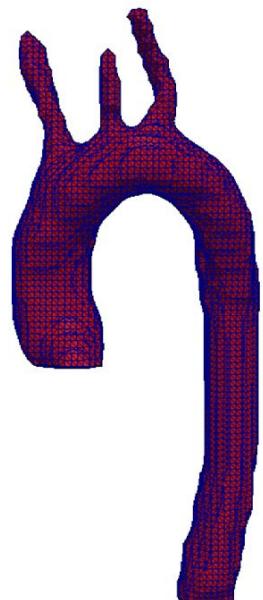


# **Hemodynamics in the Thoracic Aorta using OpenFOAM: 4D PCMRI versus CFD**

J. Casacuberta  
E. Soudah  
P. J. Gamez-Montero  
G. Raush  
R. Castilla  
J.S. Perez



# Hemodynamics in the Thoracic Aorta using OpenFOAM: 4D PCMRI versus CFD



1. School of Industrial and Aeronautic Engineering of Terrassa. Department of Fluid Mechanics,  
LABSON, Universitat Politècnica de Catalunya, Campus Terrassa, Terrassa, Spain

Register for free at <https://www.scipedia.com> to download the version without the watermark  
2. International Center for Numerical Methods in Engineering (CIMNE)  
Computational Engineering Group, Universitat Politècnica de Catalunya, Barcelona, Spain

Monograph CIMNE N°-154, March 2015

**International Center for Numerical Methods in Engineering**  
Gran Capitán s/n, 08034 Barcelona, Spain



Register for free at <https://www.scipedia.com> to download the version without the watermark

INTERNATIONAL CENTER FOR NUMERICAL METHODS IN ENGINEERING  
Edificio C1, Campus Norte UPC  
Gran Capitán s/n  
08034 Barcelona, Spain  
[www.cimne.com](http://www.cimne.com)

First edition: March 2015

**HEMODYNAMICS IN THE THORACIC AORTA USING OPENFOAM: 4D PCMRI versus CFD**  
Monograph CIMNE M154  
© Los autores

ISBN: 978-84-943928-0-1  
Depósito legal: B-9283-2015

---

# Content

|  |           |
|--|-----------|
| <b>Content</b>   | <b>i</b>  |
| <b>1 Introduction</b>  | <b>1</b>  |
| 1.1 Aim of the work . . . . .  | 1         |
| 1.2 Justification of the work . . . . .                                    | 1         |
| 1.3 Scope of the work . . . . .  | 2         |
| 1.4 Requirements of the work . . . . .                                     | 2         |
| 1.5 Objectives . . . . .   | 3         |
| <b>2 Theoretical foundations</b>   | <b>4</b>  |
| 2.1 About fluid mechanics . . . . .  | 4         |
| 2.2 About numerical methods . . . . .                                      | 7         |
| 2.3 Description of OpenFOAM . . . . .                                      | 7         |
| 2.4 Overview of characteristics of the thoracic aorta . . . . .            | 8         |
| 2.4.1 The circulatory system . . . . .                                     | 8         |
| 2.4.2 Anatomy of the aorta . . . . .                                       | 9         |
| 2.4.3 Blood as a fluid . . . . .   | 10        |
| 2.5 Relevance of the wall shear stress in medical research . . . . .       | 11        |
| 2.6 4D flow visualization method used for comparison . . . . .             | 13        |
| 2.7 State of the art . . . . .   | 15        |
| 2.7.1 Learning OpenFOAM . . . . .  | 15        |
| 2.7.2 Thoracic aorta studies and hemodynamics CFD simulations . . . . .    | 17        |
| 2.8 Hypotheses used for aorta simulation . . . . .                         | 18        |
| <b>3 Pre-processing of the aorta with OpenFOAM</b>                         | <b>20</b> |
| 3.1 Preparing the geometry . . . . .                                       | 22        |
| 3.1.1 The 3 simulated geometries . . . . .                                 | 22        |
| 3.1.2 Smoothing the surface . . . . .                                      | 25        |
| 3.1.3 Surface feature extracting . . . . .                                 | 28        |
| 3.1.4 Meshing the geometry . . . . .                                       | 28        |
| 3.1.5 Refining the mesh to compute an accurate wall shear stress . . . . . | 32        |
| 3.1.6 Defining patches for the simulation . . . . .                        | 34        |
| 3.2 Boundary conditions . . . . .  | 41        |
| 3.3 Physical properties . . . . .  | 50        |

Register for free at <https://www.scipedia.com> to download the version without the watermark

## CONTENT

---

|   |            |
|---|------------|
| 3.4 Control, discretization and linear-solver setting . . . . .   | 51         |
| 3.5 Justification of the mesh refinement . . . . .  | 51         |
| <b>4 Results of the aorta simulation</b>  | <b>54</b>  |
| 4.1 Module of the velocity field . . . . .  | 54         |
| 4.2 Streamlines . . . . .   | 59         |
| 4.3 Vector field . . . . .  | 60         |
| 4.4 Wall shear stress . . . . .   | 62         |
| <b>5 Analysis of the results</b>  | <b>65</b>  |
| 5.1 Analysis of the OpenFOAM simulation . . . . .   | 65         |
| 5.1.1 Validation of results . . . . .   | 65         |
| 5.1.2 Discussion of results . . . . .   | 67         |
| 5.2 Comparison between the CFD OpenFOAM simulations and the 4D medical images . . . . .                               | 68         |
| 5.3 Analysis of the wall shear stress . . . . .   | 73         |
| 5.3.1 Comparison of results according to OpenFOAM and the 4D method . . . . .   | 73         |
| 5.3.2 Analysis of wall shear stress results depending on the spatial discretization used for the simulation . . . . . | 77         |
| 6 Conclusions   | 84         |
| 6.1 Further steps . . . . .   | 86         |
| <b>A Appendix</b>   | <b>87</b>  |
| A.1 surfaceFeatureExtractDict . . . . .   | 87         |
| A.2 blockMeshDict . . . . .   | 87         |
| A.3 boundary . . . . .  | 89         |
| A.4 snappyHexMeshDict . . . . .   | 90         |
| A.5 checkMesh . . . . .   | 98         |
| A.6 topoSetDict . . . . .   | 102        |
| A.7 createPatchDict . . . . .   | 104        |
| A.8 Script for patch definition . . . . .   | 107        |
| A.9 p . . . . .   | 112        |
| A.10 U . . . . .  | 113        |
| A.11 transportProperties . . . . .  | 115        |
| A.12 RASProperties . . . . .  | 116        |
| A.13 controlDict . . . . .  | 116        |
| A.14 fvSchemes . . . . .  | 117        |
| A.15 fvSolution . . . . .   | 118        |
| <b>Bibliography</b>   | <b>120</b> |

---

# 1. Introduction

## 1.1 Aim of the work

The aim of this work is the study of fluid dynamics models using the CFD software OpenFOAM, an open source software allowing meshing, manipulation, simulation and post-processing of many problems involving fluid mechanics.

The work consists of a study with OpenFOAM of a real engineering problem, namely to analyze hemodynamics in the thoracic aorta in collaboration with CIMNE (Centre Internacional de Mdes Numcs en Enginyeria) and LABSON-UPC (Laboratorio de Sistemas Oleohidricos y Neumcos). Specifically, the study aims to compute the shear stress that blood causes to aorta walls.

## 1.2 Justification of the work

Register for free at <https://www.scipedia.com> to download the version without the watermark

OpenFOAM allows operation and manipulation of fields, and, more specifically, fluid fields. As an open source software, OpenFOAM offers users complete freedom to customize and extend its existing functionality. However, there does not exist any freely available official document with examples for the main utilities and explanations of how to deal with the great amount of possibilities that OpenFOAM can offer. The users guide provided by OpenFOAM contains information about the main controls and solvers, but without showing clear cases allowing users to understand how to set them. By way of example, the OpenFOAM users guide enumerates the resolution codes that exist to simulate a flow depending on its nature (incompressible laminar, incompressible turbulent) but it does not show how to program them. Users have to use examples done by others in order to configure their own cases and take profit of the OpenFOAM forums created by the community.

This makes it difficult for beginners to introduce themselves to the software, understand its controls, and figure out how to set cases properly depending on their nature. Consequently, the guide developed in this work is conceived to show clear-cut examples of representative cases, help understanding how to program the required

## 1. Introduction

steps, and present useful tools for pre-processing, resolution and post-processing. As the author has recently been introduced to OpenFOAM himself, in the appendix section will count with the point of view of a fresh beginner, thus reflecting his understanding of what kind of problems arise in the first steps of an OpenFOAM user and how to deal with them. This section pretends to become a starting point for new learners.

The aorta is the main artery in the human body, distributing blood directly from the heart. As a consequence, there is much medical interest in understanding the dynamics of blood passing through it. The increasing amount of available techniques and CFD software allows enhanced treatment and simulation, attempting to analyze the influence of hemodynamics on medical diseases.

CIMNE has carried out studies with medical images obtained from patients by magnetic resonance techniques. However, it is necessary to validate their data, as well as to determine how noise affects measurements. The study with OpenFOAM will provide contrast data. Comparison between real measures and numerical simulation will offer an interesting approach and relevant conclusions.

The main goal of the study is to determine the shear stress that blood causes to aorta walls. This is a direct factor in the accumulation of atheroma plates in vessels, which may cause detachments and therefore embolism. Since it is very difficult to obtain wall shear stress values experimentally, CFD analysis will contribute to its

assessment and therefore to a better understanding of the human body.

### **1.3 Scope of the work**

- Simulate the hemodynamics of blood in a real aorta geometry
- Obtain with OpenFOAM an accurate distribution of the wall shear stress that blood causes to aorta walls
- Compare and contrast medical images with OpenFOAM simulations

### **1.4 Requirements of the work**

- Simulate blood flow through a real aorta geometry in steady state conditions
- Simulate the most critical conditions regarding wall shear stress during a cardiac cycle (peak systolic conditions)

- 
- Consider blood as an incompressible Newtonian fluid
  - Consider a static aorta geometry with rigid and blood-impermeable walls
  - Consider laminar flow conditions
  - Use a real aorta geometry, including the three supra-aortic branches but without the coronary arteries

## 1.5 Objectives

The main objectives are to:

Solve with OpenFOAM the fluid dynamics in a real aorta geometry

Compute accurately wall shear stress caused by blood to aorta walls using OpenFOAM and its meshing tool (*snappyHexMesh*)

Determine interesting flow variables from the point of view of medical analysis

Compare OpenFOAM results with real medical images, especially focusing on their wall shear stress distribution

Register for free at <https://www.scipedia.com> to download the version without the watermark

## 2. Theoretical foundations

### 2.1 About fluid mechanics

The main ideas used to introduce the fluid mechanics field in this project are extracted from [1], a very useful article explaining the evolution with time of fluid mechanics.

The art of fluid mechanics arguably has its roots in prehistoric times when streamlined spears, sickle-shaped boomerangs and fin-stabilized arrows evolved empirically.

The Greek mathematician Archimedes (287–212 BC) provided an exact solution to the fluid-at-rest problem, long before calculus or the modern laws of mechanics were known. Leonardo da Vinci (1452–1519) correctly deduced the conservation of mass equation for incompressible, one-dimensional flows and also pioneered flow visualization. Little more than a century and half after Newton's *Principia Mathematica* was published in 1687, the first principles of viscous fluid flows were affirmed in the form of the Navier-Stokes equations. With very few exceptions, the Navier-Stokes equations provide an excellent model for both laminar and turbulent flows.

Register for free at <https://www.scipedia.com> to download the version without the watermark

Each of the fundamental laws of fluid mechanics, conservation of mass, momentum and energy, are presented next. At every point in space-time and in Cartesian tensor notation, the three conservation laws read as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_k}(\rho u_k) = 0 \quad (2.1)$$

$$\rho \left( \frac{\partial u_i}{\partial t} + u_k \frac{\partial u_i}{\partial x_k} \right) = \frac{\partial \sum k_i}{\partial x_k} + \rho g_i \quad (2.2)$$

$$\rho \left( \frac{\partial e}{\partial t} + u_k \frac{\partial e}{\partial x_k} \right) = -\frac{q_k}{\partial x_i} + \sum k_i \frac{\partial u_i}{\partial x_k} \quad (2.3)$$

where  $\rho$  is the fluid density,  $u_k$  is an instantaneous velocity component ( $u$ ,  $v$ ,  $w$ ),  $\sum k_i$  is the second order stress tensor (surface force per unit area),  $g_i$  is the body force per unit mass,  $e$  is the internal energy per unit mass, and  $q_k$  is the sum of heat

flux vectors due to conduction and radiation. The independent variables are time  $t$  and the three spatial variables  $x$ ,  $y$  and  $z$ .

The fundamental laws of fluid mechanics are listed in their *raw form*, i.e., assuming only that the speeds involved are non-relativistic and that the fluid is continuum. The latter assumption implies that the derivatives of all the dependent variables exist in some reasonable sense. In other words, local properties such as density and velocity are defined as averages over elements large compared with the microscopic structure of the fluid but small enough in comparison with the scale of the macroscopic phenomena to permit the use of differential calculus to describe them.

Equations 2.1, 2.2 and 2.3 constitute five differential equations for the 17 unknowns  $\rho$ ,  $u_i$ ,  $\sum k_i$ ,  $e$  and  $q_k$ . Absent any body couples, the stress tensor is symmetric having only six independent components, which reduces the number of unknowns to 14. To close the conservation equations, relation between the stress tensor and deformation rate, relation between the heat flux vector and the temperature field, and appropriate equations of state, relating the different thermodynamic properties are needed. For a Newtonian, isotropic, Fourier, ideal gas, for example, these relations read:

$$\sum k_i = -p\delta_{ki} + \mu \left( \frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} \right) + \lambda \left( \frac{\partial u_j}{\partial x_j} \right) \delta_{ki} \quad (2.4)$$

$$q_i = -\kappa \frac{\partial T}{\partial x_i} + \text{Heat flux due to radiation} \quad (2.5)$$

Register for free at <https://www.scipedia.com> to download the version without the watermark

$$de = c_v dT \quad \text{and} \quad p = \rho RT \quad (2.6)$$

where  $p$  is the thermodynamic pressure,  $\mu$  and  $\lambda$  are the first and second coefficient of viscosity, respectively,  $\delta_{ki}$  is the unit second-order tensor (Kronecker delta),  $\kappa$  is the thermal conductivity,  $T$  is the temperature field,  $c_v$  is the specific heat at constant volume, and  $R$  is the gas constant. Newtonian implies a linear relation between the stress tensor and the symmetric part of the deformation tensor. The isotropy assumption reduces the 81 constants of proportionality in that linear relation to 2 constants.

The Stokes' hypothesis relates the first and second coefficients of viscosity,  $\lambda + \frac{2}{3}\mu = 0$ . With the above constitutive relations and neglecting the radiative heat transfer, Equations 2.1, 2.2 and 2.3 respectively read:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_k} (\rho u_k) = 0 \quad (2.7)$$

$$\rho \left( \frac{\partial u_i}{\partial t} + u_k \frac{\partial u_i}{\partial x_k} \right) = - \frac{\partial p}{\partial x_i} + \rho g_i + \frac{\partial}{\partial x_x} \left[ \mu \left( \frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} \right) + \delta_{ki} \lambda \frac{\partial u_j}{\partial x_j} \right] \quad (2.8)$$

$$\rho c_v \left( \frac{\partial T}{\partial t} + u_k \frac{\partial T}{\partial x_k} \right) = \frac{\partial}{\partial x_k} \left( \kappa \frac{\partial T}{\partial x_k} \right) - p \frac{\partial u_k}{\partial x_k} + \phi \quad (2.9)$$

The three components of the vector equation 2.8 are the Navier-Stokes equations expressing the conservation of momentum for a Newtonian fluid. In the thermal energy Equation 2.9,  $\phi$  is the dissipation function and is always positive. For a Newtonian, isotropic fluid, the viscous dissipation rate can be expressed as a function of the coefficients of viscosity and the derivatives of the velocity.

There are now six unknowns,  $\rho$ ,  $u_i$ ,  $p$  and  $T$ , and the five coupled Equations (2.7), (2.8) and (2.9) plus the equation of state relating pressure, density and temperature. These six equations together with sufficient number of initial and boundary conditions constitute a well-posed, albeit formidable, problem.

As for playing a relevant role in this work, it is important to add information regarding the shear stress. For a Newtonian fluid, the viscosity, by definition, depends only on temperature and pressure, not on the forces acting upon it. If the fluid is incompressible the equation governing the viscous stress (in Cartesian coordinates) is

$$\tau_{ij} = \mu \left( \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \quad (2.10)$$

Finally, because of its great importance in the field of fluid mechanics and because the solved cases in the guide are sorted according to it, the Reynolds number is presented:

$$Re = \frac{\text{inertia force}}{\text{viscous force}} = \frac{\rho U L}{\mu} \quad (2.11)$$

The relative importance of the ratio of the inertial forces to the viscous forces to determine the flow conditions is quantified by taking  $L$  as the characteristic scale of flow and  $U$  as characteristic velocity flow.  $Re$  represents a dimensionless number that can also be obtained when the Navier-Stokes equation is converted to a dimensionless form.

---

## 2.2 About numerical methods

As the theoretical fundamentals of the numerical methods and the finite volume method in particular are complex and difficult to widely explain, only the most relevant ideas are exposed. The scheme shown in [2] has been used as a model.

The equations of conservation of mass and momentum exposed above are difficult to solve in general. They are non-linear and coupled, which makes it difficult to treat them with the existing mathematical tools. An analytical solution is only feasible in very simple problems without much relevance. Even in some cases where simplifications of the equations are possible, the resolution remains complex.

To achieve a solution using numerical methods, a discretization of the domain is required, whose quality is determinant for the validity of the results. Numerical solutions are always approximations due to the existence of sources of errors: the discretization process, modified differential equations, iterative resolution methods, etc. It is possible to reduce the error of the discretization using more precise approximations, but it then negatively impacts on the cost and time of the simulations.

Numerical methods are mathematical tools, mainly used to easily achieve solutions which may be extremely complex. As mathematical schemes, they must meet specific criteria, ensuring that the solution is coherent and valid. For instance, the discretization must be such that when geometrical and/or temporal spacing tend to zero, the discretized equation and the exact equation coincide.

Moreover, the method must not diverge. This is to ensure that the error is being reduced while the iterative method proceeds.

Something important to be kept in mind is that nice and colorful results do not necessarily have a real physical meaning. It is necessary to contrast the results and figure out if they are within an appropriate range. Furthermore, it is compulsory to develop laboratory methodologies to carry out experimental work in order to compare and contrast numerical results with *in-lab*, *in-vitro* and *in-vivo* tests.

## 2.3 Description of OpenFOAM

OpenFOAM is first and foremost a *C++ library*, used primarily to create executables, known as *applications*. The applications fall into two categories: *solvers*, that are each designed to solve a specific problem in continuum mechanics; and *utilities*, that are designed to perform tasks that involve data manipulation. The OpenFOAM distribution contains numerous solvers and utilities covering a wide range of

problems.

One of the strengths of OpenFOAM is that new solvers and utilities can be created by its users with some pre-requisite knowledge of the underlying method, physics and programming techniques involved.

OpenFOAM is supplied with pre- and post-processing environments. The interface to the pre- and post-processing are themselves OpenFOAM utilities, thereby ensuring consistent data handling across all environments. The overall structure of OpenFOAM is shown in Figure ??:

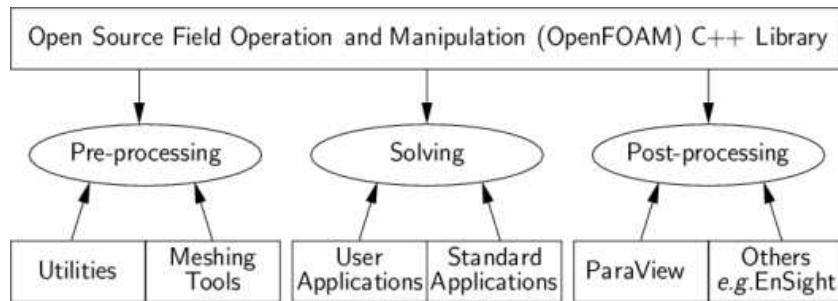


Figure 2.1: Overview of OpenFOAM structure, extracted from [3]

In this project, Version 2.2.1 of OpenFOAM has been used.

## 2.4 Overview of characteristics of the thoracic aorta

### 2.4.1 The circulatory system

The circulatory system in humans include three important parts: a heart, blood and blood vessels. The heart pumps the blood through the vessels in a loop, and the system is able to adapt to a large number of inputs as the demand on circulation varies throughout the body, day and life.

During systole, the left ventricle in the heart contracts and ejects the blood volume into the aorta. The blood pressure in aorta increases and the arterial wall is distended. After the left ventricle has relaxed, the aortic valve closes and maintains the pressure in the aorta while the blood flows throughout the body. The blood continues to flow through smaller and smaller arteries, until it reaches the capillary bed where water, oxygen and other nutrients and waste products are being exchanged, and is then transported back to the right side of the heart through the venous system. The right side of the heart pumps the blood to the lungs for oxygenation,

---

which then enters the left side of the heart again, closing the loop [4].

### 2.4.2 Anatomy of the aorta

The blood leaves the left ventricle of the heart during systole and is ejected through the aortic valve into the ascending aorta. After the ascending aorta the blood deflects into three larger branching vessels in the aortic arch which supplies the arms and head, or makes a 180 turn and continues through the descending and thoracic aorta towards the abdomen.

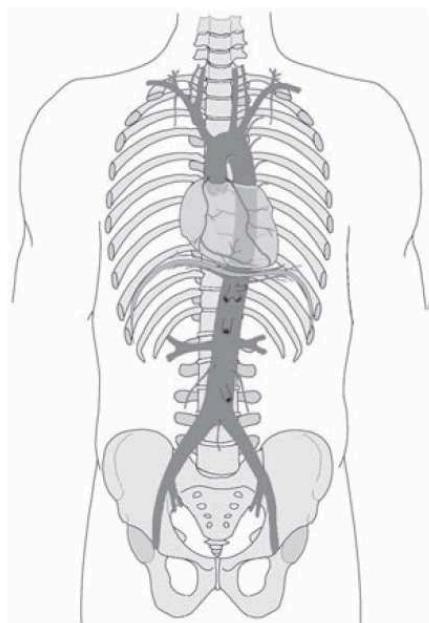


Figure 2.2: Scheme of the aorta in the human body, extracted from [4]

The parts of the aorta all have different shapes, in terms of branching, branching and tapering, creating different flow fields. The flow behaviour in the ascending aorta is characterized by the flow through the aortic valve, and the curvature can create a skewed velocity profile. The flow in the arch is highly three-dimensional, with helical flow patterns developing due to the curvature. The flow patterns that are created in the ascending aorta and arch are still present in the descending aorta, where local recirculation regions may appear as a result of the curvature and bending of the arch.

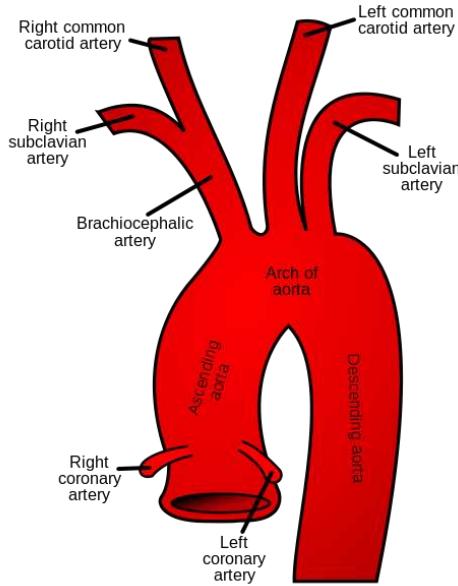


Figure 2.3: Main parts of a healthy thoracic aorta, extracted from [5]

The aortic wall is elastic in its healthy state, and will deform due to the increase or decrease in blood pressure [4].

### 2.4.3 Blood as a fluid

From the point of view of fluid mechanics, the main physical properties of blood required to prepare the OpenFOAM simulations are the following:

- **Viscosity:**  $\mu = 3.5 \times 10^{-3} \frac{kg}{m \cdot s}$
- **Density:**  $\rho = 1040 \frac{kg}{m^3}$

A controversial issue is whether to consider blood as a Newtonian or non-Newtonian flow. According to [6], in large arteries, the shear stress exerted on blood elements is linear with the rate of shear, and blood behaves as a Newtonian fluid, as Equation 2.10 shows. However, in the smaller arteries, the shear stress acting on blood elements is not linear with shear rate, and the blood exhibits a non-Newtonian behavior. Then, different models as the *power law* model or the *Carreau* model should be used.

In this work, and for modelling blood through a large artery as the aorta, blood has been considered a Newtonian fluid.

---

## 2.5 Relevance of the wall shear stress in medical research

There are two main forces applied to the arterial wall by blood, shown in the following Figure:

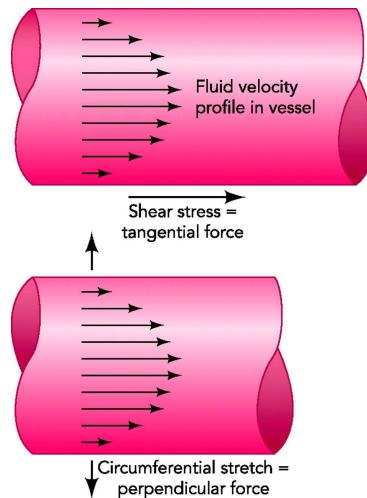


Figure 2.4: Forces applied by blood to the walls of a vessel, extracted from [7]

Blood pressure is a force that is directed perpendicular to the wall and is responsible for the cyclical distension of the vessel wall. As the blood pressure changes during the cardiac cycle, so the vessel wall extends then distends. The second force is the wall shear stress. This is the force acting on the inner lining of the artery wall, the endothelium, and is a frictional force resulting from the viscous drag of blood on the wall. Typically, wall shear stress has values of 0.5-20 Pa in a healthy artery, compared with the 10000 Pa blood pressure. As OpenFOAM works with kinematic pressure ( $\frac{p}{\rho}$ ), this range of the wall shear stress becomes  $4.8 \times 10^{-4}$  to  $0.02 \text{ m}^2/\text{s}^2$ .

