDiastole.png

[13] Byoung-Kwon Lee, MD, Computational Fluid Dynamics in Cardiovascular Disease, Division of Cardiology, Department of Internal Medicine, Gangnam Severance Hospital, Yonsei University College of Medicine, Seoul, Korea.

[14] https://en.wikipedia.org/wiki/Loglog-plot

List of Figures

List of Tables