The stretching of the artery during the cardiac cycle induces stress or tension within the artery wall. In health the increased stress acts to return the arterial wall to its *resting* position in a same way that a string will return to its resting position once the stretching force is released. This *tissue stress*, like blood pressure, is a large force in comparison with wall shear stress. It is high *tissue stress* caused by blood pressure that is responsible for the eventual rupture of both atherosclerotic plaque<sup>1</sup>

---

<sup>1</sup>**atherosclerotic plaque:** a deposit of fat and other substances that accumulate in the lining of the artery wall. Its rupture triggers a cascade of events that leads to clot enlargement, which may quickly obstruct the flow of blood. A complete blockage leads to ischemia of the myocardial (heart) muscle and damage

## 2. Theoretical foundations

---

and aneurysms<sup>2</sup>, not wall shear stress, which is a relatively tiny force incapable of producing the stresses required for rupture.

The wall shear stress plays an important role in the long-term pathology of cardiovascular disease. Studies have shown that changes in wall shear stress result in a host of cell signalling events which give rise to effects over different timescales, from seconds (release of nitric oxide) to hours (alignment of endothelial cells<sup>3</sup> with the wall shear stress direction) to weeks (change in diameter of arteries).

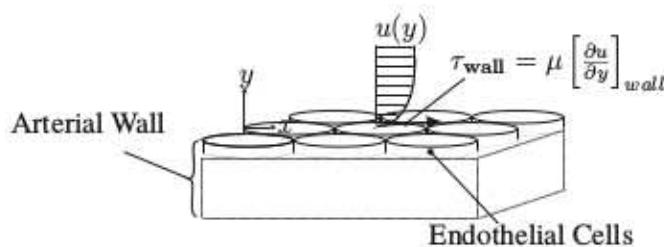


Figure 2.5: Scheme of the wall shear stress on the endothelial cells, extracted from [8]

It is increasingly recognised that disease progression is a complex interplay between local biology and local mechanical forces, including both wall shear stress and *tissue stress*. In atherosclerosis<sup>4</sup>, initiation of disease has long been recognised as occurring at regions of low wall shear stress. Reviews of the role of wall shear stress in disease development have emphasised the importance of low wall shear stress in disease initiation. Studies noted an inverse relation between intima-media thickness (IMT)<sup>5</sup> and wall shear stress.

It was hypothesised that high wall shear stress stimulates macrophage activity, leading to thinning of fibrous cap (or arterial lumen, the space inside the artery), which is then at risk of rupture through high tissue stress. The role of wall shear stress in stimulating the inflammatory process has been detected, with an atheroprotective (that protects against the formation of atherosclerosis) effect on high wall shear stress [9].

---

<sup>2</sup>**aneurysm:** localized, blood-filled balloon-like bulge in the wall of a blood vessel

<sup>3</sup>**endothelial cells:** cells forming the endothelium, which is the thin layer that lines the interior surface of blood vessels and lymphatic vessels, forming an interface between circulating blood or lymph in the lumen and the rest of the vessel wall

<sup>4</sup>**atherosclerosis:** specific form of arteriosclerosis in which an artery wall thickens as a result of invasion and accumulation of white blood cells

<sup>5</sup>**intima-media thickness:** is a measurement of the thickness of tunica intima and tunica media, the innermost two layers of the wall of an artery. IMT is used to detect the presence of atherosclerotic disease in humans

---

## 2.6 4D flow visualization method used for comparison

Thanks to the new medical techniques, such as MRI and CT scanning, it is possible to obtain a lot of information non-invasively and to visualize complex geometry of the patients. The comparison with the OpenFOAM simulations presented in this work will be done according to the results obtained with DiPPo, a tool able to integrate the velocity profile determined by 4D phase-contrast magnetic resonance in computational fluid dynamics, developed by the authors of [10].

One of the most important parts for the data acquisition is the magnetic resonance. For the aorta exposed in this work, measurements were carried out using a 3 T MR system (Magnetom TRIO; Siemens, Erlangen, Germany) time-resolved, 3-dimensional MR velocity mapping based on an RF-spoiled gradient-echo sequence with interleaved 3-directional velocity encoding (predefined fixed velocity sensitivity = 150 cm/s for all measurements). Data were acquired in a sagittal-oblique, 3-dimensional volume that included the entire thoracic aorta and the proximal parts of the supra-aortic branches. Each 3-dimensional volume was carefully planned and adapted to the individual anatomy (spatial resolution,  $2.1 \times 3.2\text{--}3.5 \times 3.5\text{--}5 \text{ mm}^3$ ).

In the in vivo situation, measurements may be compromised by the active cyclic motion of the heart (cardiac contraction and dilation) and the passive motion of the heart due to respiration. These motion components may lead to image artifacts and uncertainties about the exact measurement site in the aorta. Only if the breathing state was within a predefined window data was accepted for the geometrical reconstruction. To resolve the temporal evolution of vascular geometry and blood flow, measurements were synchronized with the cardiac cycle. The velocity data was recorded in intervals of Temporal Resolution (TeR) throughout the cardiac starting after the R-wave of the ECG. The initial delay after R-wave detection was required for execution of the navigator pulse and processing of the navigator signal. Two-fold acquisition ( $k$ -space segmentation factor = 2) of reference and 3-directional velocity sensitive scans for each cine time frame resulted in a temporal resolution of 8 repetition time = 45 to 49 milliseconds. To minimize breathing artifacts and image blurring, respiration control was performed based on combined adaptive  $k$ -space reordering and navigator gating. Further imaging parameters were as follows: rectangular field of view =  $400 \times (267\text{--}300) \text{ mm}^2$ , flip angle = 15 degrees, time to echo = 3.5 to 3.7 milliseconds, repetition time = 5.6 to 6.1 milliseconds, and bandwidth = 480 to 650 Hz per pixel. For velocity measurements a voluntary healthy, male

## 2. Theoretical foundations

---

subject underwent MR examinations; written informed consent was obtained from the subject [10].

Another relevant issue of the image treatment is blood flow velocity decoding. Blood flow velocity in each voxel depends on acquisition velocity and gray scale. In general the velocities are encoded in the phase difference images such that there is a linear relationship between the gray scale value and the underlying velocity:

$$Velocity = \frac{PixelValue - GrayScale}{GrayScale} \cdot V_{enc} \quad (2.12)$$

where  $V_{enc}$  is the velocity sensitivity in cm/s (in the current case,  $V_{enc} = 150$  cm/s),  $GrayScale$  depends of the DICOM (2048 in the current case) and  $PixelValue$  is the value of the gray color of the phase contrast image [10].

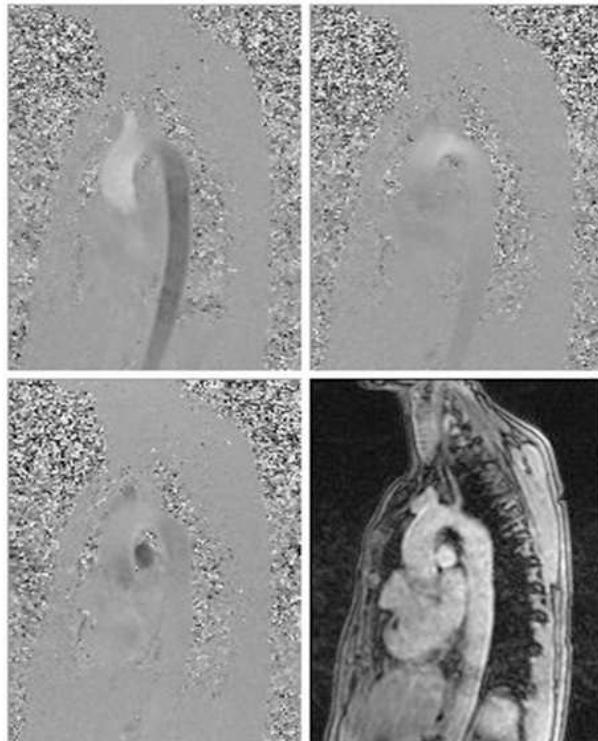


Figure 2.6: Phase contrast image (through plane velocity encoding) in  $V_x$ ,  $V_y$ ,  $V_z$  and magnitude image at the third iteration

---

## 2.7 State of the art

### 2.7.1 Learning OpenFOAM

OpenFOAM (named FOAM in its origins) was developed in the late 1980s at Imperial College (London), being sold some years later by UK company Nabla Ltd. In 2004, this company ceased operation and released FOAM as open source under GNU General Public License, under the name *OpenFOAM*. At this moment, two independent companies were created:

- OpenCFD Ltd.
- Wikki Ltd.

Both of these companies were started by some of FOAM's original developers and have nothing to do with one another. Each company maintains their own variation of OpenFOAM [11].

The Official OpenFOAM project is maintained by OpenCFD Ltd., whose variation is likely the most installed. On September 2012, the ESI Group announced the acquisition of OpenFOAM Ltd. from SGI.

This report explains at several places the difficulties found by new users to learn OpenFOAM due to the lack of maintained documentation. The main (and only) official tutorials freely provided by OpenCFD are:

- **User Guide:** is the main documentation for initiating with OpenFOAM. It contains information about its general running, as for instance data structures, compilation, applications, libraries, mesh generation, post-processing and more
- **Programmer's Guide:** contains more exhaustive explanations about the OpenFOAM programming issues
- **Example cases:** OpenFOAM cases with pre-prepared codes to be run

In spite of the fact that these documents show a very complete description of the OpenFOAM possibilities, they only offer an initial idea of the enormous potential of OpenFOAM. This is the reason why there has been an increasing use of Internet forums, most of them managed by other CFD learners. There, it is possible to expose personal problems, ask for existing functionalities, provide new utilities, and much

## 2. Theoretical foundations

---

more. There are intense debates where it is possible to participate and contribute with new ideas and solutions.

Moreover, one of the main contributions to expand the OpenFOAM knowledge has been the creation of websites containing solved cases, showing how to program specific utilities, explaining more deeply the different kinds of OpenFOAM solvers, etc. However, in the majority of websites, all the cases done by other users are basically academic or particular projects. It implies that the contents are based on the analysis of the results rather than exposing how the OpenFOAM simulations were carried out. Examples of very interesting works developed with OpenFOAM as university theses are [12], [13], [14], [15].

Consequently, these contents are very useful to contrast programming techniques and to extract main ideas, but are not adequate to introduce or improve the OpenFOAM skills, as it would be done with a guide in which the users can learn step by step. Although in most of them the OpenFOAM codes used to program the simulations are included, it is rarely possible to find accurate teaching explanations of its OpenFOAM simulations.

On the other hand, it would be unfair to say that there are no tutorials selflessly distributed by OpenFOAM users. Some examples are the following, encompassing different kinds of topics:

Filling of a tank

Airflow over a car

A comprehensive tour of snappyHexMesh

Airfoil simulation using gmsh

Dynamic meshes

These and many more tutorials are complete and allow an understanding of the OpenFOAM parameters involved in those simulations. The only handicap is that some of them develop and explain advanced tools, as for instance dynamic meshing. As a consequence, there is a gap between completing the first tutorial (explained in the official User Guide [3]) and accessing these complex tutorials. It is commonly accepted that the *Cavity tutorial* exposed in [3] is perfect to start with OpenFOAM and learn its basic fundamentals. However, other simple cases like this one, focusing on different OpenFOAM aspects, would be needed before moving to advanced tutorials and/or websites.

---

In conclusion, the OpenFOAM guide developed in this work allows new users to establish and extend their OpenFOAM background once the main tutorial of the official guide is done. Then, with the current guide it would be possible to improve the comprehension of the OpenFOAM structure, learn programming techniques, understand how to mesh different kind of geometries (2D and 3D), familiarize with the main pre- and post-processing OpenFOAM and ParaView capabilities, figure out which solvers and physical models are more adequate for each kind of fluid mechanics problem and many more. With it, it would be then much easier to use complex utilities found in internet and follow specific tutorials focused on advanced tools. Moreover, while most of the internet tutorials are each focused on one specific case, the current guide offers five different solved fluid mechanics problems with high applicability and all included in the same document. It has been conceived to guide the user through several cases, starting by simple ones and then following an increasing degree of difficulty. As all of them will be explained following the same schemes, structure and approach, the learning process may be simpler.

### **2.7.2 Thoracic aorta studies and hemodynamics CFD simulations**

The use of CFD techniques in simulating blood patterns and modelling cardiovascular systems has become widespread within bioengineering and medical research in the past few decades. There are several advantages in using CFD to characterize the cardiovascular system (called *in silico* models), instead of the traditional *in vivo* experimental studies [16]:

The relatively low costs associated with *in silico* models

The less invasive nature of numerical simulations, since only minimal measurements from the patient are needed

The ability to precisely control boundary conditions in the models

The ability to accurately compute quantities that are difficult to measure *in vivo*, such as the wall shear stress

Coupling medical imaging and CFD allows to calculate highly resolved blood flow patterns in anatomically realistic models of the thoracic aorta, thus obtaining the distribution of WSS at the luminal surface. However, the increasing reliance on CFD for hemodynamic simulations requires a close look at the various assumptions

## 2. Theoretical foundations

---

required by the modeling activity. In particular, much effort has been spent in the past to assess the sensitivity to assumptions regarding boundary conditions [17].

Regarding the geometry, description of the arterial geometry used in CFD simulations is very important and essential for the results. Local arterial geometry components as curvature and smoothness will highly influence the results. The resolution of human aortas will also be influenced by the general geometry or topology of the number and location of branches [8].

Regarding the boundary conditions, an accurate and validated mechanical model would be required for the description of vessel movements during the cardiac cycle. There is a debate in the literature on the validity of currently proposed models for this purpose [8]. If instantaneous wall shear stress is important, it is necessary to consider a fluid-structure interaction simulation that can account for the deformation of the arterial wall [18]. On the other hand, the assumption of rigid wall geometry is commonly accepted in computational hemodynamics [8]. It is recommended to prescribe outflow boundary conditions based on *in vivo* accurate measurements. Depending on its location and type, the inlet velocity profile seems to influence both bulk flow and wall shear stress distribution [17][19].

The inlet flow profile was measured with MRI and prescribed in the ascending aorta, while an impedance pressure boundary condition was set in the thoracic aorta. Velocity contours in the descending aorta were found to be in very good agreement with MRI measurements, with prediction of flow reversal on the inner side in the descending aorta [18].

The average peak Reynolds number was higher in the ascending ( $\approx 4500$ ) and descending aorta ( $\approx 4200$ ) than in the aortic arch ( $\approx 3400$ ). Thirty young healthy volunteers were examined by MRI. According to the calculated critical Reynolds numbers, flow instabilities were prominent in the ascending (14 out of 30 volunteers) and descending aorta (22 out of 30 volunteers) but not in the aortic arch (3 out of 30 volunteers). The supraventricular Reynolds number, indicating flow instabilities, is significantly correlated with body weight, aortic diameter and cardiac output. While the findings might suggest the presence of flow instabilities in the healthy aorta at rest, this does not involve fully turbulent flow [20].

## 2.8 Hypotheses used for aorta simulation

The hypotheses assumed for the aorta simulation are the following:

Peak systolic conditions

---

Incompressible flow

Laminar flow

Newtonian flow

Rigid, blood-impermeable and smooth aorta walls

Static aorta geometry

Uniform inlet velocity profile

Right and left coronary arteries are not included

It is important to remark that the simulations will be carried out for a particular time instant, in peak systolic. It represents the time with higher inlet flow speed and therefore maximum values of wall shear stress. As one of the main goals of the work is to determine wall shear stress that blood causes in order to prevent medical diseases, the more critical conditions are simulated. As a consequence, the simulations are not going to be transitory, with constant boundary conditions, and thus the results must be analyzed accordingly.

### 3. Pre-processing of the aorta with OpenFOAM

The explanations contained in this Chapter detailing how the case was prepared for the simulations have four main purposes:

Expose the methodology used for the pre-processing of the OpenFOAM simulations of the aorta in case it would be necessary to repeat it in future studies

Show the progression of the work itself, encompassing programming of the OpenFOAM codes, development of tools used to improve the computation of the fluid dynamics parameters and adequacy of the initial aorta geometries for a suitable study

Expose the main steps and key factors when simulating internal laminar flow with OpenFOAM. Although the study of hemodynamics presented in this work focuses on a particular case to be solved (the aorta), the pre-processing of the case shown below can be used for simulations encompassing similar fluid mechanics characteristics

Expose the main steps and key factors when treating with OpenFOAM imported irregular geometries. As the geometry of the aorta used in this work has been obtained from the images of a real human body, one of the main difficulties of the case resolution has been to adapt this irregular geometry while making it suitable to be simulated with OpenFOAM

When exposing the methodology used for computer simulations, it is somewhat difficult to explain which steps were more determinant and complex to overcome before achieving the final results. This is the reason why in each one of the Sections of the current Chapter a brief text is included exposing the main difficulties found during the development of each part.

The OpenFOAM case (named **AortaFoam**) for the aorta simulations presents a structure of directories and subdirectories similar to the **aircraft** case exposed in the guide.

---

The process of segmentation by which the current aorta geometry was obtained (done with DiPPo [10]) is explained as follows: An  $n$ -phase 4D (spatio-temporal) image can be viewed as a discrete set of  $n$  volumetric images defined at  $n$  different time instants. The 4D aortic surface can also be viewed as a sequence of surfaces. During the segmentation stage, the 4D segmentation algorithm consists of the following steps:

- Aortic surface pre-segmentation: a 4D fast marching level set method simultaneously yields approximate 4D aortic surfaces
- Centerline extraction: aortic centerlines are determined from each approximate surface by skeletonization
- Accurate aortic surface segmentation: an accurate 4D aortic surface is obtained simultaneously with the application of a novel 4D optimal surface detection algorithm

In order to improve the aorta segmentation, a new step was added. It consists in introducing an optimization of Dijkstra's shortest path algorithm. Thanks to this algorithm, the region of interest is more precise, the tubular structures are improved and it is possible to select the branches of the aorta according to some values.

To understand the capabilities offered by the 4D method, a figure showing the module of the velocity field in peak systolic conditions is shown next:

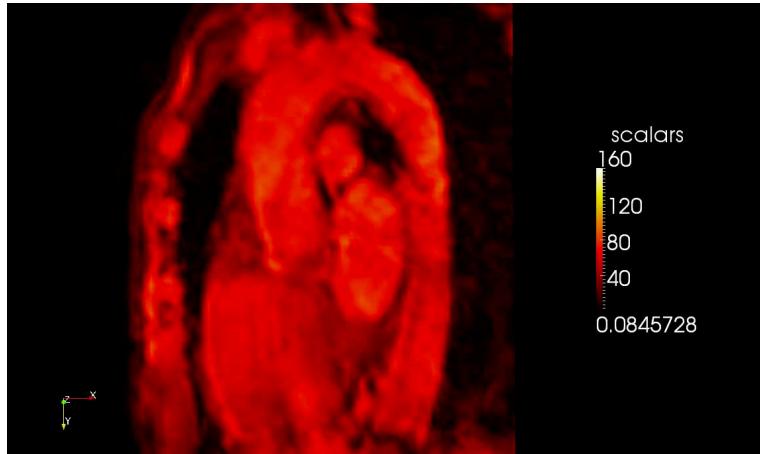
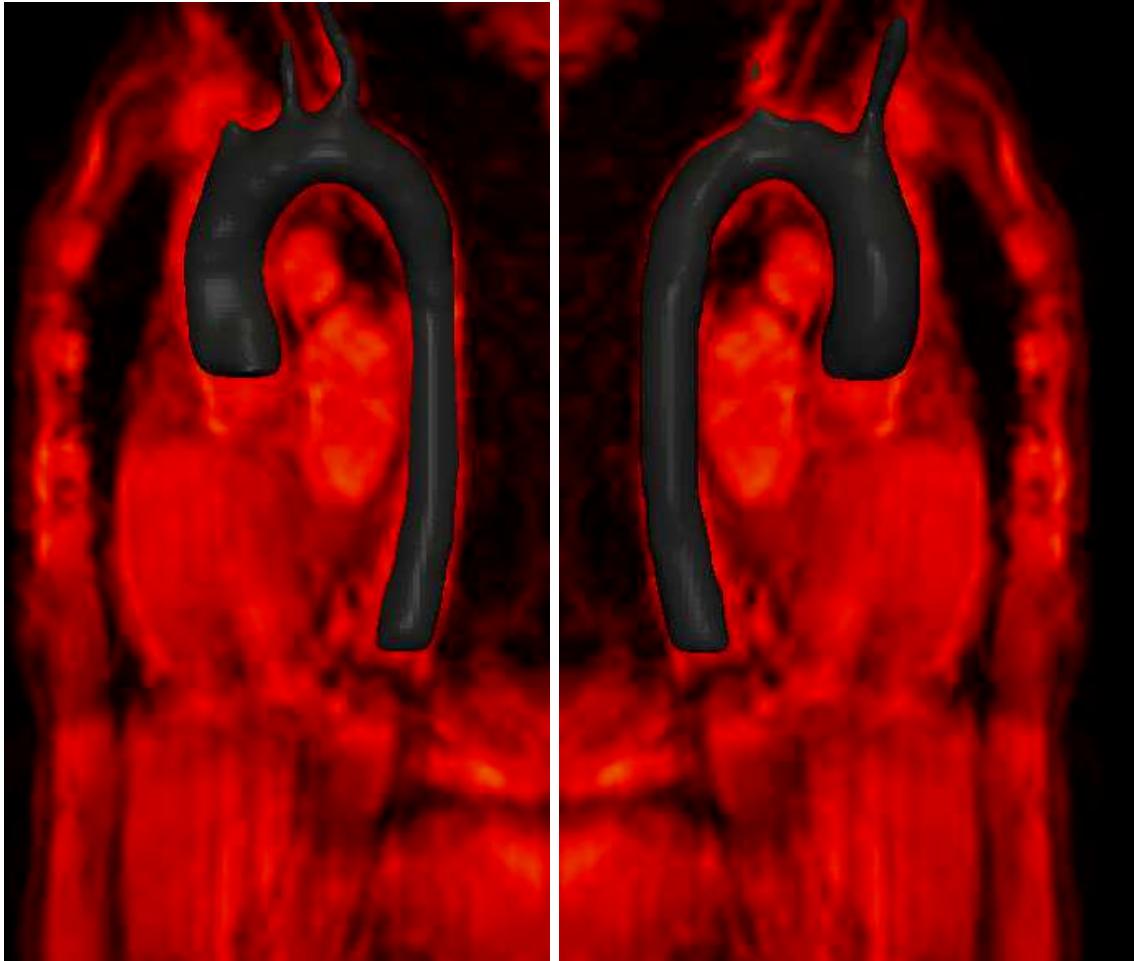


Figure 3.1:  $|\mathbf{U}|$  in a longitudinal slice of the aorta obtained from the 4D method (cm/s)

Figure 3.1 shows the behaviour of blood through the aorta obtained by using MRI techniques, containing both the aorta and other human tissues. The figures below

show the shape of the aorta (superposed to the medical image) used for the computation of the wall shear stress and obtained applying the segmentation process exposed above. It is the same geometry used for the OpenFOAM simulations:



Images used for the segmentation process with the shape of the aorta geometry used for the OpenFOAM simulations

The segmentation has been done during the maximum systole.

### 3.1 Preparing the geometry

#### 3.1.1 The 3 simulated geometries

**Aim of the task:** in order to investigate the fluid dynamics in the aorta, 3 different models have been used. Only the third (Figure 3.3) represents a real

---

geometry, while the others are simplifications used for an easiest treatment of the case until the most complex steps of the pre-processing were fulfilled.

**Main characteristics of each model:**

- 3 models of study**
- First model: primitive geometry with approximate inlet and outlet and without any supra-aortic vessel
  - Second model: more realistic but with only two of the three supra-aortic vessels
  - Third model: real geometry with the main inlet, the main outlet and the three supra-aortic vessels

**Pictures of some models:**



Figure 3.2: First model of aorta used in the simulations

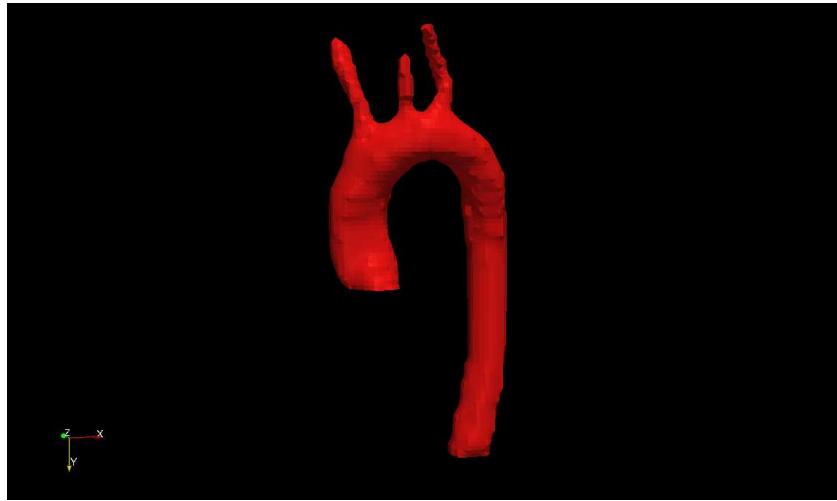


Figure 3.3: Third model of aorta used in the simulations

#### Working scheme:

Each one of the models has been used to study and test specific features of the pre-processing:

1. **First model** → as it is a very simple version of a real aorta, it has been used in the first steps of the investigation, especially with the preparation of the geometry. With this model, it has been possible to:
  - investigate how to smooth the surface
  - understand how to adapt external irregular geometry to OpenFOAM standards
  - find an adequate mesh for the case
  - develop a method to define the patches where the boundary conditions are applied
  - study how to obtain a high cell refinement at the walls of the aorta to compute an accurate value of wall shear stress
2. **Second model** → it presents a more realistic geometry of aorta. With the above questions being faced, it has been very important to investigate which boundary conditions would provide the more physically realistic flow conditions [21]. These tests have been done with model 2. This issue is further explained in Section 3.2, but the fact is that the boundary conditions imposed in the supra-aortic vessels are crucial for the mass flow rate distribution and the global behaviour of the fluid.

- 
3. **Third model** → it represents a real geometry, being the one used for the final simulations and for the analysis of the results once the main programming difficulties had been overcome with models 1 and 2. This is the reason why the pre-processing explanations and the methodology presented in the current section are going to be entirely done using the third geometry.

#### Main difficulties:

- Each time a new model was introduced, it was necessary to readapt all the pre-processing parameters
- The instructions of Section 3.1.6, which are laborious, had to be computed for each one of the models
- Since geometries were simplified and despite simulating properly, the results of models 1 and 2 might not be physically realistic

### 3.1.2 Smoothing the surface

**Aim of the task:** as can be seen in Figure 3.3, the surface of the initial aorta geometry is irregular, rough and in some regions with a bad cell definition. If it were directly meshed, these flaws might lead to misleading conclusions regarding the wall shear stress. As could be seen in the first tests, a marked irregularity on the surface is a stress concentrator and therefore the wall shear stress in these areas grows with no physical reason.

**Work methodology:** a laplacian smoother was applied to the initial STL surface. The `surfaceSmooth` OpenFOAM utility was used, and although most of the irregularities on the surface could not be erased but only smoothed, this tool highly helped in adapting the geometry for the simulation.

This utility is included in the OpenFOAM surface mesh tools and no dictionaries are needed to run it. To use the utility, once the STL file containing the aorta geometry (named `aortaGeometry.stl`) has been saved within `constant/triSurface`, it is necessary to type:

```
surfaceSmooth aortaGeometry.stl 0.5 10 lastAortaSmooth10It.stl
```

where the inputs are as follows:

### 3. Pre-processing of the aorta with OpenFOAM

---

```
<utility> <originalFileName> <relaxationFactor> <numberOfIterations> <outputFileName>
```

**Results:** the differences between the surfaces of the aorta with or without smoothing can be observed in the following figures:

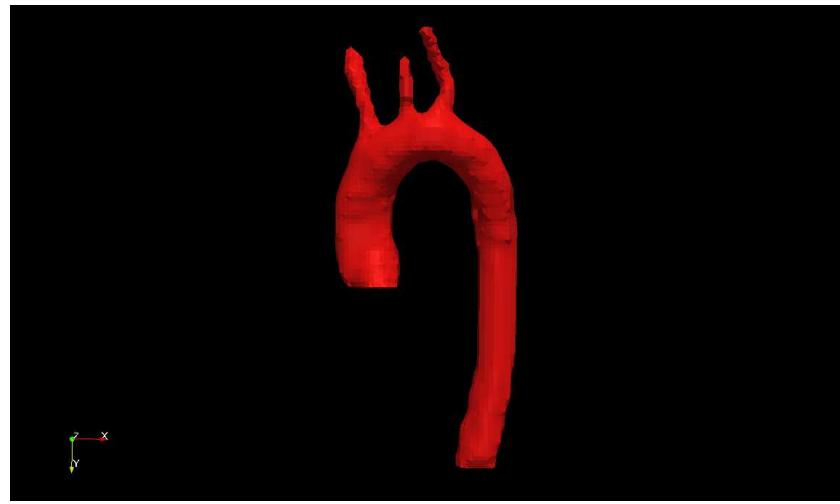


Figure 3.4: Surface of the aorta without smoothing

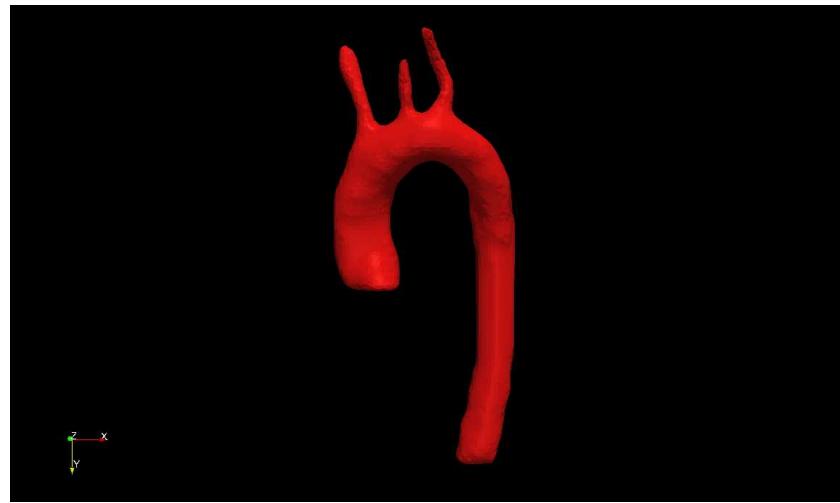


Figure 3.5: Surface of the aorta with smoothing

The differences between the surfaces with edges of the aorta with or without smoothing are the following:

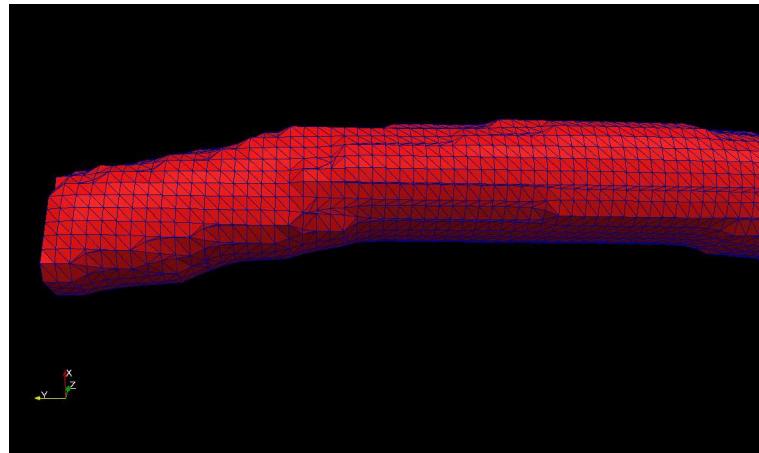


Figure 3.6: Surface with edges of the descending aorta without smoothing

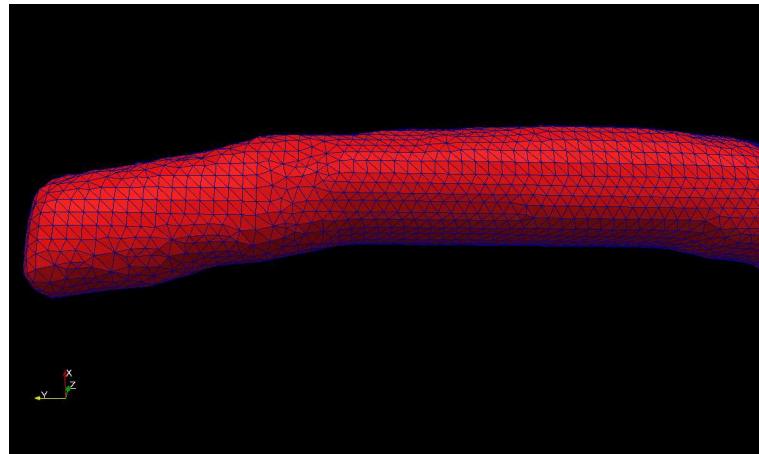


Figure 3.7: Surface with edges of the descending aorta with smoothing

### Main difficulties:

- Understand, considering the time it takes for simulating, how many iterations were required to obtain good results
- The `surfaceSmooth` utility is not commonly used and it took time to investigate its functionality
- Although the utility is very useful, some parts of the geometry could not be completely smoothed

### 3.1.3 Surface feature extracting

**Aim of the task:** with the improved geometry, the next step was to use the `surfaceFeaturesExtract` utility, which extracts feature edges from tri-surfaces (as explained in the sixth chapter of the guide). It is not a tool to improve neither the geometry nor the computation of the fluid dynamics parameters, but it is required when simulating cases using the `snappyHexMesh` utility.

**Work methodology:** to execute this utility, it is first necessary to edit a dictionary (`surfaceFeatureExtractDict`) located within `system`. It contains the code shown in Appendix A.1.

Once it is prepared, it is necessary to type:

```
surfaceFeatureExtract
```

### 3.1.4 Meshing the geometry

**Aim of the task:** the meshing process is one of the main steps when pre-processing the case, mainly because the quality of the mesh is directly related to the accuracy of the results. In fact, the boundary layer can only be adequately solved if a very high cell refinement is obtained at the walls of the aorta. Furthermore, as one of the main objectives of this work is to determine the wall shear stress, this issue becomes even more crucial.

**Work methodology:** different kinds of meshes were tested, analyzing the influence of the number of cells and the cell refinements at the walls with the results and the convergence time. This analysis is described in Section 3.5, where the mesh used for the main simulations and exposed in this Section is justified.

The `snappyHexMesh` meshing utility used in the `AortaFoam` case is shown in the sixth chapter of the guide. Once the parameters of the final mesh had been chosen, the main meshing steps were:

1. **Creation of a background mesh:** it is necessary to generate a structured mesh, whose cells will be divided in smaller cubes to mesh the aorta geometry:

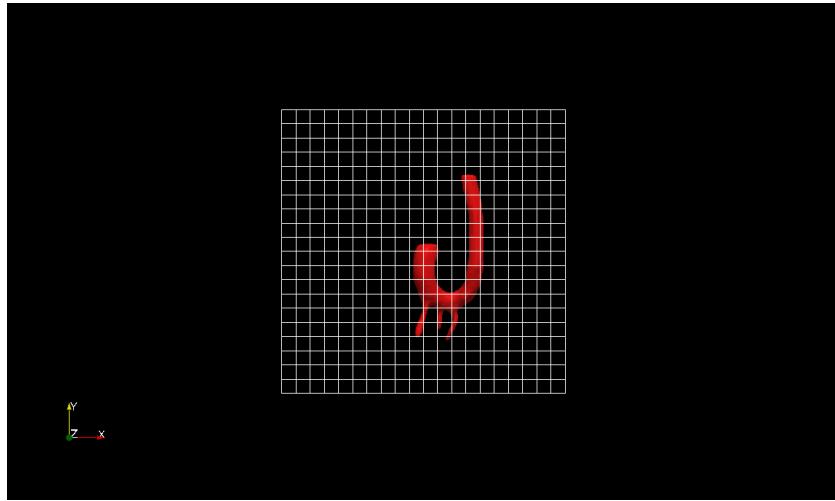


Figure 3.8: Surface of the aorta with the background mesh

This background mesh can be created by using `blockMesh`, whose dictionary is shown in Appendix A.2. The `boundary` file with the results of this process is shown in Appendix A.3.

2. **First meshing process of snappyHexMesh (`castellatedMesh`):** it creates a first and rudimentary cube-based mesh. As can be seen in Figure 3.9, the geometry of the aorta has been divided in small cubes but with a bad surface resolution:

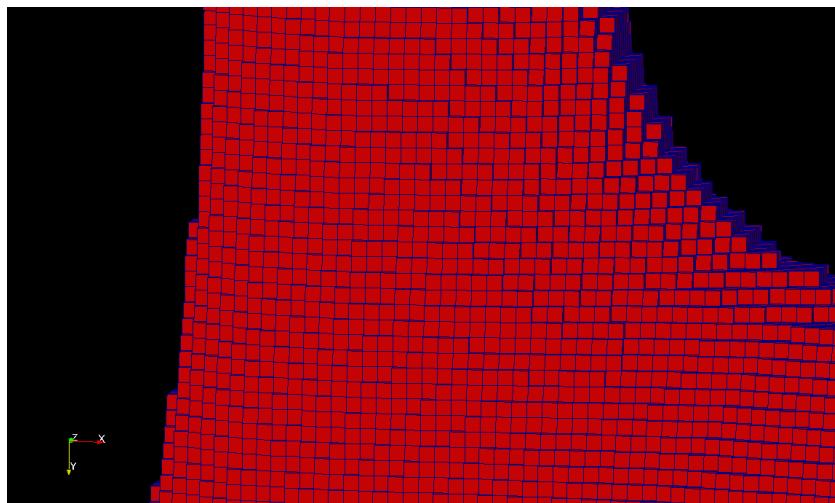


Figure 3.9: Mesh of the aorta after the first process of `snappyHexMesh`

3. **Second meshing process of snappyHexMesh (`snap`):** it works on the cells at the walls to adapt their vertices to the initial geometry in order to obtain a smooth and realistic mesh surface:

### 3. Pre-processing of the aorta with OpenFOAM

---

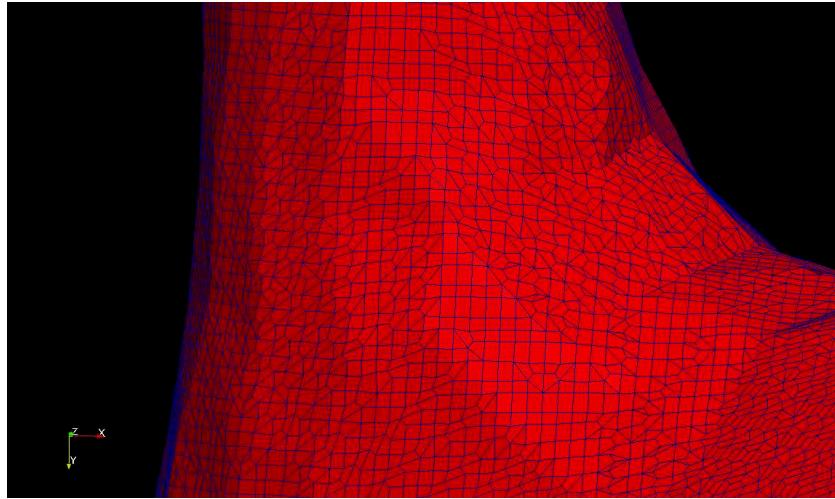
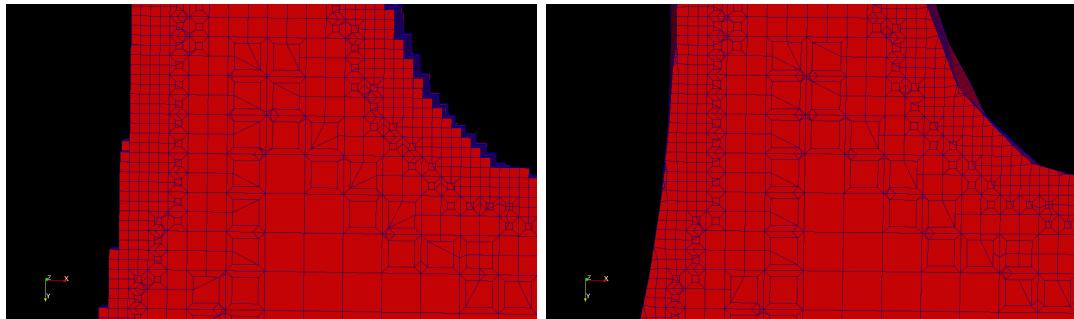


Figure 3.10: Mesh of the aorta after the second process of `snappyHexMesh`

The differences between the aorta mesh after and before this second meshing process:



Comparative between the mesh after each one of the processes carried out to mesh the geometry with `snappyHexMesh`

The dictionary to run the previous meshing processes is shown in Appendix A.4.

**Results:** among all the meshes, the chosen one presents a uniform core of cell level 4. The refinement level indicates how many times the initial cubes have been divided until the required refinement is achieved. Near the walls, and for a proper computation of the wall shear stress, the mesh is being refined up to level 6, with transition layers of level 5. This configuration is commonly used; a higher refinement would imply extremely slow iterations. The cell refinement can be observed in the following figures:

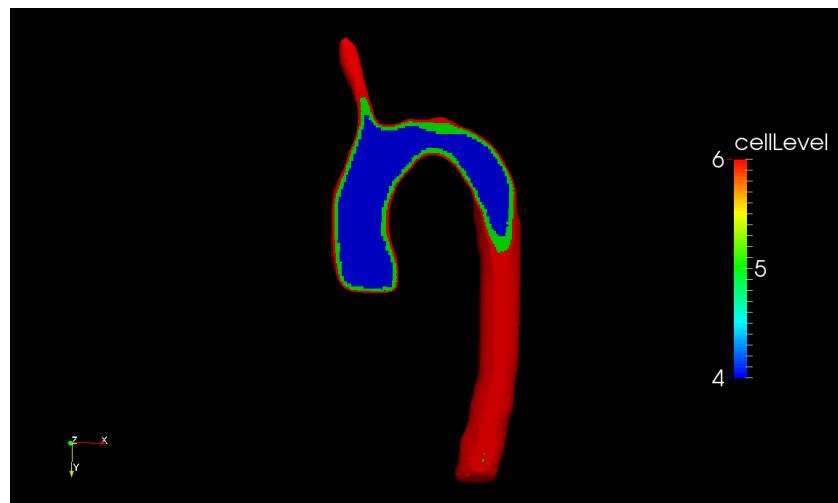


Figure 3.11: Cell level at the aortic arch of the initial mesh

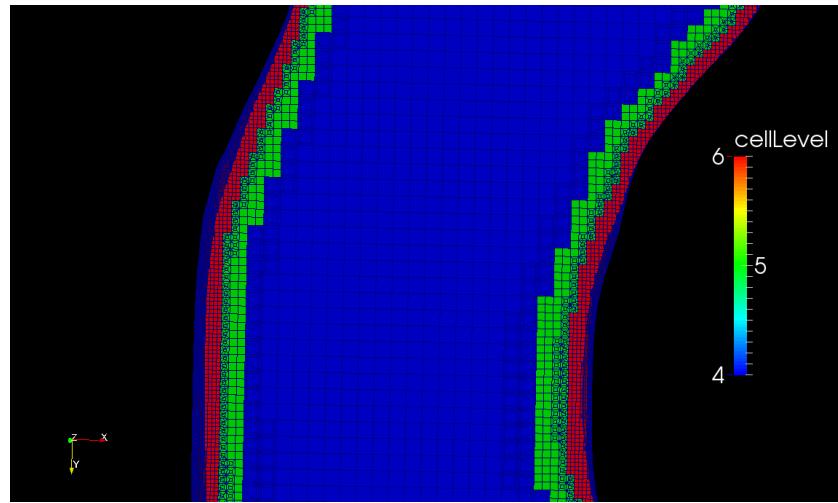


Figure 3.12: Detail of the cell level at the initial mesh of the aorta

### Main difficulties:

- The dictionary for using `snappyHexMesh` is complex and therefore it is difficult to have a thorough command of it
- The time needed to carry out the meshing process is high. Moreover, the higher the cell refinement, the slower the iterations
- Due to the previous point, it is laborious to make tests with different mesh configurations to figure out which mesh is more suitable for the case

- Compared to other dictionaries, in `snappyHexMesh` it is more complicated to find out programming errors
- Although the process runs as expected and the mesh quality is acceptable, the surface of the mesh may present some irregularities

#### 3.1.5 Refining the mesh to compute an accurate wall shear stress

**Aim of the task:** since it was one of the main objectives of the work and in order to contrast the results of the 4D method, a very accurate distribution of the wall shear stress was computed. To achieve this purpose, the distance between the walls of the aorta and the first node of the mesh has to be as low as possible. This is the reason why different mesh refining tools were investigated and tested to obtain very reliable results.

**Work methodology:** once the aorta geometry had been meshed, two different methods were used to refine the mesh at the walls of the aorta:

1. **addLayers:** it is the third and last meshing process of `snappyHexMesh`, and thus it uses the same dictionary shown in Appendix A.4. This utility does not change the geometry, but adds layers of cells at the walls of the domain. In addition to highly reducing the distance between the first nodes, this meshing process also corrects surface imperfections.

In Figures 3.13 and Figure 3.14 it is possible to observe the results after adding the layers of cells. In particular, for the `AortaFoam` case, two layers of cells have been added, being the first 0.4 times thicker than the 6-level cells, and the second 0.5 times thicker than the previous cell layer:

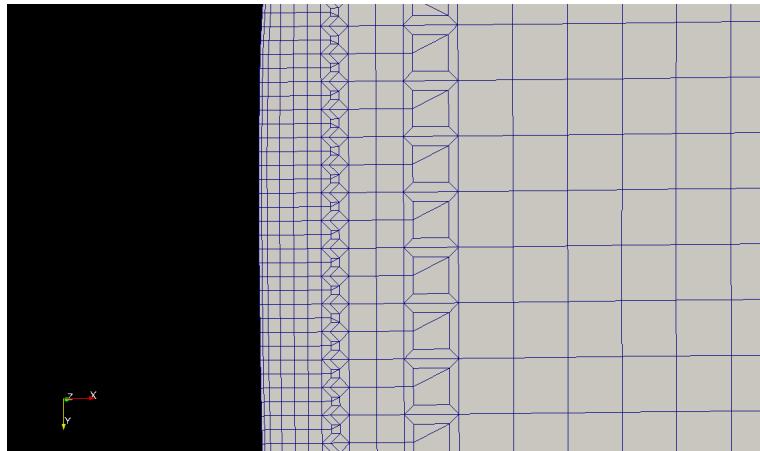


Figure 3.13: Detail of the cell refinement at the walls of the aorta

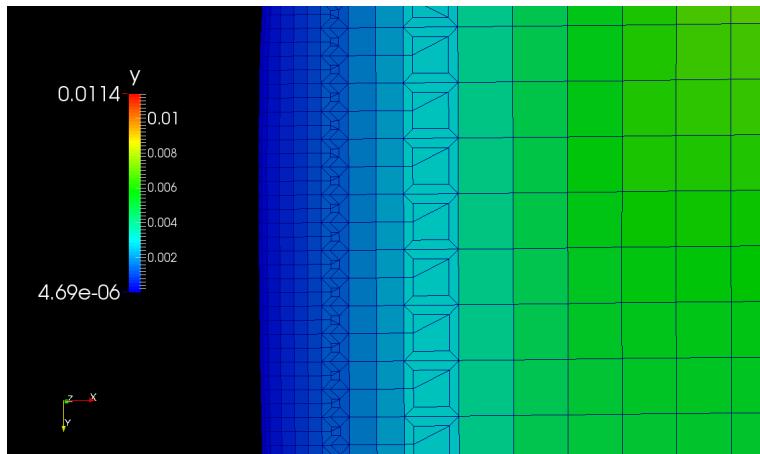


Figure 3.14: Detail of the cell refinement at the walls of the aorta with the magnitude of the distance between the wall and the nodes

As can be seen, the distance between the wall and the first node is of the order of micrometers ( $10^{-6}$  m). It is a very high cell refinement.

In Appendix A.5 the main quality parameters of the final mesh are shown once `addLayers` had been run. As can be seen, the mesh quality criteria have been maintained within an adequate range.

`addLayers` has been discovered as a fantastic tool to prepare the case for a thorough computation of the wall shear stress.

2. **refineWallLayer:** it is not a meshing process itself, but a utility which refines cells next to patches as much as specified. This method could not be finally used to pre-process the case: the quality of the mesh once it was applied was bad and inadequate for the simulation. As these

irregularities were not punctual but highly distributed along the whole mesh, no solution could be found to correct them and the `refineWallLayer` utility was finally discarded.

#### Main difficulties:

- The `addLayers` meshing process is even more slow than `castellatedMesh` and `snap`
- The second method might have given good results, but as the geometry was very irregular it could not be finally used. A lot of time was spent in trying to improve the quality of its mesh
- Only once the case has been simulated it is possible to observe the accuracy of the results of the wall shear stress as its computation is included in the post-processing

#### 3.1.6 Defining patches for the simulation

**Aim of the task:** the aim of the methodology shown in this section is to define the required patches on the surface to apply the boundary conditions. So, as initially the whole surface was a generic patch, it was necessary to make a distinction between five different patches. Figure 3.15 shows the aorta geometry with the name of these required patches and their location:

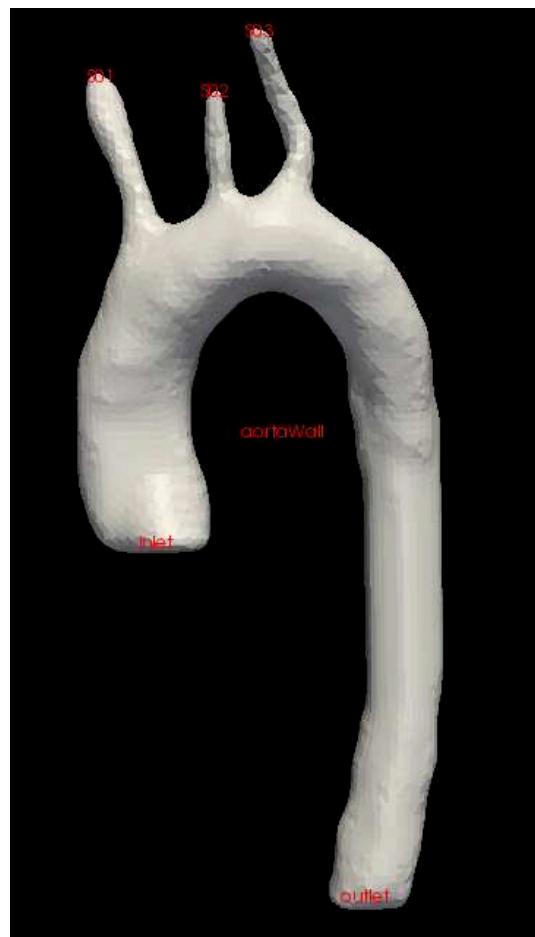


Figure 3.15: Geometry of the aorta with the name of the patches

**Main characteristics of each patch:**

|                          |   |
|--------------------------|---|
| <b>AortaFoam patches</b> | <p>inlet: patch including the faces at the inlet of the ascending aorta. Through this patch, the blood pumped in the heart penetrates into the ascending aorta</p> <p>outlet: patch including the faces at the outlet of the descending aorta</p> <p>S01: patch including the faces at the outlet of the brachiocephalic trunk</p> <p>S02: patch including the faces at the outlet of the left common carotid artery</p> <p>S03: patch including the faces at the outlet of the left subclavian artery</p> <p>aortaWall: patch including all the remaining faces, and where the wall shear stress is computed</p> |
|--------------------------|---|

**Work methodology:** in the majority of OpenFOAM simulations, this step would not have had any special relevance. However, as the aorta geometry has not been created with a 3D CAD software and the surface is irregular, a special treatment had to be used. OpenFOAM does not have a graphic interface for pre-processing, and thus the process of selecting a specific group of faces and redefine them as a new and different patch must be done by programming OpenFOAM files.

To face with this issue, two OpenFOAM utilities were used: `topoSet` and `createPatch`. The first one operates with `cellSets/faceSets/pointSets` through a dictionary. The second is a utility to create patches out of selected boundary

---

faces. During the pre-processing, these two utilities were combined to define the required patches.

The main idea is that **topoSet** includes a group of selected faces (conforming the **inlet**, **S01**, etc.) into a set. Afterwards, **createPatch** defines this group of faces as a new patch where the appropriate boundary conditions will be applied. However, how can **topoSet** understand which specific group of faces must be included within this set?

The main steps have been the following:

1. First, it is necessary to edit the **topoSetDict** (Appendix A.6), which is the dictionary to control the **topoSet** utility, being located within **system**. In this file, it is specified (for each group) the name given to the set, what it is going to contain (cells, faces or points) and which geometric entity is going to be used. The key of the issue is that all the faces (or cells or points) of the initial geometry contained within the geometric entity will be assigned to the set.

For the aorta simulation, 5 rectangular prisms (defined as **box** in **topoSetDict**) were used, plus another instruction to select all the external faces. Two of the six parts conforming **topoSetDict** are next shown:

```
1      {
2          name      allPatchSet;
3          type      faceSet;
4          action    new;
5          source    boundaryToFace;
6          sourceInfo
7          {
8          }
9      }
10
11     {
12         name      c1;
13         type      faceSet;
14         action    new;
15         source    boxToFace;
16         sourceInfo
17         {
18             box  (0.0425  0.0558  0.153) (0.0675  0.0708  0.177);
19         }
20     }
```

2. The second step consists of editing the **createPatchDict** file (Appendix A.7), the dictionary of the **createPatch** utility. There, for each patch it

### 3. Pre-processing of the aorta with OpenFOAM

---

is specified the name which will be given to them (`inlet`, `S01`, etc.), what type of new patch it is going to be and how it is going to be constructed (either from patches or sets). Again, two of the six parts of the dictionary are shown shown:

```
1      {
2          name inlet;
3
4          patchInfo
5          {
6              type patch;
7          }
8
9          constructFrom set;
10
11         patches (aortaWall);
12
13         set definitiveInletSet;
14     }
15
16     {
17         name outlet;
18
19         patchInfo
20         {
21             type patch;
22         }
23
24         constructFrom set;
25
26         patches (aortaWall);
27
28         set definitiveOutletSet;
29     }
```

3. Once both dictionaries have been successfully set, the standard procedure would be to activate both utilities typing:

`topoSet`

and afterwards:

`createPatch`

However, it does not work because `createPatch` can only use external faces to define the patches, whilst a great number of the faces contained within the sets created by `topoSet` are internal. To solve this problem, a

---

script using *bash* programming was developed (completely unrelated to OpenFOAM dictionaries).

4. As it has previously been said, although there are 5 inlets and outlets in the geometry, a sixth instruction was defined, whose function is to select all the external faces of the geometry. Consequently, as each of the faces contained in the sets is represented by a number, the main idea consists of comparing the numbers of the set of external faces with the numbers of a set containing the faces selected by the **boxes** (where some are internal and some are external). According to this, two sets of numbers are compared (for instance, **allPatchSet** with **boxOutletSet**), and those faces in common are written to a third set. Finally, **createPatch** works directly to these third sets, and therefore only external faces are defined as new patches.

The script (shown in Appendix A.8) to execute it must be used after **topoSet** and before **createPatch**. Next one of the five parts of the code is shown (the one used to define the **inlet**):

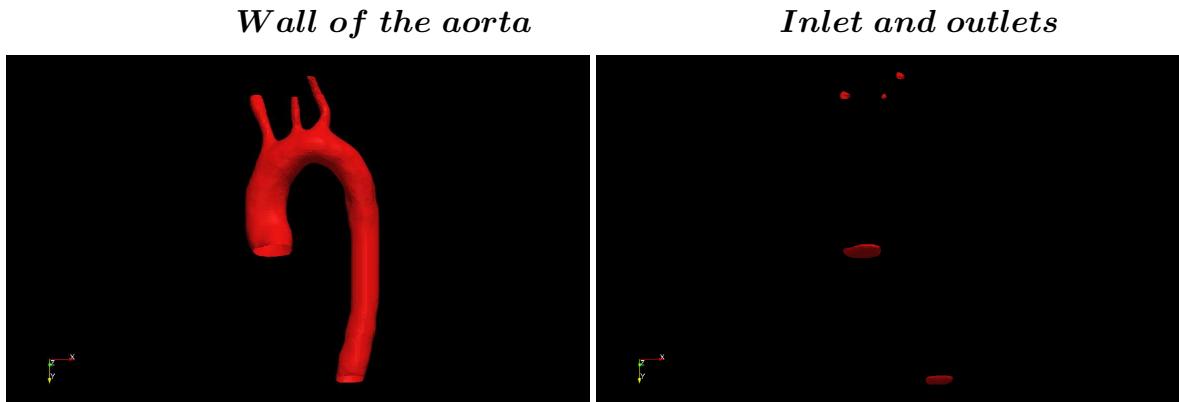
```
1 #!/bin/bash
2
3 cp AortaFoam/3/polyMesh/sets/c1 createPatchScript/c1Intermediate #Copy
   data from topoSet
4
5 sed -i 1,+19d createPatchScript/c1Intermediate #Erase header
6
7 sed -i '$d' createPatchScript/c1Intermediate
8
9 sed -i '$d' createPatchScript/c1Intermediate
10
11 sed -i '$d' createPatchScript/c1Intermediate
12
13
14 cp AortaFoam/3/polyMesh/sets/allPatchSet
      createPatchScript/allPatchSetIntermediate #Copy data from topoSet
15
16 sed -i 1,+19d createPatchScript/allPatchSetIntermediate #Erase header
17
18 sed -i '$d' createPatchScript/allPatchSetIntermediate
19
20 sed -i '$d' createPatchScript/allPatchSetIntermediate
21
22 sed -i '$d' createPatchScript/allPatchSetIntermediate
23
24 sort createPatchScript/c1Intermediate >
      createPatchScript/c1Intermediate.sort #Sort (necessary for the comm
      instruction)
25 sort createPatchScript/allPatchSetIntermediate >
      createPatchScript/allPatchSetIntermediate.sort
26
27 comm -12 createPatchScript/c1Intermediate.sort
      createPatchScript/allPatchSetIntermediate.sort >
      createPatchScript/definitiveInletSetIntermediate #Compare to find
```

### 3. Pre-processing of the aorta with OpenFOAM

---

```
common faces shearing the condition of external face and being
inside the box
28
29  wc -w < createPatchScript/definitiveInletSetIntermediate >
      createPatchScript/WordCounter #Count the number of faces of the
      resulting file
30
31  ed -s createPatchScript/definitiveInletSet <<< $'20r
      createPatchScript/definitiveInletSetIntermediate\nw'
32
33  ed -s createPatchScript/definitiveInletSet <<< $'18r
      createPatchScript/WordCounter\nw'
34
35  > createPatchScript/c1Intermediate #Reinitialize to 0 all the files
      which were used
36  > createPatchScript/allPatchSetIntermediate
37  > createPatchScript/WordCounter
38  > createPatchScript/c1Intermediate.sort
39  > createPatchScript/allPatchSetIntermediate.sort
40  > createPatchScript/definitiveInletSetIntermediate
41
42  cp createPatchScript/definitiveInletSet
      AortaFoam/3/polyMesh/sets/definitiveInletSet
43  cp createPatchTemplates/definitiveInletSetTemplate
      createPatchScript/definitiveInletSet
```

At the end of the process, the 6 different patches have been separated and independently defined in *polyMesh/boundary*. The results are shown in the following figures, where it is possible to observe which part was defined as aorta wall and which parts were defined as inlets or outlets:



Comparison between the wall of the aorta and the faces were boundary conditions are applied once the patch definition process was carried out

#### Main difficulties:

- As it has been explained, no OpenFOAM utility could be found to solve the problem regarding `createPatch` and the external faces. This

---

is the reason why many different methods were tested before concluding that this problem would require solutions external to OpenFOAM. After finding no alternative, the ad hoc solution using bash programming was developed

- The solution described in the previous point took a process of familiarization with bash programming
- As the method used in the script does not work directly to OpenFOAM dictionaries, there were at first adjustment problems between the script code and OpenFOAM
- The coordinates of the **boxes** used in *topoSetDict* had to be exhaustively calculated to accurately define the patches. Otherwise problems with the direction and magnitude of the boundary conditions might have appeared

## 3.2 Boundary conditions

**Aim of the task:** along with the mesh and the cell refinement, the boundary conditions are one of the main points of the pre-processing. The geometry of the aorta has been obtained from the images of a real body; however, to execute a realistic simulation of the mechanics of blood in the thoracic aorta, appropriate boundary conditions had to be applied.

For the application of the boundary conditions, the hypotheses shown in Section 2.8 were used. As peak systolic conditions are simulated (when the most critical conditions are achieved, and therefore higher values of wall shear stress are obtained), the boundary conditions are constant over time.

**Initial approach:** when simulating flow through pipes, the most common boundary conditions are to impose an inlet velocity and an outlet pressure (or viceversa). This is what was done in Model 1: a uniform inlet velocity (whose module and direction were obtained from the 4D viewing method, and therefore these values are not estimated but real) and a uniform outlet pressure. This pressure was set to 0 Pa, becoming the reference value.

However, when Models 2 and 3 were simulated, these boundary conditions had to be adapted due to the outgoing flow of the supra-aortic vessels.

**Work methodology:**

As explained in Section 2.4, the internal pressure in the supra-aortic vessels (patches S01, S02 and S03) is different than the pressure at the **outlet** of the thoracic aorta. As a consequence, if the same pressure boundary conditions would be applied to all of them, the results would not be physically realistic. In principle, neither the mass flow rate distribution nor the flow structure would match reality.

According to literature, and to solve this problem, it would have been necessary to introduce an electrical analogy to compute the pressure drop in each region. Nevertheless, two alternatives were tested thanks to the information provided by the 4D viewing method:

- Impose uniform outlet velocity at the supra-aortic vessels
- Impose uniform volumetric flow rate at the supra-aortic vessels

Both kinds of boundary conditions were pretended to be equivalent to imposing uniform values of pressure in each one of the supra-aortic vessels. However, as it is very difficult to obtain experimental pressure data in those regions, boundary conditions related to the velocity field were tested.

This is the reason why 3 different cases regarding boundary conditions were tested. Common features in all of them are a uniform aorta inlet velocity and a uniform aorta outlet pressure, but each case presents different boundary conditions at the supra-aortic vessels:

#### 1. First Case: uniform supra-aortic vessels outlet velocity

##### Scheme of the boundary conditions:

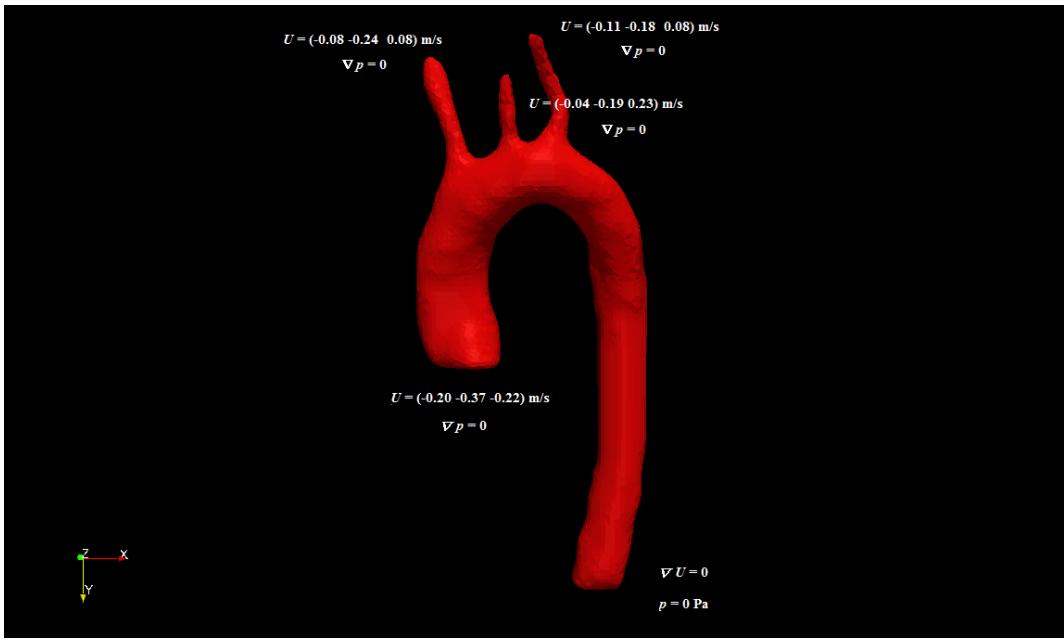


Figure 3.16: Boundary conditions applied to the First Case

### Results:

Although the case converged, the solution did not present a physically realistic behaviour of the flow. Singular points appeared with large flow accelerations, mainly in the supra-aortic vessels.

## 2. Second Case: uniform supra-aortic vessels volumetric flow rate

### Scheme of the boundary conditions:

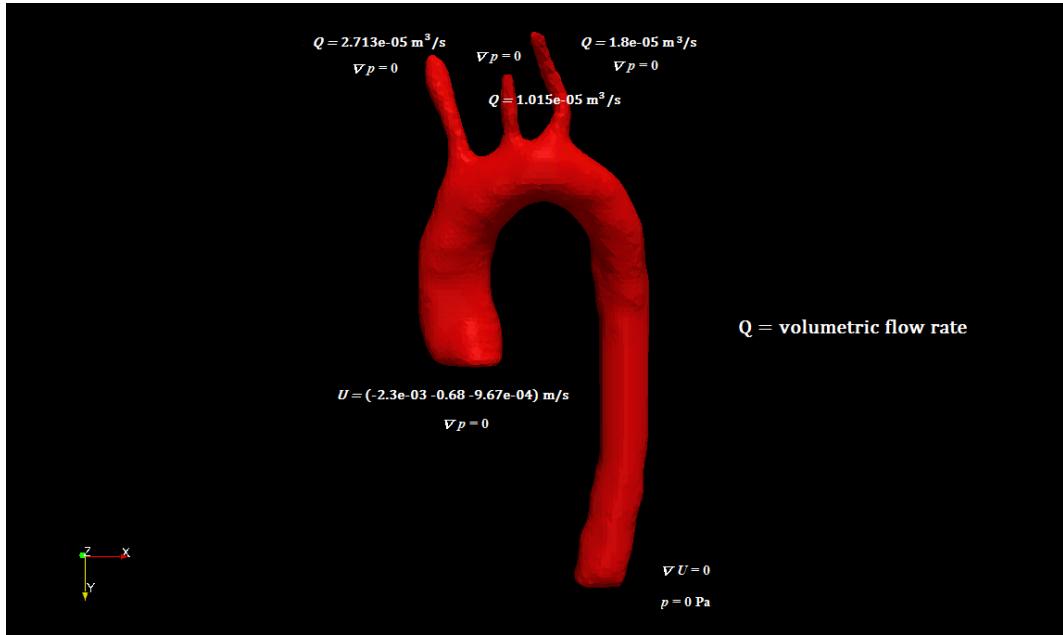


Figure 3.17: Boundary conditions applied to the Second Case

### Results:

As the First Case, it converged. At a first glance, the results seemed to adapt to reality. Before comparing them with the 4D method, it could be seen that they were realistic by observing specific flow structures as the flow detachment at the aortic arch, the behaviour of the streamlines and the velocity range, which was maintained within expected values.

Taking all this into account, the boundary conditions used in the Second Case were adopted as the good ones for simulating the main case.

### 3. Third Case: uniform supra-aortic vessels pressure (equal to the value at the aorta outlet)

#### Scheme of the boundary conditions:

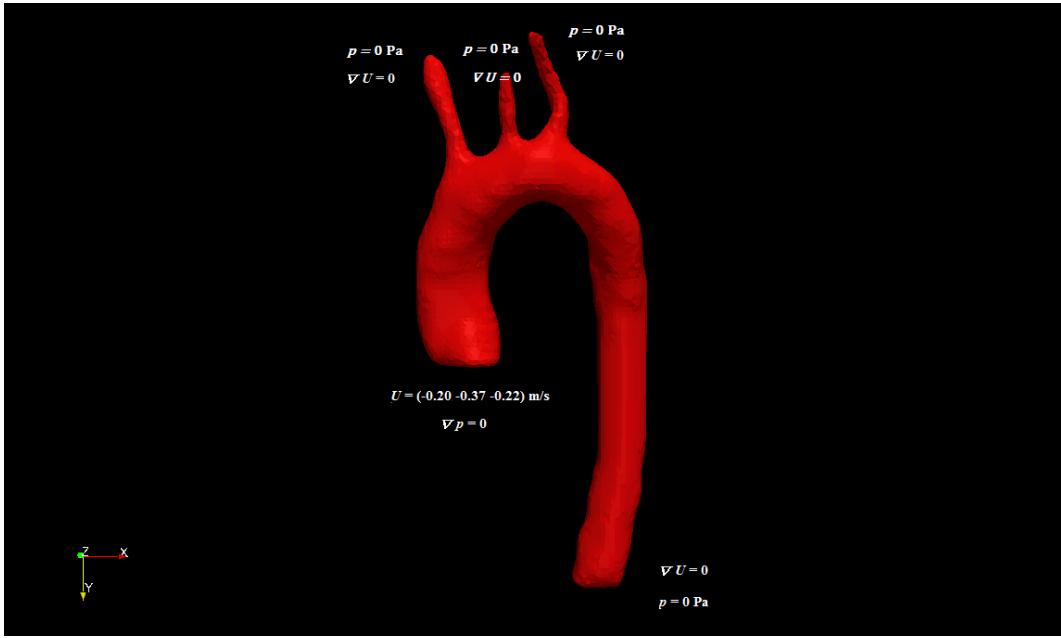


Figure 3.18: Boundary conditions applied to the Third Case

### Results:

This kind of boundary conditions were known to be erroneous (as it has previously been explained). However, the Third Case had been simulated in order to observe which differences exist with respect to the Second Case. Furthermore, it would be possible to estimate how big is the error by assuming that all the outlets are at the same external pressure.

In the following table, it is possible to observe the distribution of flow rate depending on whether the boundary conditions of the Second Case or the Third Case are used:

Table 3.1: Percentage of flow rate in each vessel with respect to the inlet flow rate

| Patch      | Second Case | Third Case |
|------------|-------------|------------|
| <b>S01</b> | 10.47 %     | 13.73%     |
| <b>S02</b> | 3.92 %      | 4.20%      |
| <b>S03</b> | 6.95 %      | 6.56%      |

The differences between the flow behaviour in each case can be observed in Figures 3.19 and 3.20:

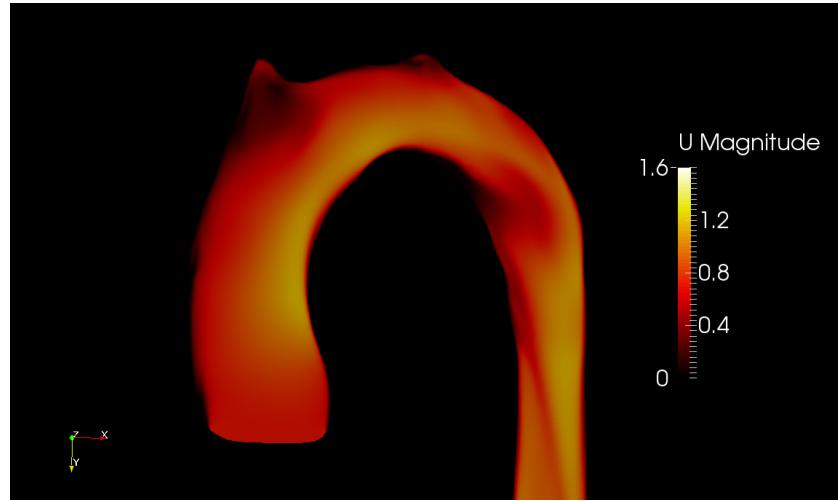


Figure 3.19:  $|\mathbf{U}|$  in a section of the Second Case

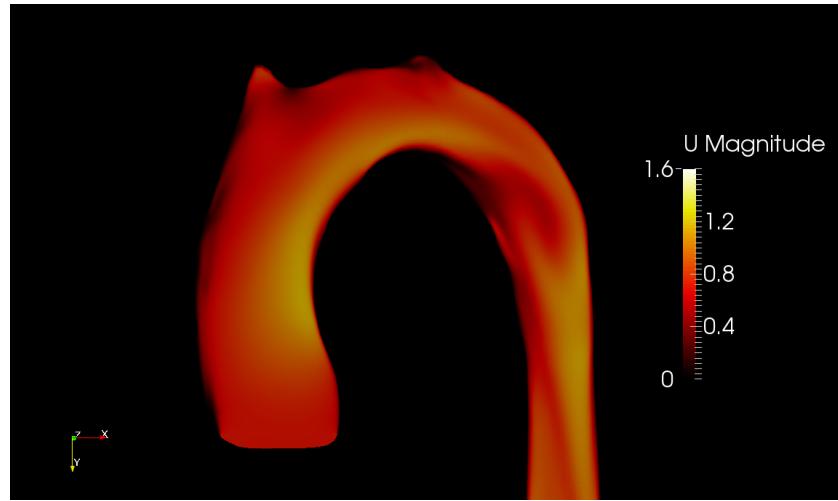


Figure 3.20:  $|\mathbf{U}|$  in a section of the Third Case

As can be seen, although in the Third Case the boundary conditions are not appropriate, both the flow rate distribution and the flow behaviour are not very far from the results obtained in the Second Case. Perhaps the main difference comes from the fact that in the first supra-aortic vessel (patch S01) the fluid is being more strongly suctioned and therefore the flow rate increases with respect to the expected one. This higher suction can be explained by considering that in the Third Case the pressure at the outlets is lower than what is required and thus the suction force increases. However, as the values shown in Table 3.1 are not significantly different, and glancing at Figures 3.19 and 3.20 it can be appreciated that the main

flow structure is similar, it might be hypothesized that the pressure drop between the supra-aortic vessels and the outlet of the thoracic aorta is not as different as it would be initially thought.

### Computation of the boundary conditions in the Second Case:

As explained in Chapter 2, the inlet velocity is not uniform. As the flow is being pumped from the heart by a valve with a specific geometry, the inlet velocity profile presents a similarity with this geometry, as shown in Figure 3.21:

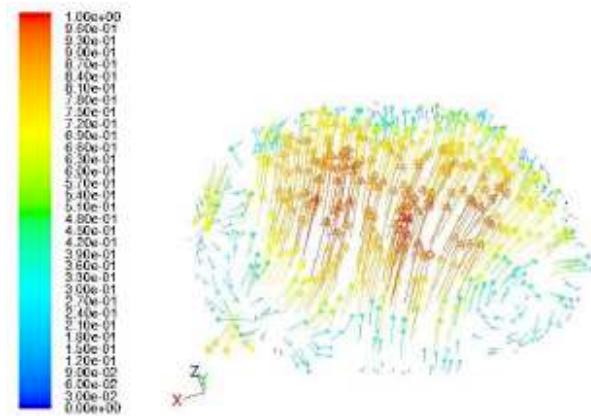


Figure 3.21: Real inlet velocity profile at the aorta inlet (m/s), extracted from [22]

This characteristic velocity profile is produced by the effect of the valve opening in the middle. However, once the fluid enters the aorta it rapidly acquires a parabolic velocity profile, so imposing a uniform velocity at the inlet does not significantly affect the results of the simulation.

For the computation of the inlet velocity vector, the following steps were followed:

- As the Second Case is based on imposing outlet flow rates in the supra-aortic vessels, the inlet flow rate needs to be set accordingly. By considering the real measures obtained from the 4D method, the inlet volumetric flow rate for that aorta geometry is 259.14 ml/s
- As the velocity boundary condition needs to be a vector and a standard model was considered, its direction was set parallel to the normal vector of the **inlet** patch
- The area vector of patch **inlet** was computed by using OpenFOAM utilites and is equal to  $\vec{A} = (-1.2916 \times 10^{-6} \hat{x} + 3.7814 \times 10^{-4} \hat{y} - 5.33789 \times 10^{-7} \hat{z}) m^2$ . Its module is  $\|\vec{A}\| = 4.31163 \times 10^{-4} m^2$

- (d) The module of the main inlet velocity was computed considering the **inlet** area and the volumetric flow rate:

$$U_{mean} = k_c \frac{Q_{inlet}}{\|\vec{A}\|} \quad (3.1)$$

where  $k_c$  is a constant that needs to be included because the inlet patch is not completely bidimensional:

$$k_c = \frac{\text{flux required}}{\text{flux of a 3D inlet}} = 0.42$$

This constant was necessary because when the simulation was carried out with an inlet velocity according to Equation 3.1 without  $k_c$ , the inlet flow rate computed after the simulation was higher than expected. It was concluded that this was due to the fact that the **inlet** patch was not completely two-dimensional.  $k_c$  then expresses the relation between the flow rate that was erroneously obtained and the required volumetric flow rate (259.14 ml/s).

- (e) The area vector of patch **inlet** was normalized so that  $\vec{n} = \frac{1}{\|\vec{A}\|} \vec{A} = (-2.9956 \times 10^{-3} \ 0.877 \ -1.238 \times 10^{-3})$
- (f) Finally, the velocity vector which was used as boundary condition was expressed according to:

$$\overrightarrow{U_{mean}} = U_{mean} \cdot \vec{n} = (-2.34 \times 10^{-3} \ -0.68 \ -9.67 \times 10^{-4}) \text{ m/s}$$

In the following figure it is possible to observe the uniform inlet velocity profile used as inlet boundary condition, being applied parallel to the **inlet**'s patch normal vector for a realistic simulation:

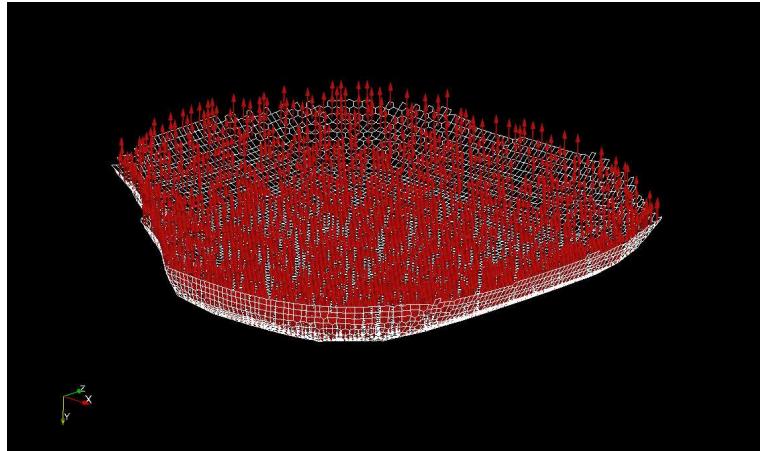


Figure 3.22: Aorta inlet velocity profile. Vectors perpendicular to the patch

Appendices A.9 and A.10 show dictionaries of the pressure ( $p$ ) and velocity ( $\mathbf{U}$ ). By way of example, the instructions of the pressure boundary conditions at the `inlet` and the `outlet` are next shown:

```

1      inlet
2      {
3          type          zeroGradient;
4      }
5
6      outlet
7      {
8          type          fixedValue;
9          value         uniform 0;
10     }

```

and the velocity boundary conditions at the `inlet` and the `S01`:

```

1      inlet
2      {
3          type          fixedValue;
4          value         uniform (-2.3407e-03 -0.6853 -9.674e-04);
5      }
6      S01
7      {
8          type          flowRateInletVelocity ;
9          volumetricFlowRate -2.713e-05;
10         value uniform (0 0 0);
11     }

```

### Main difficulties:

- Initially, the First Case and the Third Case were used, as they represent standard types of boundary conditions (imposing either a velocity or a pressure). Once it could be seen that none of them offered suitable results, it took a long time to test new models and boundary conditions. Finally, the volumetric flow rate condition was implemented, and its results became the most valid
- Due to the fact that a volumetric flow rate is not commonly used as a boundary condition, its validity had to be studied and proved
- Initially, the flow rate distribution did not match reality because the velocity measures obtained from the 4D method had to be adapted to the OpenFOAM aorta patches (as they are differently defined). Accuracy with the proportion of flow rates in each patch was crucial in order to obtain suitable results and to simulate the aorta as realistically as possible
- The first method used to compute the flow rate in each vessel suggested that the flow is parabolic within the aorta when it is fully developed. Then, for parabolic velocity profiles,  $U_{mean} = \frac{U_{max}}{2}$ , and by using each patch area, the volumetric flow rate was computed. However, by using this method, the flow rate distribution did not offer realistic results and the process of computing the boundary conditions had to be changed

### 3.3 Physical properties

**Aim of the task:** the codes shown in this section complete the physical information of the simulations (together with the boundary conditions). Mainly, it is necessary to specify the kinematic viscosity and the nature of the case (laminar or turbulent).

**Work methodology:** as has been explained in Section 2.4, in this work blood is considered a Newtonian fluid (and therefore with constant viscosity). This information, as well as the value of the kinematic viscosity is included in the *transportProperties* file, in *constant*. It is shown in Appendix A.11.

On the other hand, as the case is laminar, no turbulence models had to be implemented. This information is included in the *RASProperties* file (Appendix A.12), also located within *constant*.

---

## 3.4 Control, discretization and linear-solver setting

**Aim of the task:** the codes shown in this section contain information related to the time control, the discretization and the linear-solver settings. They are mainly related to the mathematical treatment of the simulation and play a relevant role on its convergence.

**Work methodology:** to define these parameters, 3 different dictionaries were used:

1. **controlDict:** it specifies the solver of the case and how many time steps are used. Again, as peak systolic conditions are modeled (steady flow), `simpleFoam` has been used. `simpleFoam` is a steady-state solver for incompressible, turbulent flow. The `controlDict` dictionary is shown in Appendix A.13.
2. **fvSchemes:** it specifies the choice of finite volume discretization schemes (Appendix A.14).
3. **fvSolution:** it specifies the linear equation solvers, tolerances and other algorithm controls (Appendix A.15).

## 3.5 Justification of the mesh refinement

**Aim of the task:** as explained in Section 3.1.5, a very accurate cell refinement was obtained at the walls of the aorta (of the order of micrometers). However, as good as it may seem, it was necessary to estimate the error made when computing the wall shear stress. Only if the error stays bounded to low values, this refinement could be considered suitable.

**Work methodology:** as the real values of wall shear stress at the walls of the aorta are unknown (in fact, this is the aim of the work), there is no way of contrasting the results of the simulation. Therefore and with all due caution, an analogue of the aorta such as a circular pipe was modeled and simulated.

Although the geometries of a healthy aorta and a straight, large, circular pipe are different, by imposing the same cell refinement, boundary conditions, Reynolds number and volumetric flow rate, the comparison could provide acceptable conclusions regarding the error made by using this mesh refinement.

It could be done because in a large straight circular pipe the exact value of the wall shear stress can be analytically computed.

In Figure 3.23, a wedge of the pipe is shown with its values of refinement.  $y$  is the distance between the pipe wall and each node:

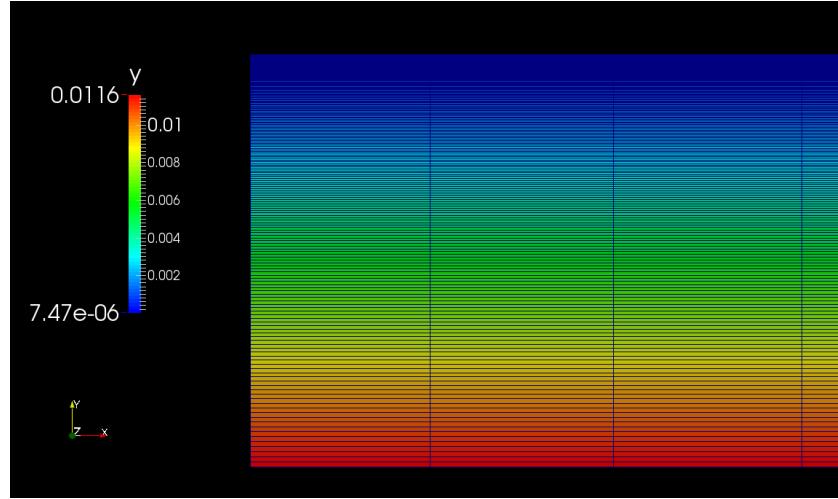


Figure 3.23: Wedge of the circular pipe used for testing the mesh refinement

Mathematically, the wall shear stress in a large, circular pipe is (in cylindrical coordinates):

$$\tau_w = \mu \frac{\partial U}{\partial r} = - \frac{dp}{dz} \frac{r}{2} \quad (3.2)$$

where  $r$  is the radius of the pipe. According to the boundary conditions, the analytical wall shear stress is:

$$\tau_w = 235.2355 \cdot 5.855 \times 10^{-3} = 1.37730 \text{ Pa}$$

On the other hand, the wall shear stress computed by OpenFOAM with a mesh refinement at the walls similar to the one used for the aorta simulation is:

$$\tau_w = 1.37599 \text{ Pa}$$

By comparing both values, a relative error of  $e_r = 0.095\%$  is obtained.

**Conclusion:** the relative error obtained is very low. Consequently, although the geometry of a real aorta is different and the behaviour of the flow is more

---

complex (there are detachments and vortices), the mesh refinement can be considered adequate. It can be concluded that although in the real simulation the average error may increase, since the relative error in the pipe simulation is extremely low, the real wall shear stress value will stay bounded into a suitable range.

## 4. Results of the aorta simulation

In the current chapter, the results of the simulation are presented, carried out according to the pre-processing instructions, processes and dictionaries included in Chapter 3.

### 4.1 Module of the velocity field

First of all, plots of the velocity field are displayed. In the following figures, the module of the velocity ( $|\mathbf{U}|$ ) in each point is shown:

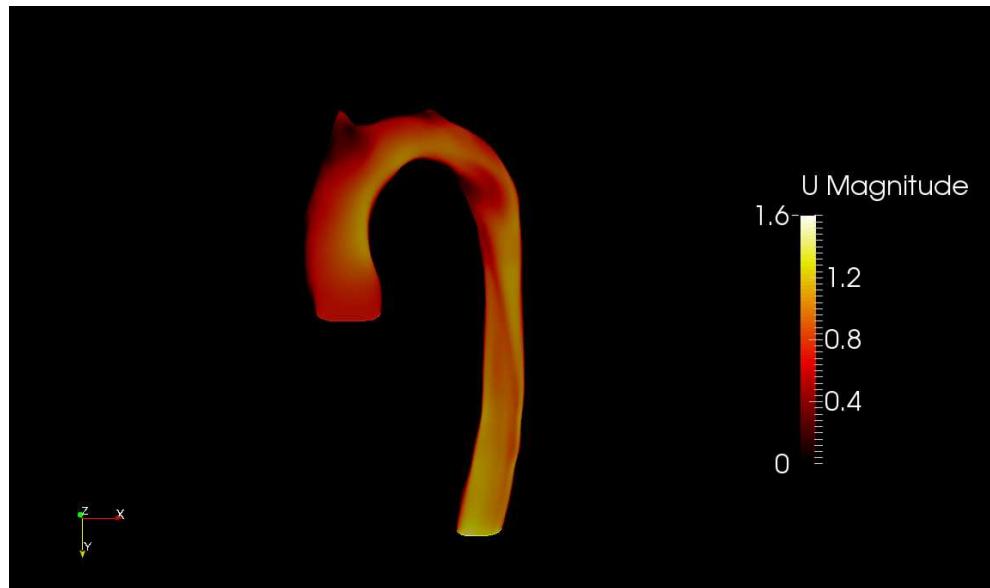


Figure 4.1:  $|\mathbf{U}|$  in a general longitudinal slice of the aorta (m/s)

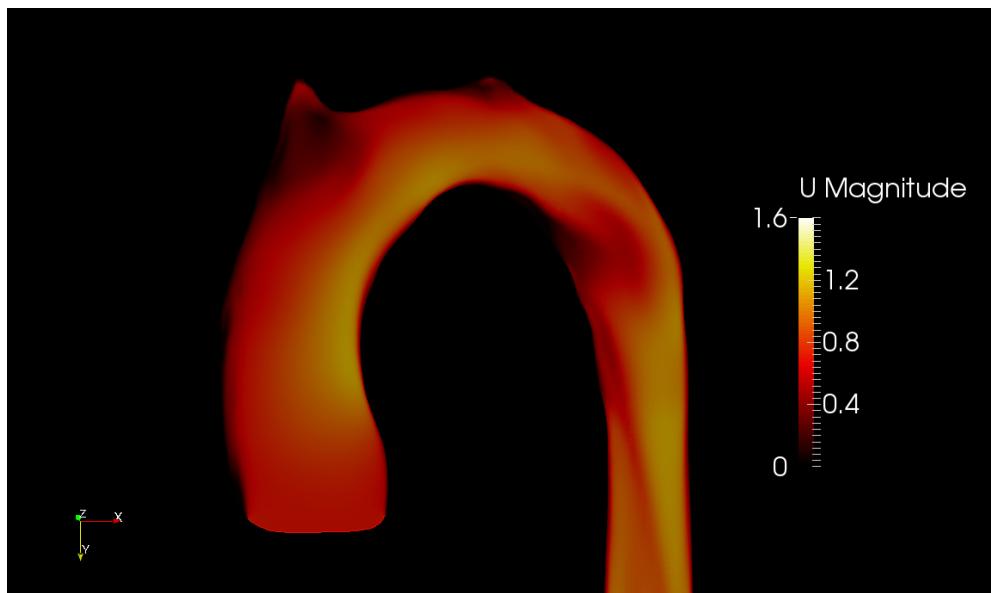


Figure 4.2:  $|\mathbf{U}|$  in a longitudinal slice of the aortic arch (m/s)

Figures 4.3 and 4.4 show a longitudinal slice of the aorta with different transverse sections to adequately observe the behaviour of the flow in the whole geometry:

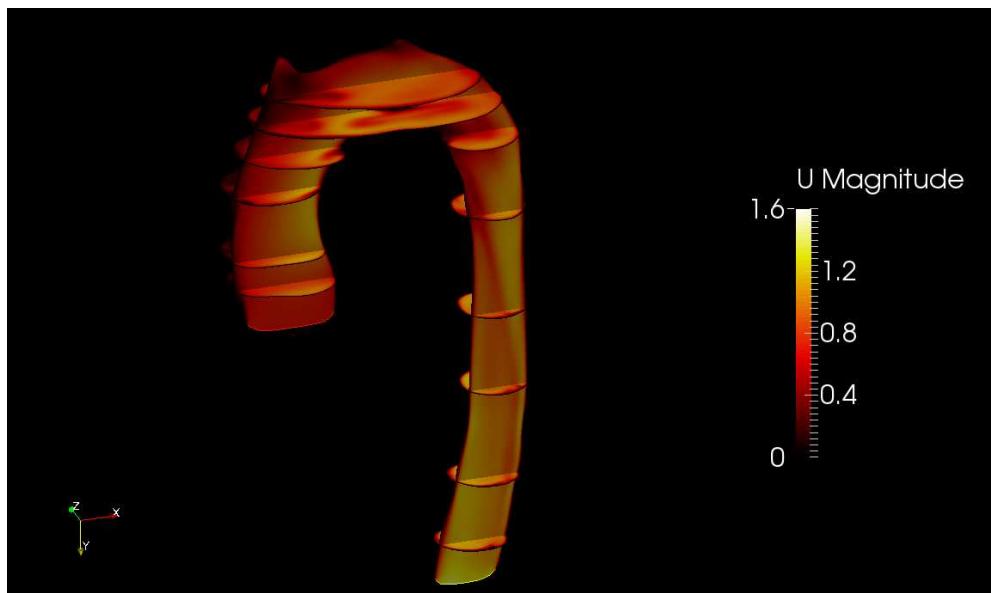


Figure 4.3:  $|\mathbf{U}|$  in a longitudinal slice of the aorta from the front part (m/s)

#### 4. Results of the aorta simulation

---

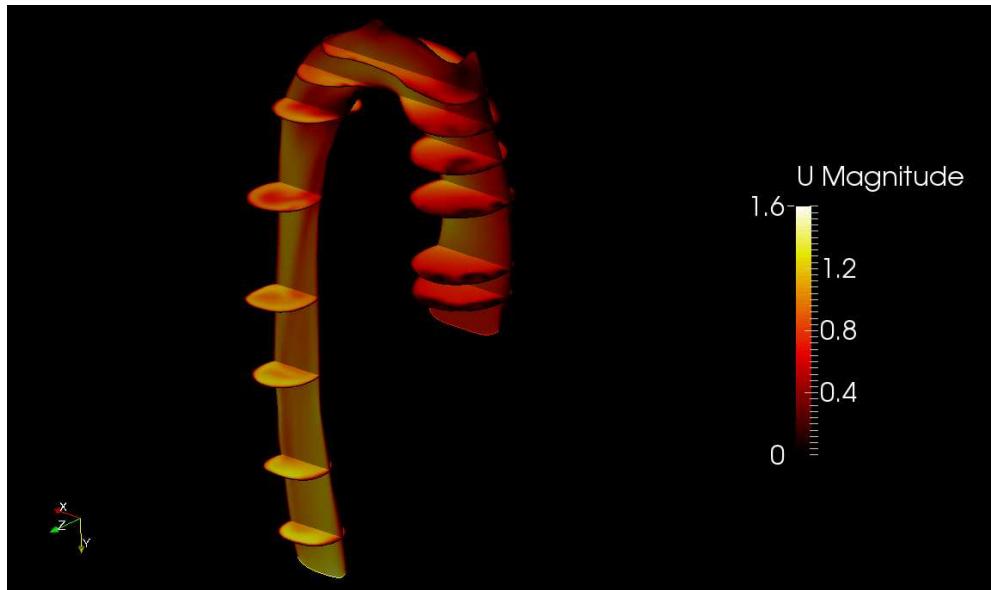
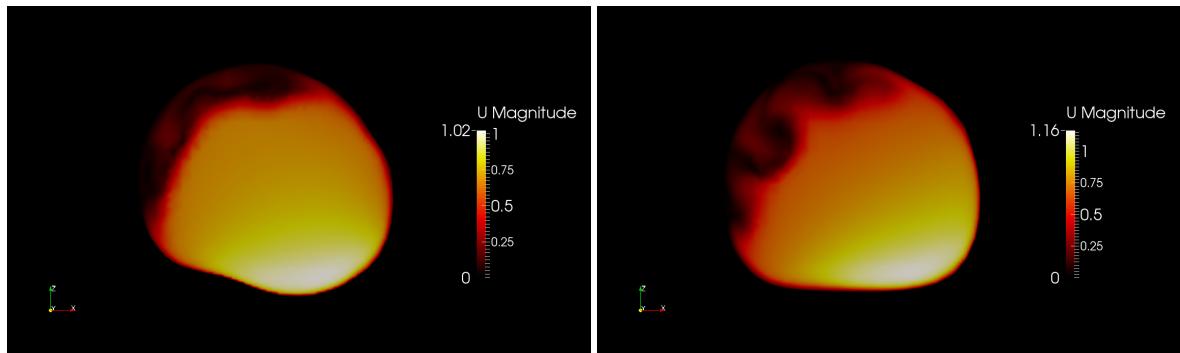


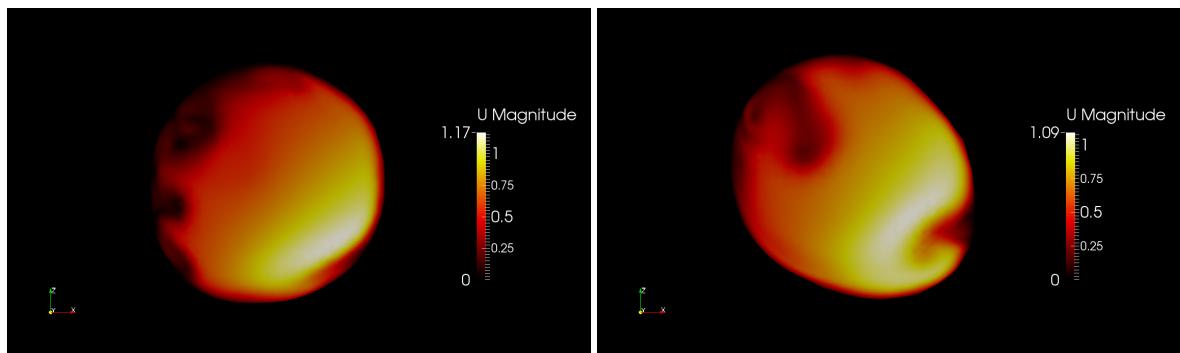
Figure 4.4:  $|U|$  in a longitudinal slice of the aorta from the back part (m/s)

Each one of the transverse slices previously shown is exposed in more detail (they are presented in an order such that they start at the inlet and end at the outlet):



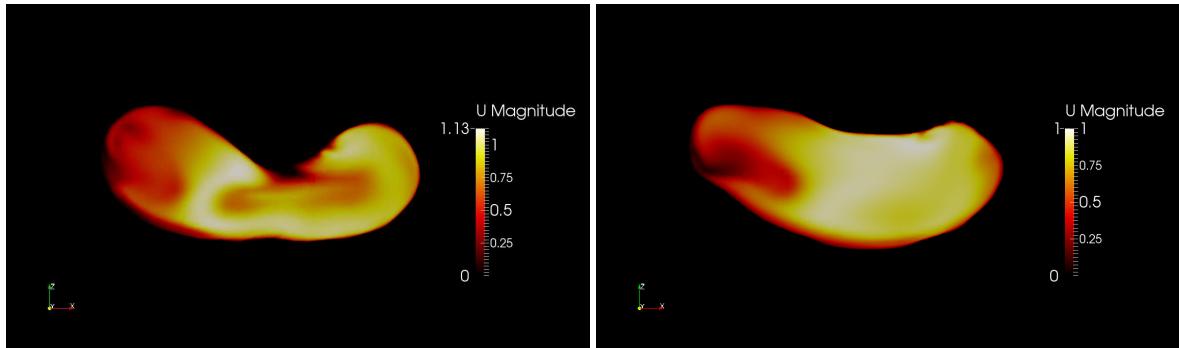
Velocity module in slices of the aorta simulated with OpenFOAM. Slices at

$$y = 0.005 \text{ and } y = 0.039$$

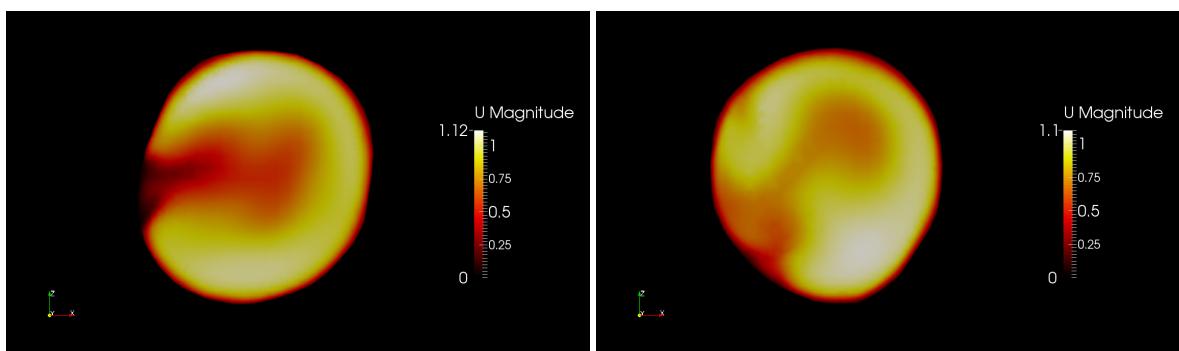


---

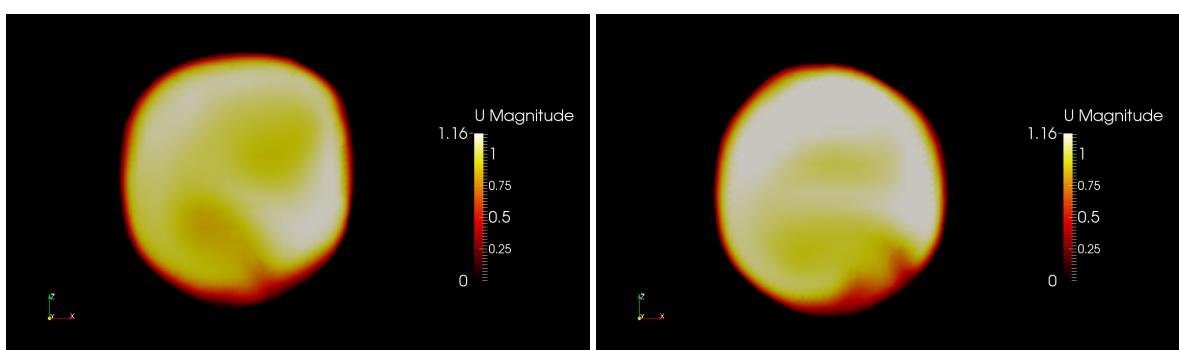
Velocity module in slices of the aorta simulated with OpenFOAM. Slices at  
 $y = 0.022$  and  $y = 0.013$



Velocity module in slices of the aorta simulated with OpenFOAM. Slices at  
 $y = 0.005$  and  $y = -0.002$



Velocity module in slices of the aorta simulated with OpenFOAM. Slices at  
 $y = 0.014$  and  $y = 0.036$

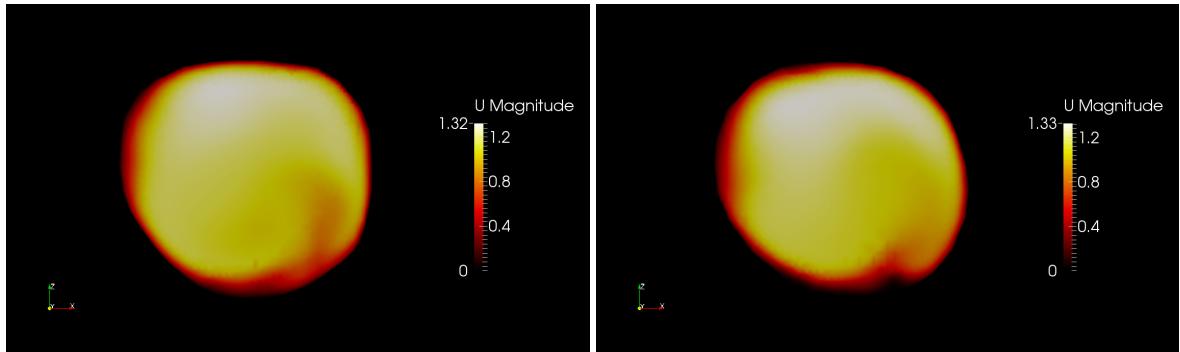


Velocity module in slices of the aorta simulated with OpenFOAM. Slices at  
 $y = 0.062$  and  $y = 0.082$

---

#### 4. Results of the aorta simulation

---



Velocity module in slices of the aorta simulated with OpenFOAM. Slices at  
 $y = 0.105$  and  $y = 0.121$

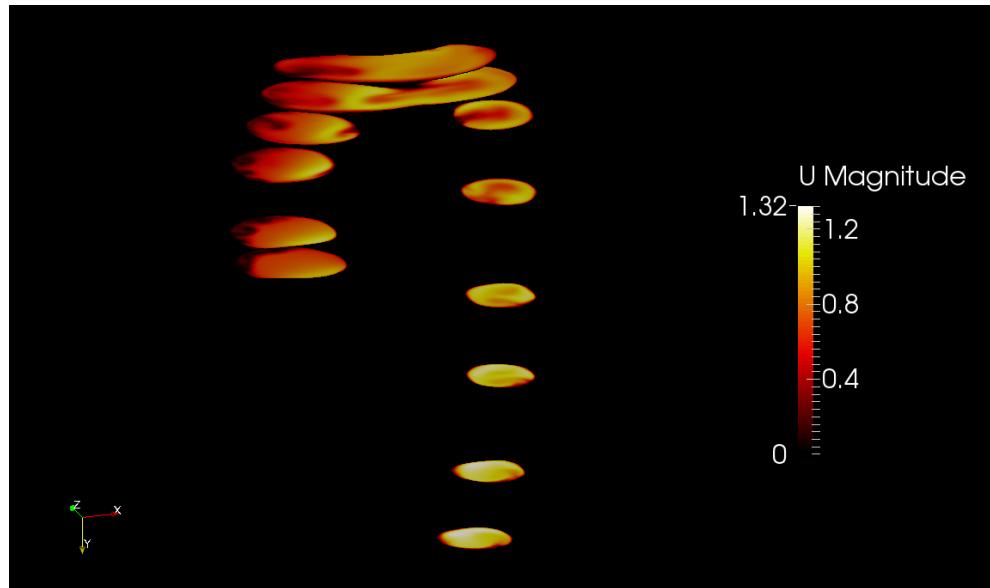


Figure 4.5:  $|\mathbf{U}|$  in transverse slices seen from the front part (m/s)

---

## 4.2 Streamlines

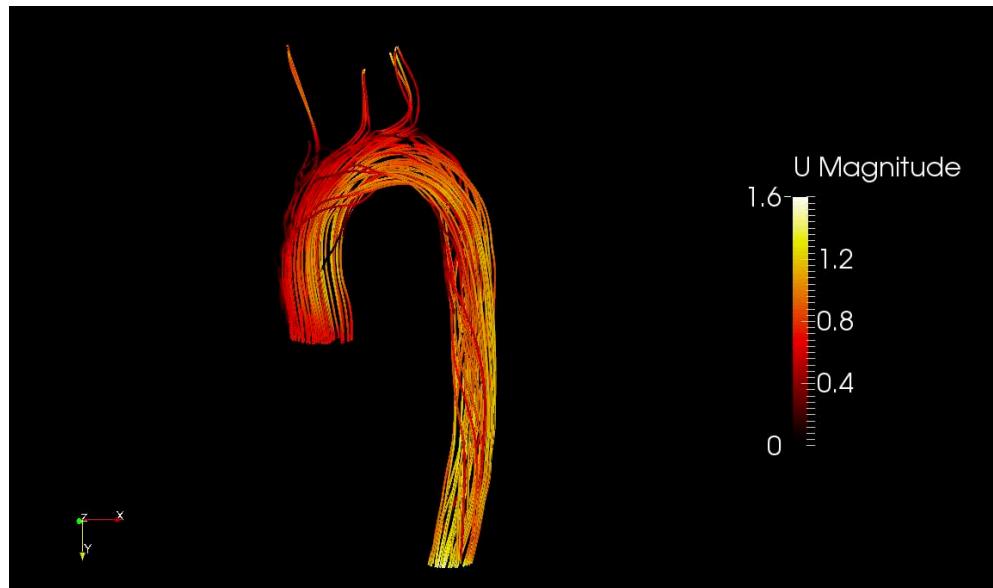


Figure 4.6: Streamlines of the flow seen from the front part (m/s)

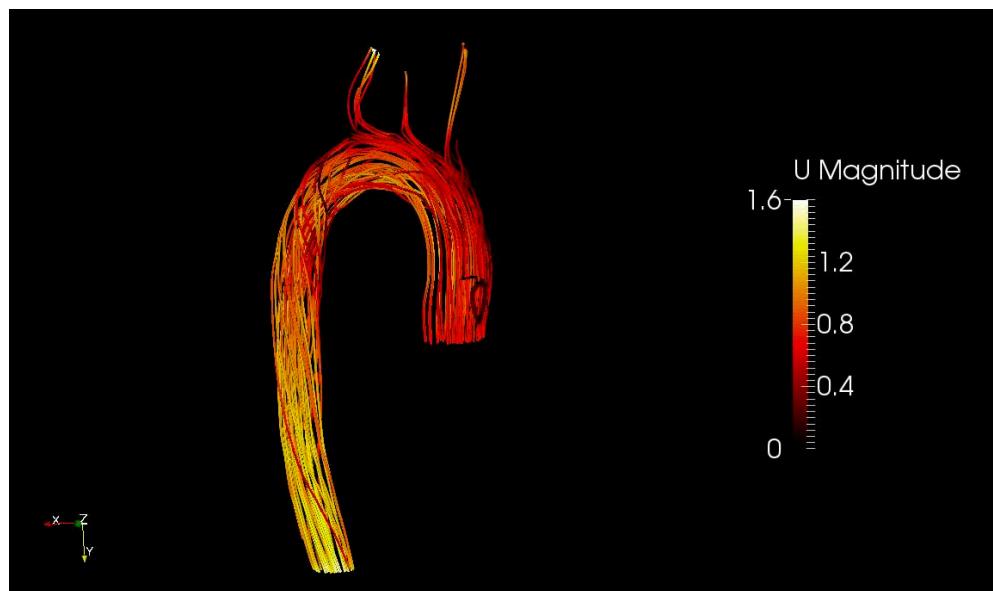


Figure 4.7: Streamlines of the flow seen from the back part (m/s)

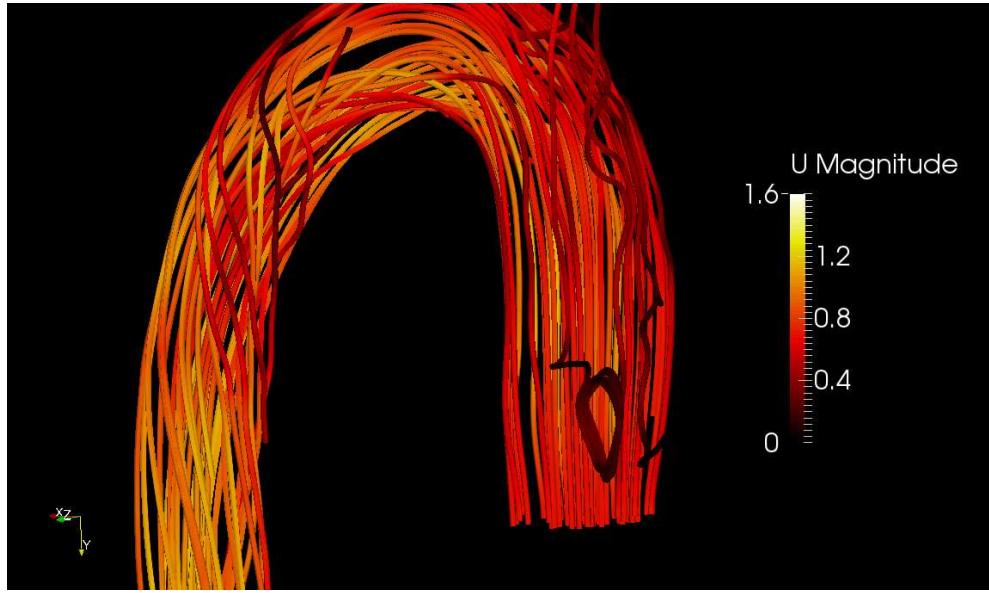


Figure 4.8: Detail of the streamlines of the flow at the inlet and the aortic arch (m/s)

Two interesting characteristics to observe are the vortices generated at the inlet and the torque component of the velocity after the aortic arch.

### 4.3 Vector field

The module of the vectors shown in the following figures does not correspond to the magnitude of their velocity, which is represented by their color. This has been done for a better understanding. The inlet vortices can be observed again in Figure 4.10:

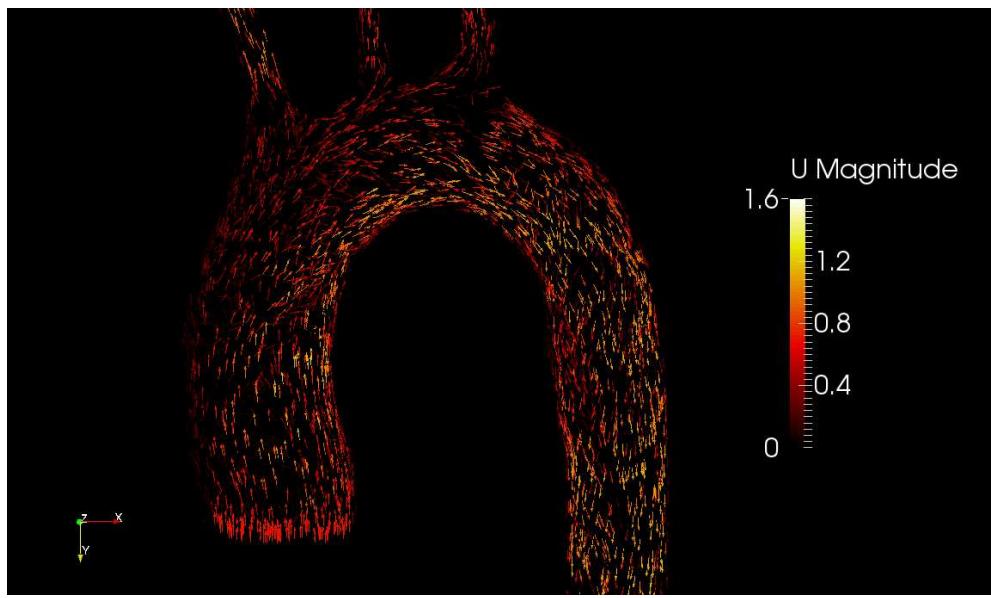


Figure 4.9: Velocity vectors at the aortic arch (m/s)

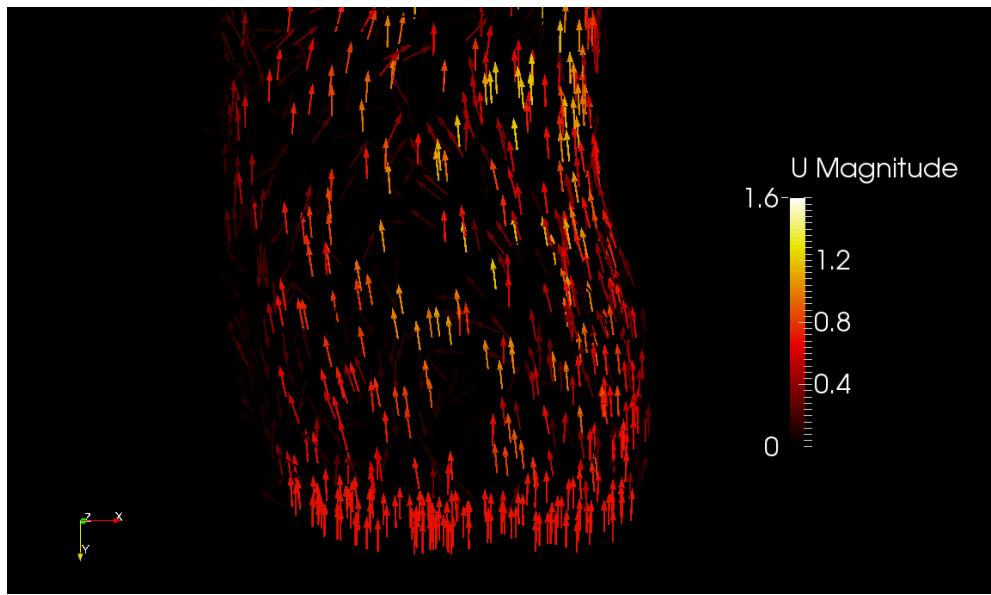


Figure 4.10: Velocity vectors at the aorta inlet (m/s)

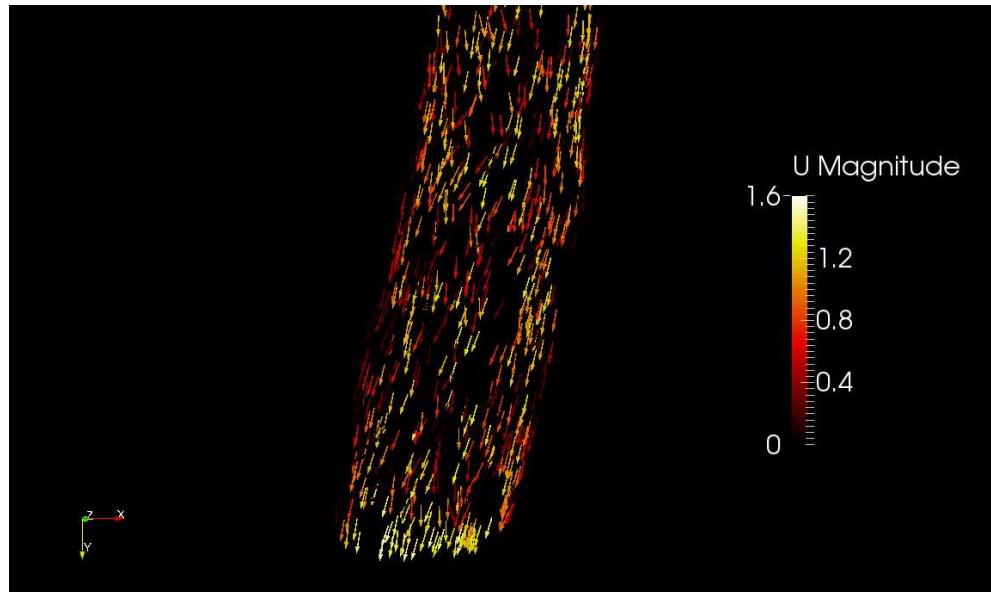


Figure 4.11: Velocity vectors at the aorta outlet (m/s)

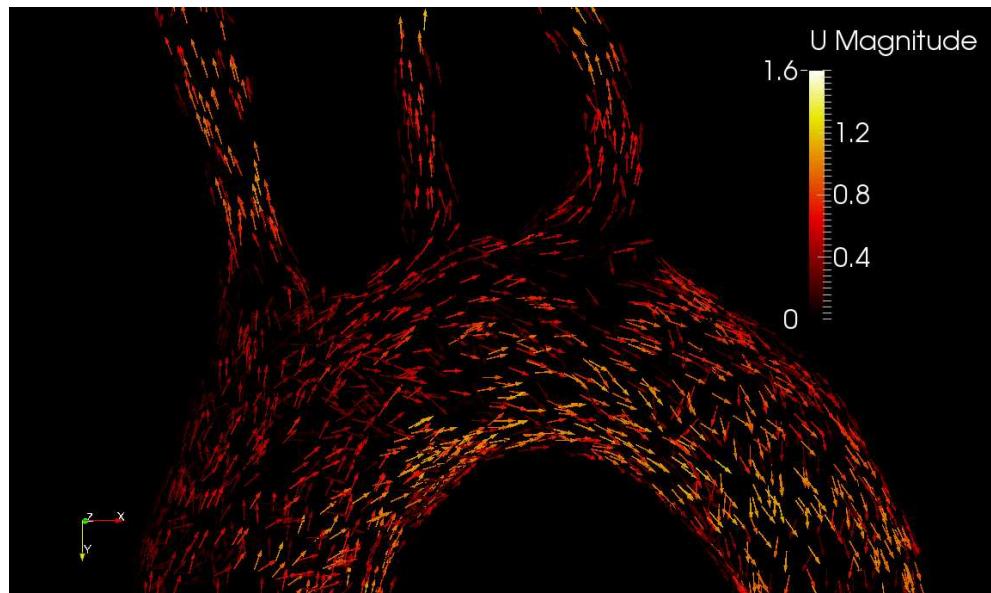


Figure 4.12: Velocity vectors at the inlet of the supra-aortic vessels (m/s)

## 4.4 Wall shear stress

It is first necessary to remember that the values of wall shear stress that OpenFOAM provides are divided by the value of the density ( $\rho = 1040 \text{ kg/m}^3$  for blood) of the fluid:

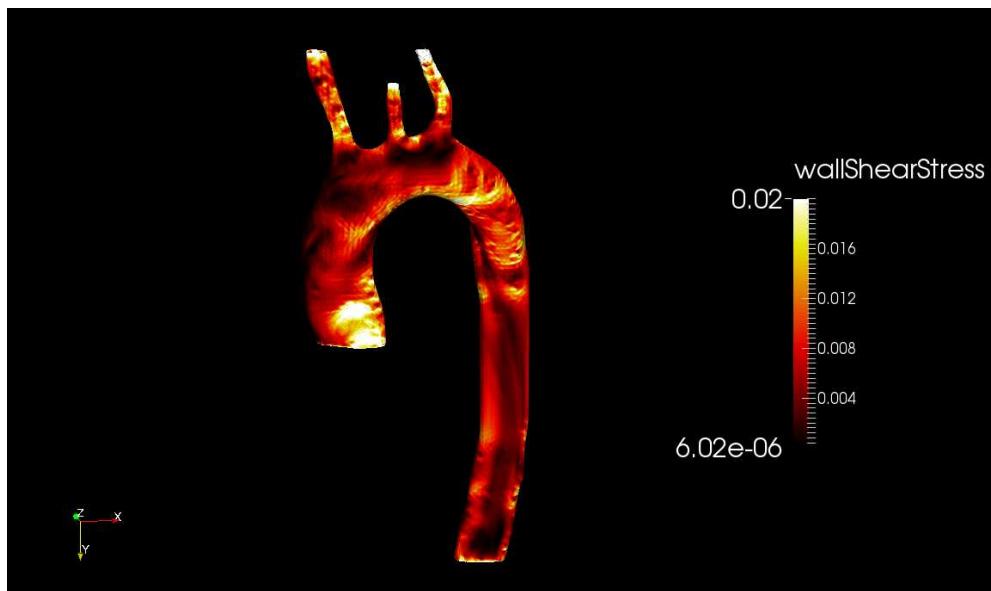


Figure 4.13: Wall shear stress at the front part of the aorta ( $m^2/s^2$ )



Figure 4.14: Wall shear stress at the back part of the aorta ( $m^2/s^2$ )

#### 4. Results of the aorta simulation

---

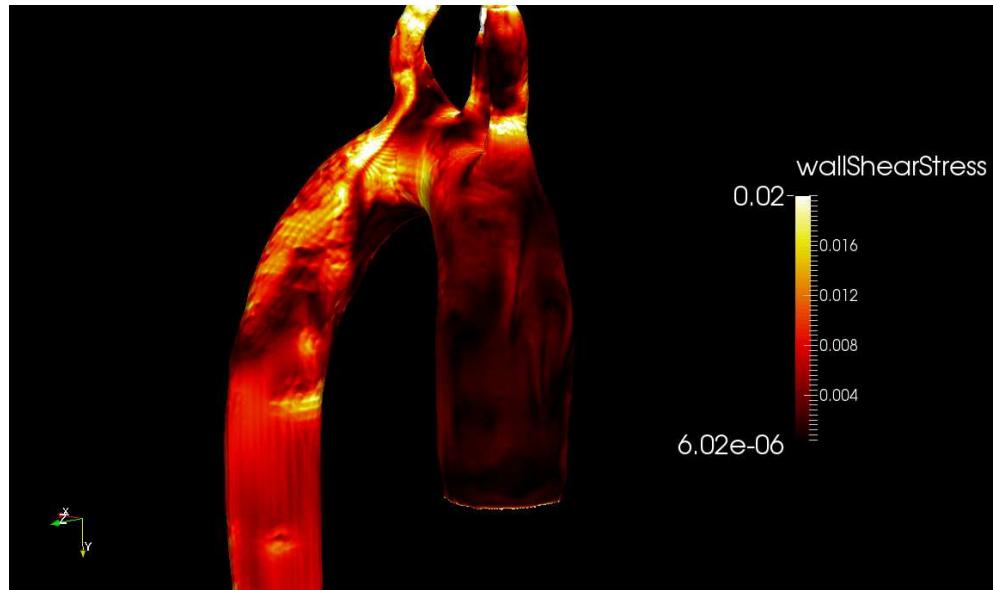


Figure 4.15: Detail of the wall shear stress at the aorta inlet and at the aortic arch ( $m^2/s^2$ )

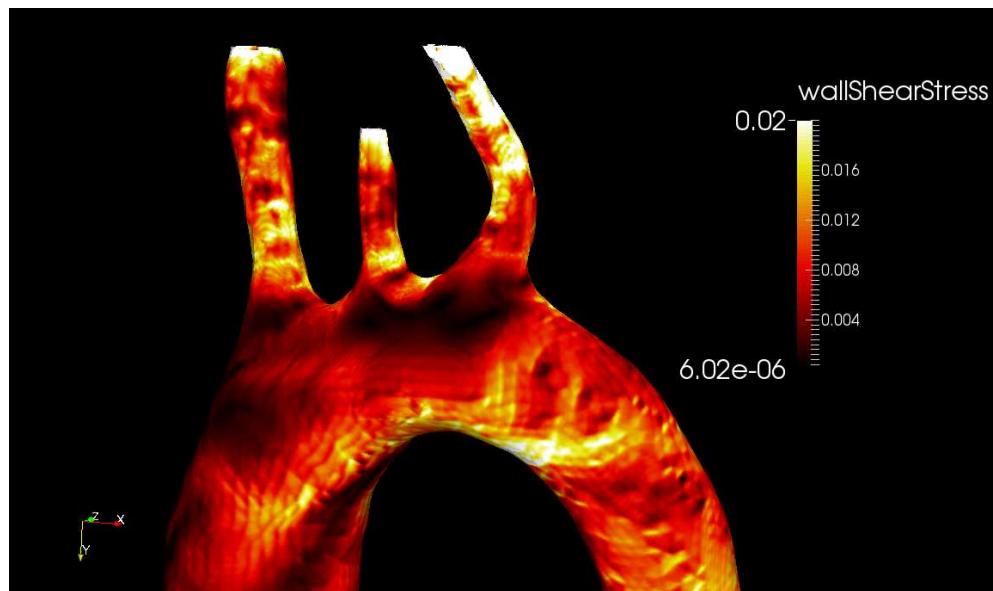


Figure 4.16: Detail of the wall shear stress at the inlet of the supra-aortic vessels ( $m^2/s^2$ )

---

## 5. Analysis of the results

### 5.1 Analysis of the OpenFOAM simulation

#### 5.1.1 Validation of results

As explained in 3.5, real distributions of wall shear stress and flow patterns in healthy thoracic aortas are unknown, thus the results need to be analyzed and compared. The first step is to ensure that the main characteristics of blood through real aortas are met. There are particular aspects in literature (coming either from medical research or other CFD simulations) which are accepted as common flow patterns for standard thoracic aorta geometries and boundary conditions. Thereby, at a first glance, the simulation must resolve adequately the following aspects:

1. Flow detaching on the aortic arch due to an adverse pressure gradient
2. Flow rate through the supra-aortic vessels between 20% and 30% of the inlet flow rate (it was imposed, so it indeed adapts to reality)
3. Wall shear stress distribution compressed between 20 Pa and 0 Pa, with predominance of values around 8 Pa

All the above characteristics have been met with the OpenFOAM simulation, and are fundamental to initially guarantee that it ran properly. Furthermore, the streamlines and the vector field also show a coherent behaviour, and the velocity values stay bounded within a suitable range.

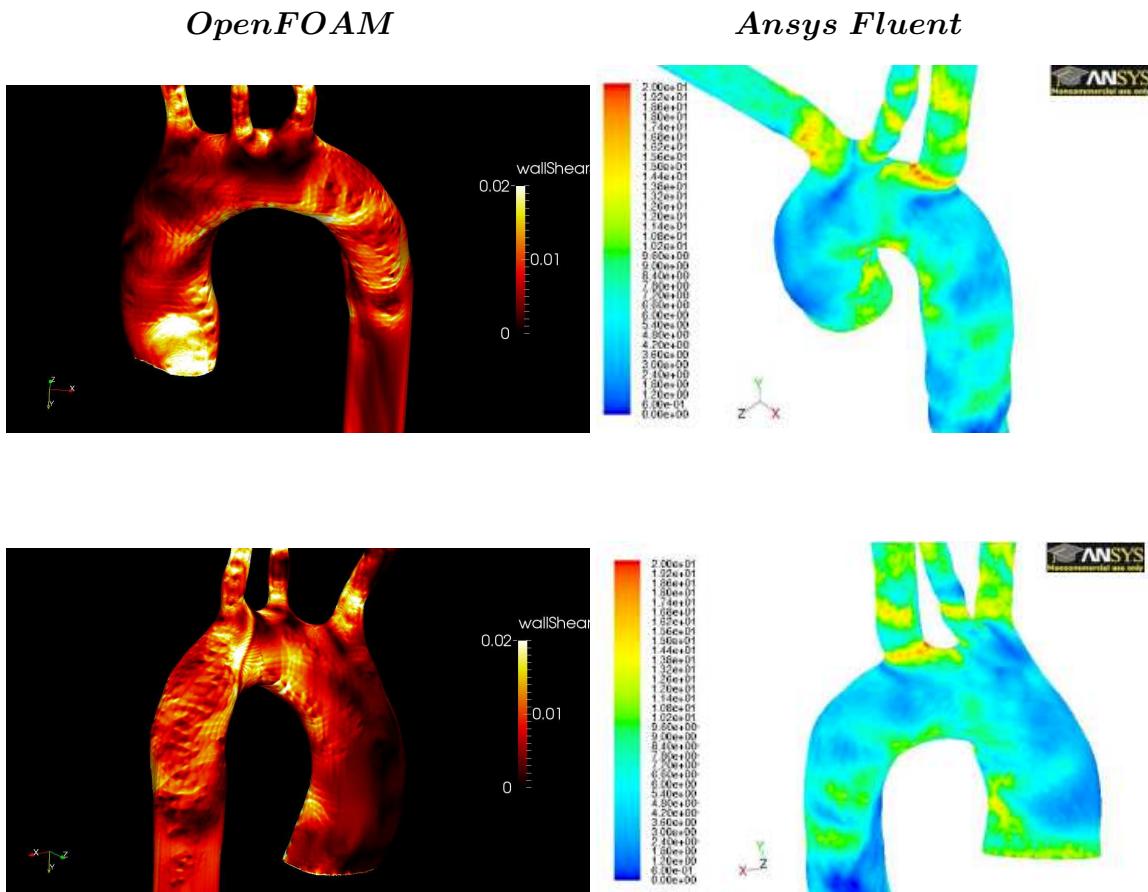
The second step is to compare the results of the OpenFOAM study with another CFD simulation. Accordingly, there is next a comparative between the wall shear stress distribution obtained with OpenFOAM and with Ansys Fluent. The study used for the comparison has been extracted from [22], the one found in the literature which resembles more the present OpenFOAM simulation.

The variable to be compared is the wall shear stress, as its computation represents the main objective of the current aorta study. Furthermore, as the wall shear stress is

## 5. Analysis of the results

---

computed in the post-processing, if its distribution and values seem to adapt to other cases, the velocity field is then likely to also fit correctly. For the comparative, it has been intended to use an external simulation as close to the one carried out in this project as possible. It implies similar geometries (otherwise the wall shear stress distribution might differ although both had been correctly simulated), flat inlet velocity profile, outlet velocity profiles as boundary conditions, laminar Newtonian flow, peak systolic conditions and rigid and static aorta walls. The results are:



Comparison between simulations carried out with OpenFOAM or Ansys Fluent

The units of the wall shear stress in the Ansys Fluent simulation are *Pascals*, while in OpenFOAM are  $m^2/s^2$ .

Comments:

A first factor indicating that the results seem reliable is that both simulations present the range of the wall shear stress compressed between the same values, 0-20 Pa. It implies that the numerical results of both simulations are kept bounded within the same interval

---

Regarding the wall shear stress distribution, both simulations present similar patterns: in the ascending aorta, low values appear on its outer part and high values on its inner part. At the inlet of the supra-aortic vessels there are high flow accelerations, and therefore high values of wall shear stress. At the same time, near these areas and in the aortic arch there are zones presenting low concentrations of wall shear stress. Finally, just at the beginning of the descending aorta there are intermittent areas with high stress values, followed by a dark stain (low values), perhaps due to the flow detachment which occurs in this region

Both cases, using different MRI data, present the same uniform inlet velocity (approximately 0.69 m/s). It corroborates the method used in this project to compute this value (Section 3.2), guarantees that the inlet flow rate matches reality and that the MRI values used in this project are correct

Despite the above, there are some differences between both simulations, such as the fact that the stress at the inlet is higher for the OpenFOAM simulation. It probably happens due to the segmentation of the geometry used in each case. Since for the OpenFOAM simulation the inlet has been defined nearer to the heart, its topology is more adapted to the geometry of the aortic valve. Consequently, the wall shear stress values are higher and less focused

A key factor for the comparison is that the most extended value of wall shear stress is very similar in both cases (the one either in areas without high accelerations or flow detachment, which has been used as a representative value). For the Ansys Fluent simulation, it is represented by turquoise, whose numerical value corresponds at around 6 Pa and 8 Pa. For the case of the OpenFOAM simulation, this representative value may be the soft red, whose numerical value is compressed between  $0.007 \cdot 1040 = 7.28$  Pa and  $0.008 \cdot 1040 = 8.32$  Pa.

### 5.1.2 Discussion of results

In the previous Section, the main results were validated and contrasted. However there are other aspects of the simulation which are worth being discussed:

1. **Inlet vortices:** as it can be appreciated in Figures 4.7 and 4.8, there is a formation of vortices near the inlet of the aorta. They can also be observed in Figure 4.5, being represented by dark stains at the outer part of the ascending aorta. The vortices are formed due to the widening of the section just after the inlet, being a part of a natural mechanism of the circulatory system.

One of the hypothesis of the simulation is that right and left coronary arteries are not included. However, these little vessels can be found in real human bodies and are located just at the inlet of the aorta. The vortices that appear in the simulation are in reality generated during systole, providing rotational kinetic energy to the fluid particles to flow through the coronary arteries during diastole. As the geometry used for the simulation does not include them, vortices extend to the aortic arch.

2. **Flow detachment:** flow detaches at the aortic arch, which among other issues affects the velocity profile at the descending aorta. As it can be observed in Figure 4.5, the velocity distribution is parabolic, but with a distinct region presenting low flow speeds due to the detachment. This becomes smooth as the stream descends through the aorta.
3. **Spin flow:** due to the fact that the ascending and descending aorta are not contained within the same plane, a spin component in flow can be observed in the descending aorta. The streamlines of Figure 4.8 offer a clear-cut representation of this pattern.

## 5.2 Comparison between the CFD OpenFOAM simulations and the 4D medical images

Once the results of the OpenFOAM simulation have been analyzed, the results of the 4D method are introduced.

First of all, a longitudinal slice is presented (speed values of the 4D images are expressed in cm/s):

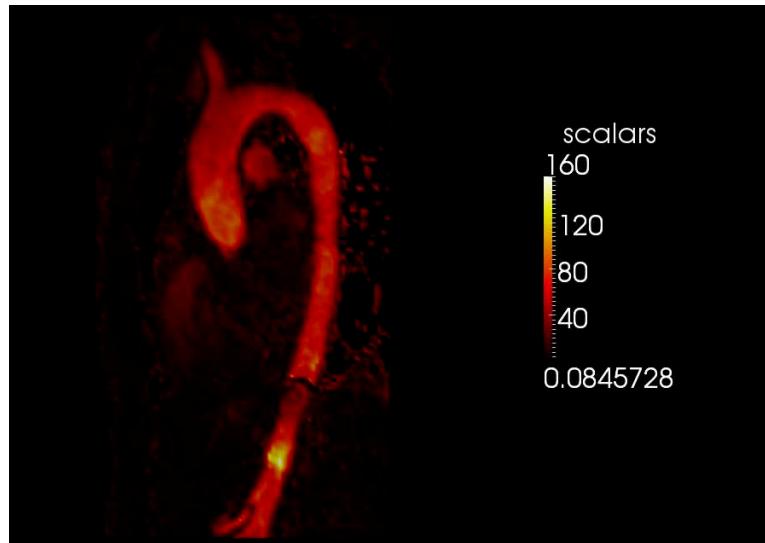
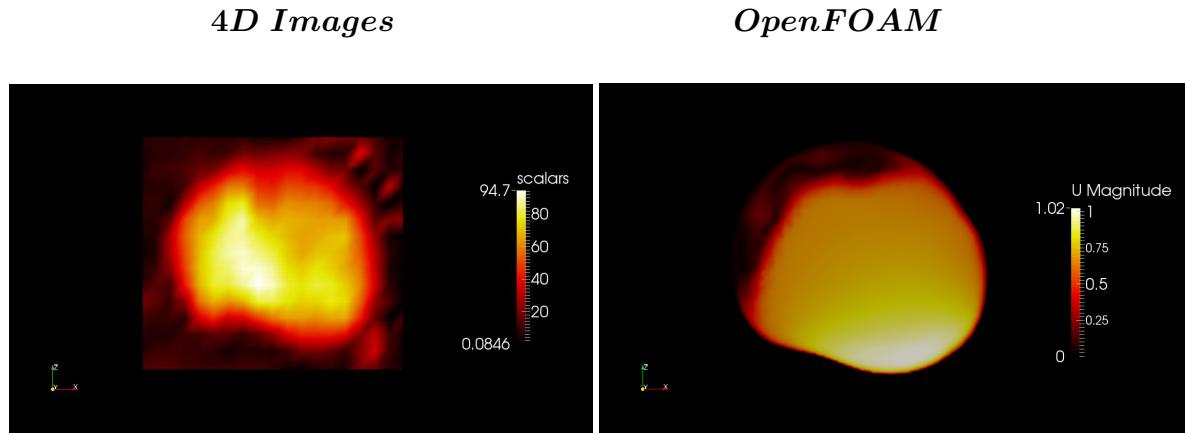
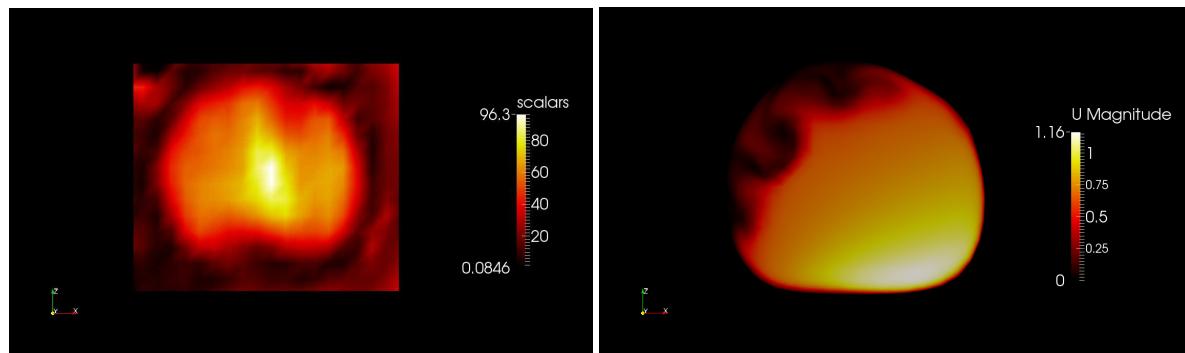


Figure 5.1:  $|\mathbf{U}|$  in a longitudinal slice of the aorta obtained from the 4D method (cm/s)

Next, the same transverse slices presented in Section 4.1 are shown:



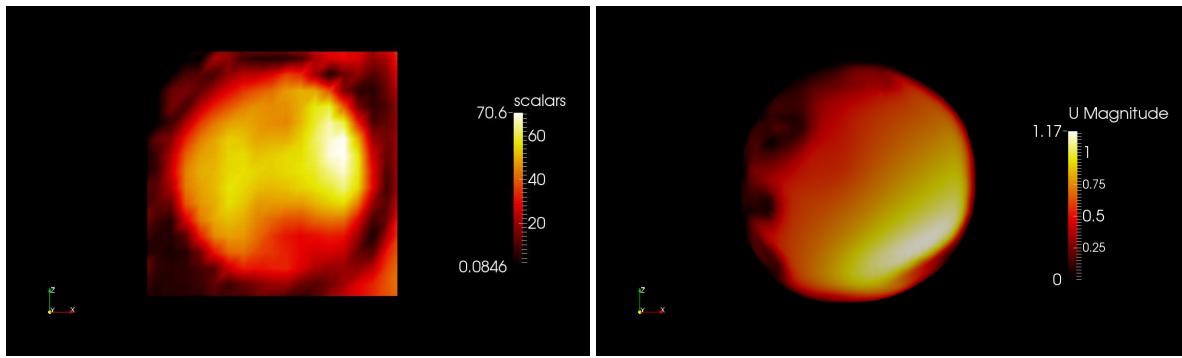
Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.046$



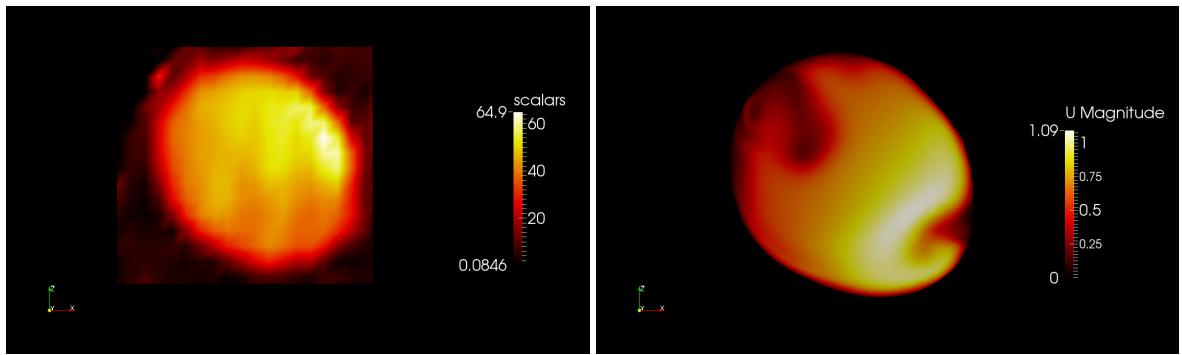
## 5. Analysis of the results

---

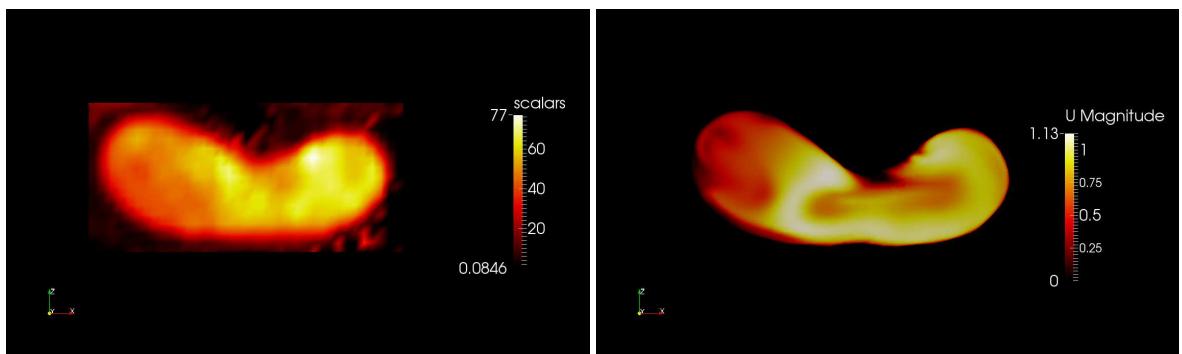
Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.039$



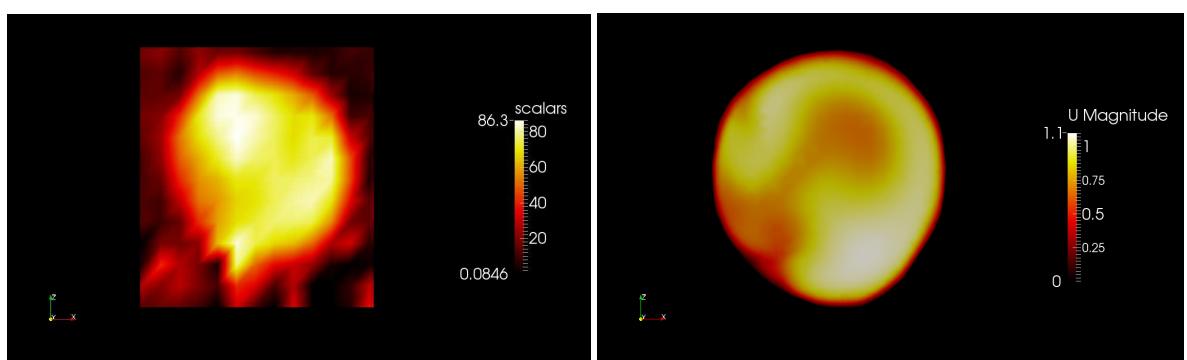
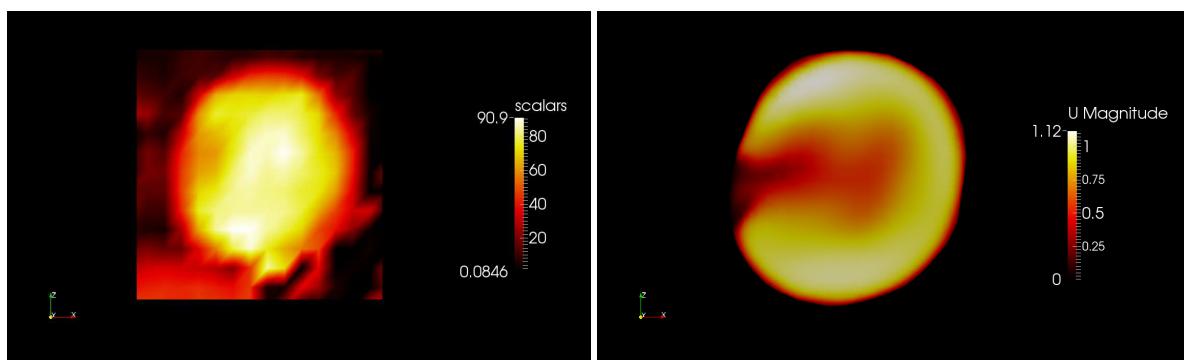
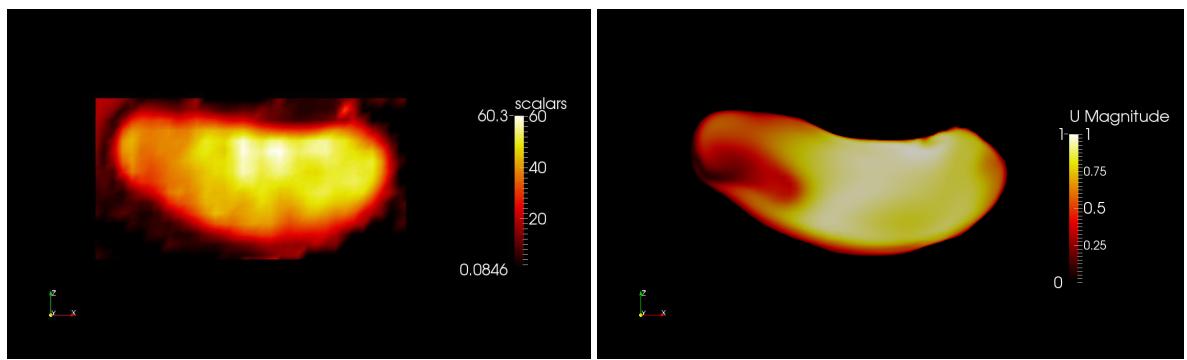
Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.022$



Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.013$

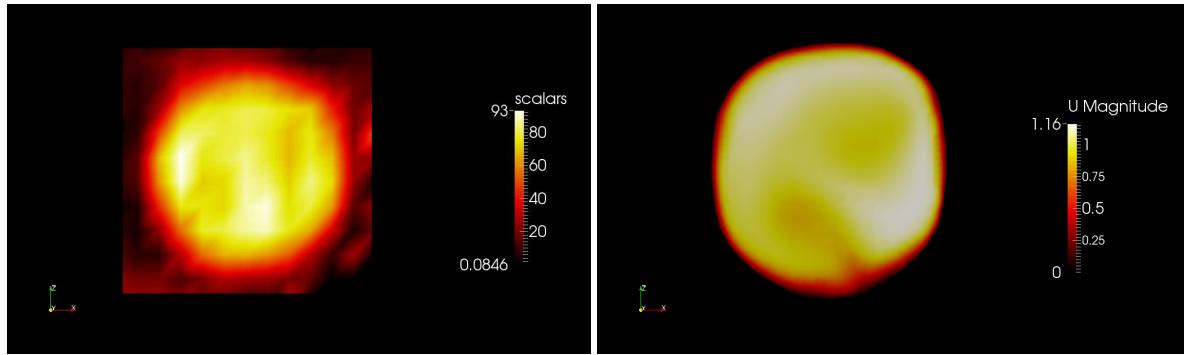


Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.005$

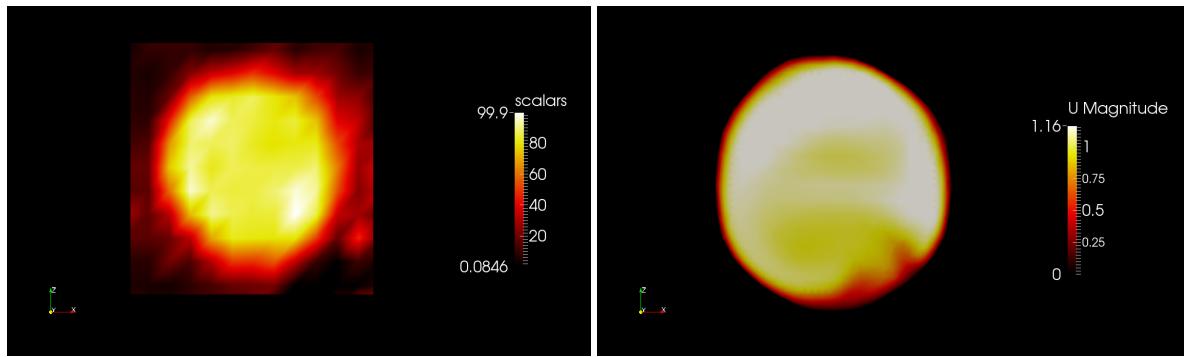


## 5. Analysis of the results

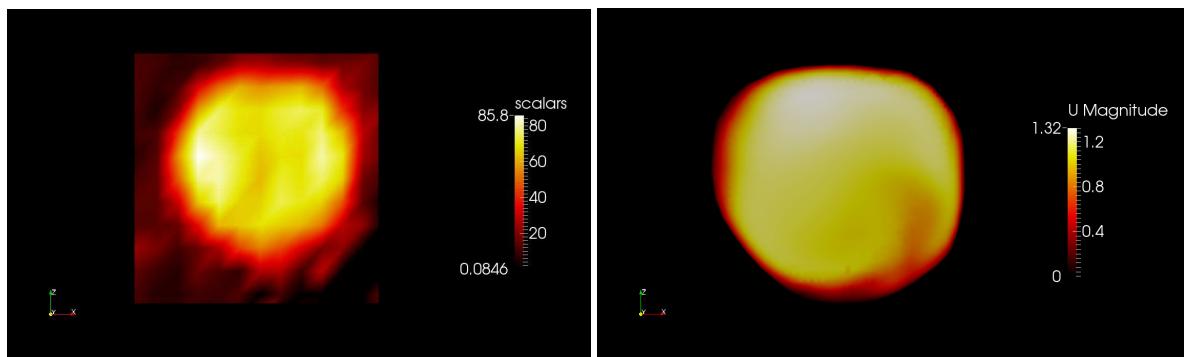
---



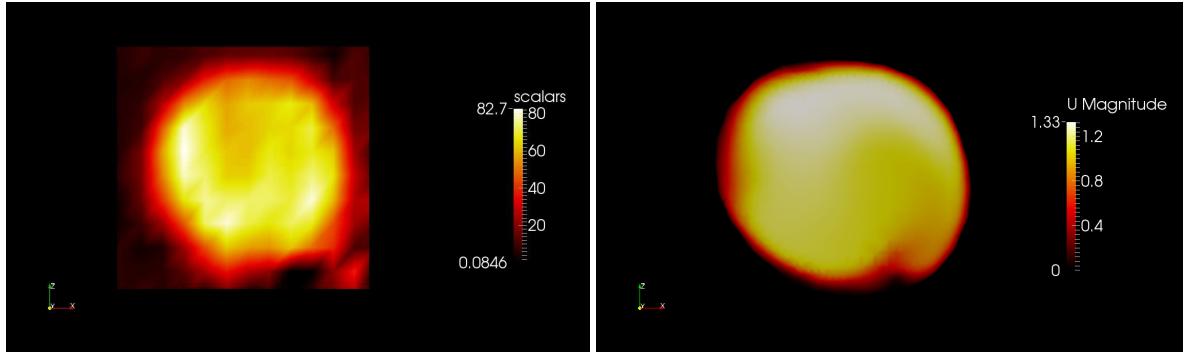
Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.062$



Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.082$



Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.105$



Comparison between the velocity module in slices of the aorta using the 4D method or simulating with OpenFOAM. Slices at  $y = 0.121$

As the images to the left are obtained directly from the 4D method, it implies that some information comes from outside the aorta and it is difficult to discern between the velocity values only related to the aorta flow and external information. When considering the velocity distribution, Figure 5.1 may help in understanding which values have to be used for the comparison:

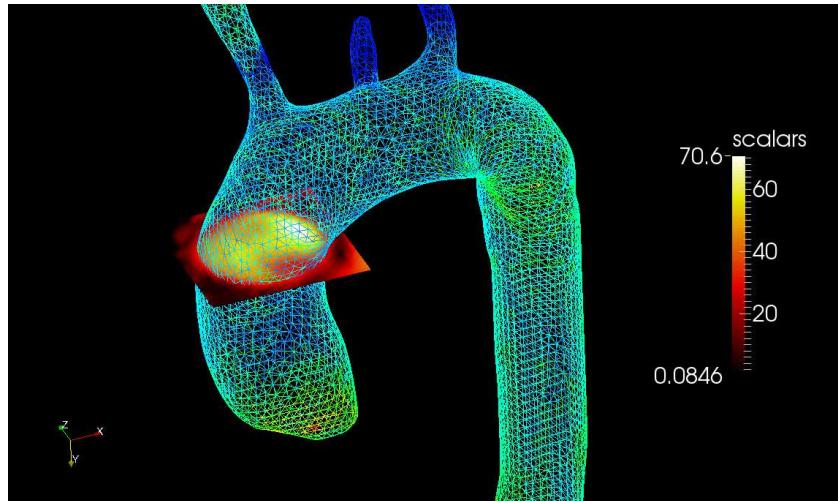


Figure 5.2: Relation between the slices shown in Section 5.2 and the aorta geometry

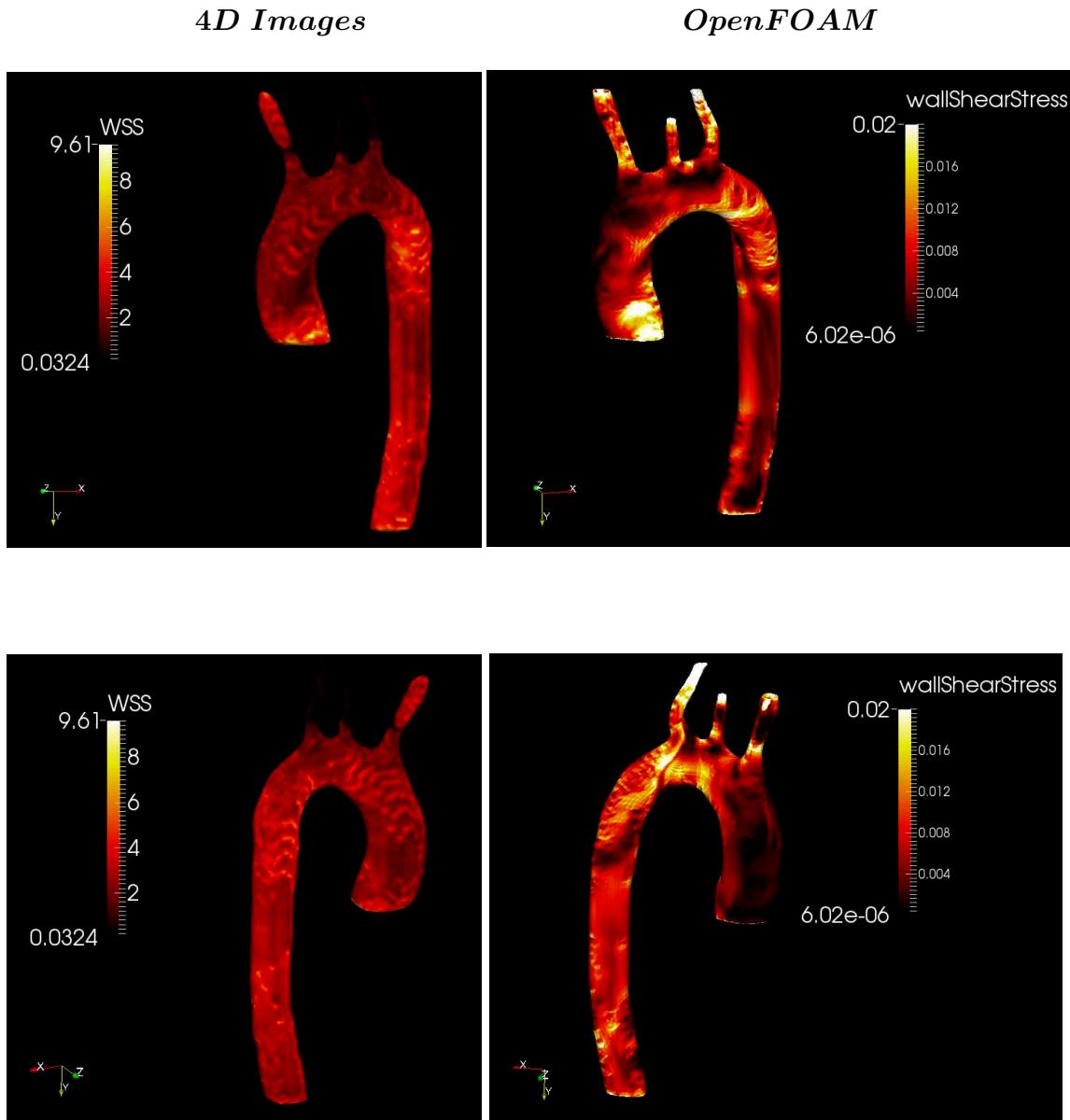
### 5.3 Analysis of the wall shear stress

#### 5.3.1 Comparison of results according to OpenFOAM and the 4D method

The wall shear stress distribution obtained using both methods:

## 5. Analysis of the results

---



Comparison between wall shear stress distribution according to the results of the 4D method or the OpenFOAM simulation

For the comparison, two aspects must be considered:

- The units of the 4D images are expressed in  $Pa$ , whereas the OpenFOAM results are in  $m^2/s^2$
- The range of the wall shear stress values in the 4D images goes from 0.0324 Pa to 9.61 Pa, whereas with the OpenFOAM simulation it goes from  $6.1 \times 10^{-3}$  Pa to 20.8 Pa. The ranges present different maximum values because the average

---

shear stress using the 4D method is lower and it would be then difficult to compare the results. Therefore, these images allow a comparison of the stress distribution more than its numerical values

As it can be seen, there are common patterns, especially reflected in those regions with higher and lower values of wall shear stress. On average, the stress is lower in the 4D method, whose values present a lower definition due to the fact that the spatial discretization is not as accurate as with the OpenFOAM mesh. As the wall shear stress is proportional to the derivative of the velocity profile at the walls of the aorta, a lower spatial discretization on the walls would imply less accuracy.

For a better analysis, the frequency distribution of wall shear stress in each case is presented. The frequency distribution shows how often are specific ranges of wall shear stress repeated with respect to the whole distribution. It allows the computation of an average value, the understanding of which values of wall shear stress are more abundant and to observe the dispersion of the results. Due to the fact that the OpenFOAM mesh contains more points (it presents a higher refinement), there are more values on the frequency distribution in Figure 5.4 than in Figure 5.5, as each value of shear stress has been calculated in a wall point of the mesh:

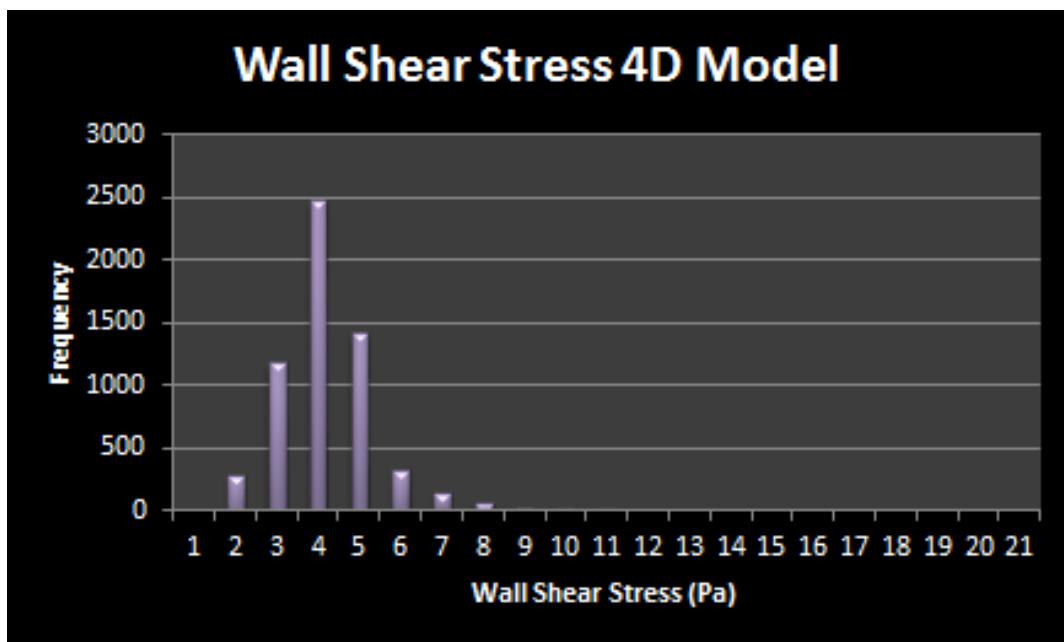


Figure 5.3: Frequency distribution of the wall shear stress obtained with the 4D method

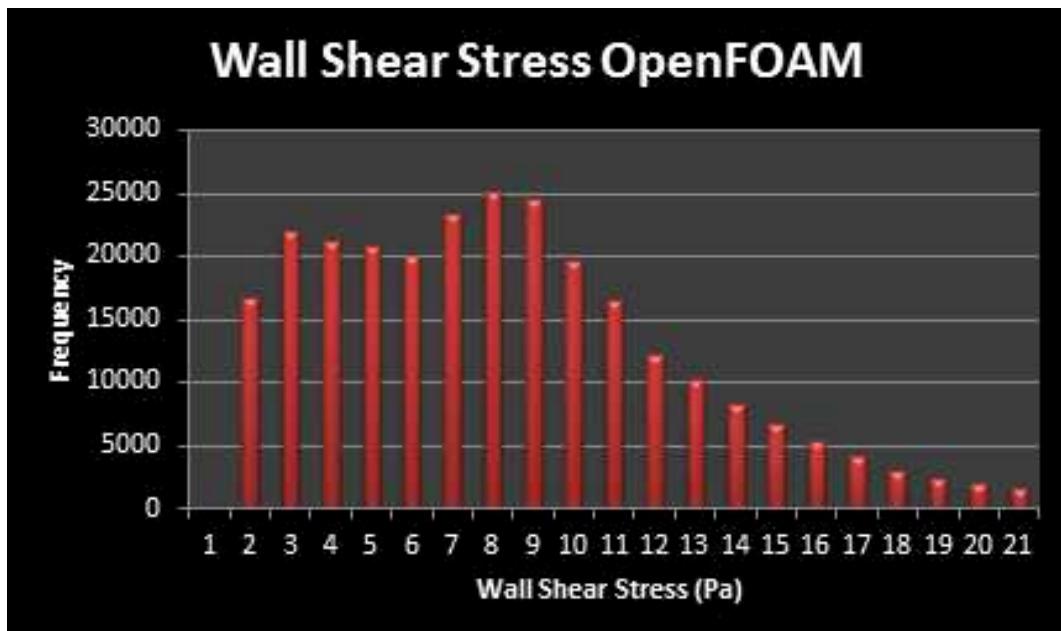


Figure 5.4: Frequency distribution of the wall shear stress obtained with OpenFOAM

Some relevant conclusions that can be extracted from the previous results are:

- The mean value and the standard deviation in both cases are:

Table 5.1: Means and standard deviations of the wall shear stress distributions

**4D Model    OpenFOAM Model**

|                           |       |       |
|---------------------------|-------|-------|
| <b>Mean</b>               | 3.18  | 7.25  |
| <b>Standard deviation</b> | 1.135 | 4.294 |

- Both the mean and the standard deviation are significantly lower in Figure 5.5
- It may be hypothesized that the standard deviation is lower in Figure 5.5 due to the fact that the spatial discretization (especially at the walls of the aorta) is not as accurate as with Figure 5.4. Consequently, with the OpenFOAM simulation it is possible to differentiate more accurately between the great variety of stress values that are obtained
- The difference in the standard deviations reflects that the two methods present different resolutions. However, the difference between the mean values indi-

---

cates that either one method tends to underestimate or the other tends to overestimate the values of the wall shear stress

### **5.3.2 Analysis of wall shear stress results depending on the spatial discretization used for the simulation**

Finally, the current section presents a comparison between the wall shear stress distribution depending on what kind of mesh is used for the OpenFOAM simulation, with the aim of:

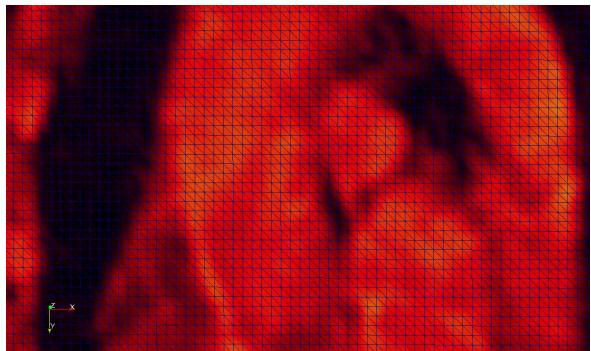
- Justificate the mesh used for the OpenFOAM simulations, taking into consideration the convergence time, the number of cells of the mesh and the results obtained
- Simulate the case with OpenFOAM using a mesh similar to the one used in the 4D method. In this case, the spatial discretization is similar, but the calculation methods are different. It may help in extracting suitable conclusions
- Compare the wall shear stress frequency distributions using different kinds of spatial discretization

Accordingly, four different cases are shown:

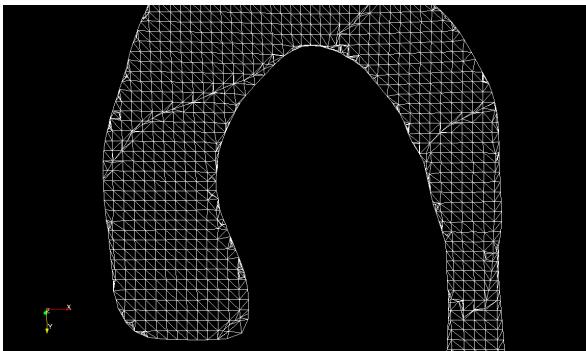
|                          |   |
|--------------------------|---|
| <b>4 models of study</b> | <p><b>4D Model:</b> case presenting a uniform spatial discretization, solved using the 4D method. Voxels with dimensions of <math>1.78 \times 1.78 \times 2</math> mm</p> <p><b>Mesh Model 1:</b> same spatial discretization as 4D Model but solved with OpenFOAM</p> <p><b>Mesh Model 2:</b> uniform spatial discretization but more refined than the previous models: uniform cell level equal to 4 with respect to the initial mesh. Solved with OpenFOAM</p> <p><b>Mesh Model 3:</b> is the main case of this work, presented in Chapter 3. Mesh refined at the walls up to level 6 and with two layers of cells to obtain a high refinement. Solved with OpenFOAM</p> |
|--------------------------|---|

The different kinds of spatial discretization are depicted in the following figures:

**4D Model**

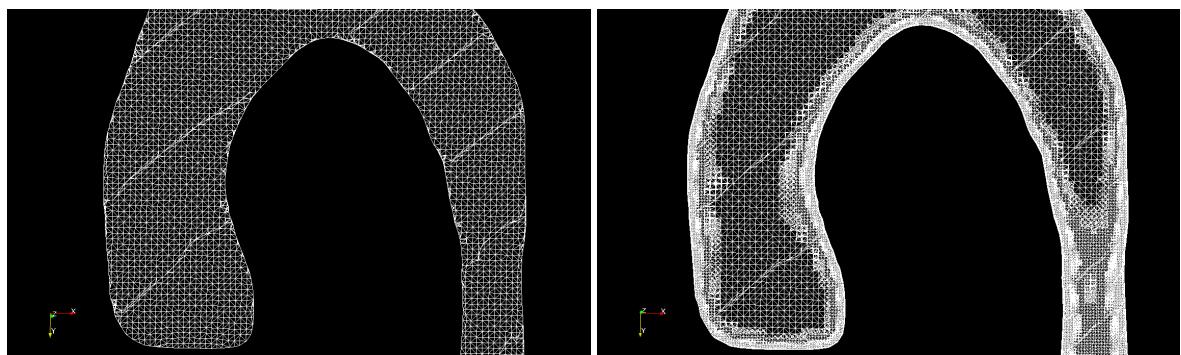


**Mesh Model 1**

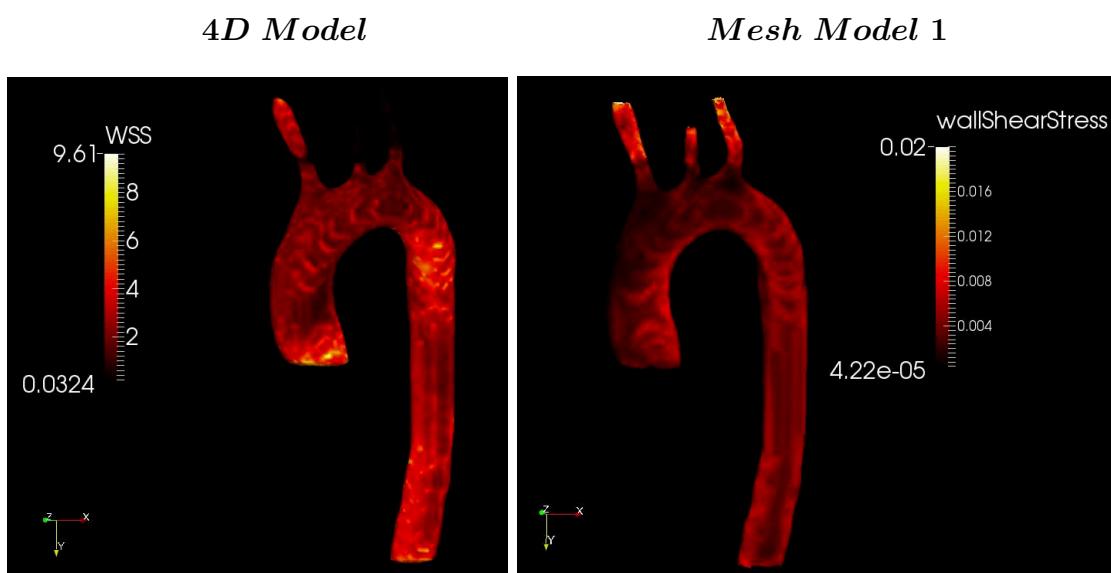


**Mesh Model 2**

**Mesh Model 3**

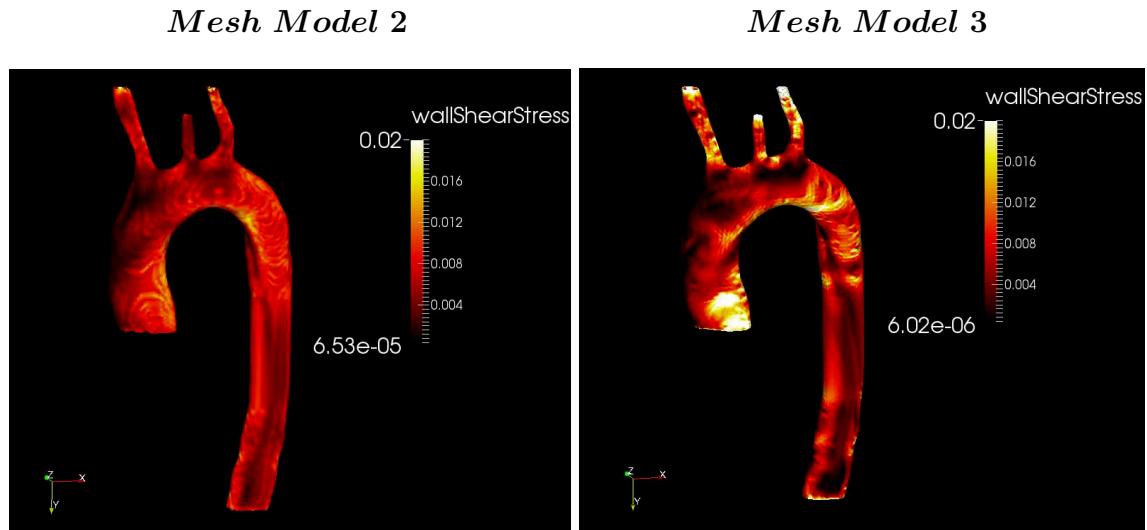


The wall shear stress distribution obtained with each case:



## 5. Analysis of the results

---



By using these simulations, the wall shear stress frequency distributions are:

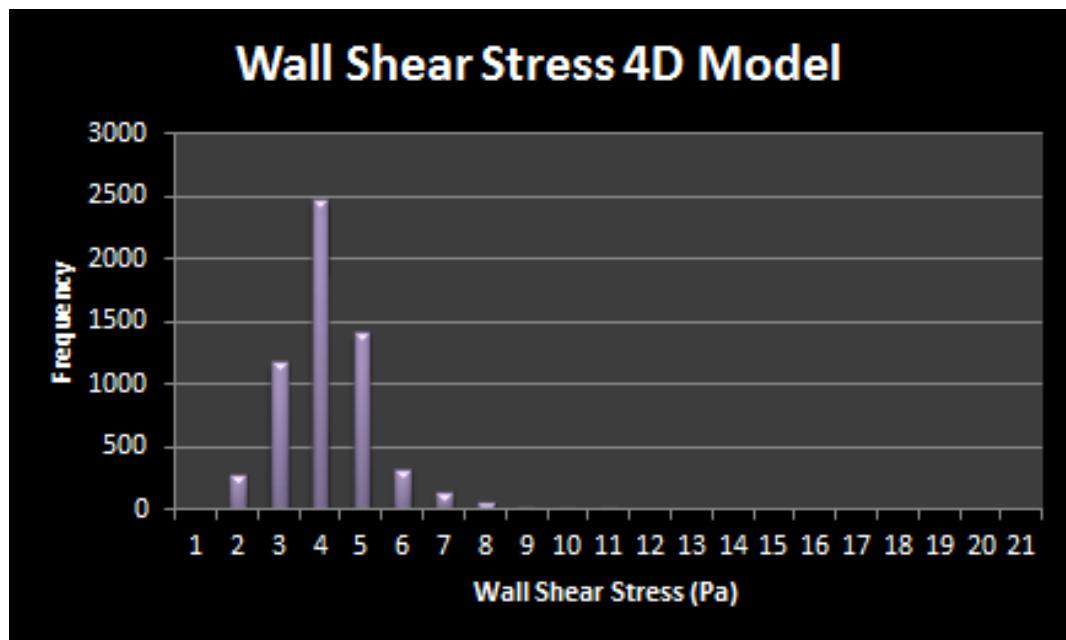


Figure 5.5: Frequency distribution of the wall shear stress obtained with 4D Model

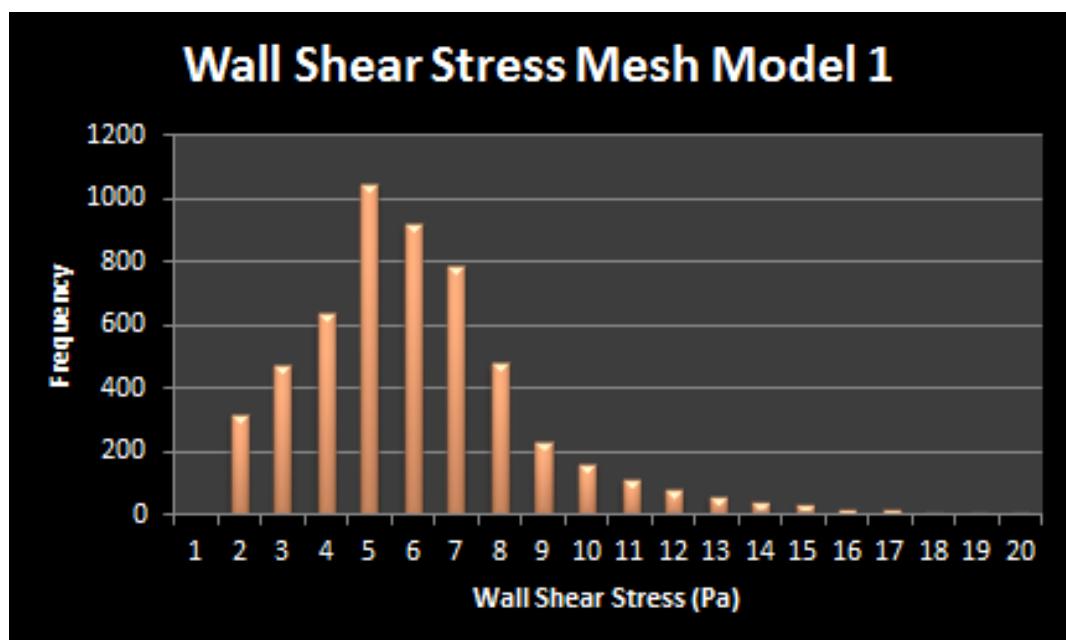


Figure 5.6: Frequency distribution of the wall shear stress obtained with Mesh Model 1

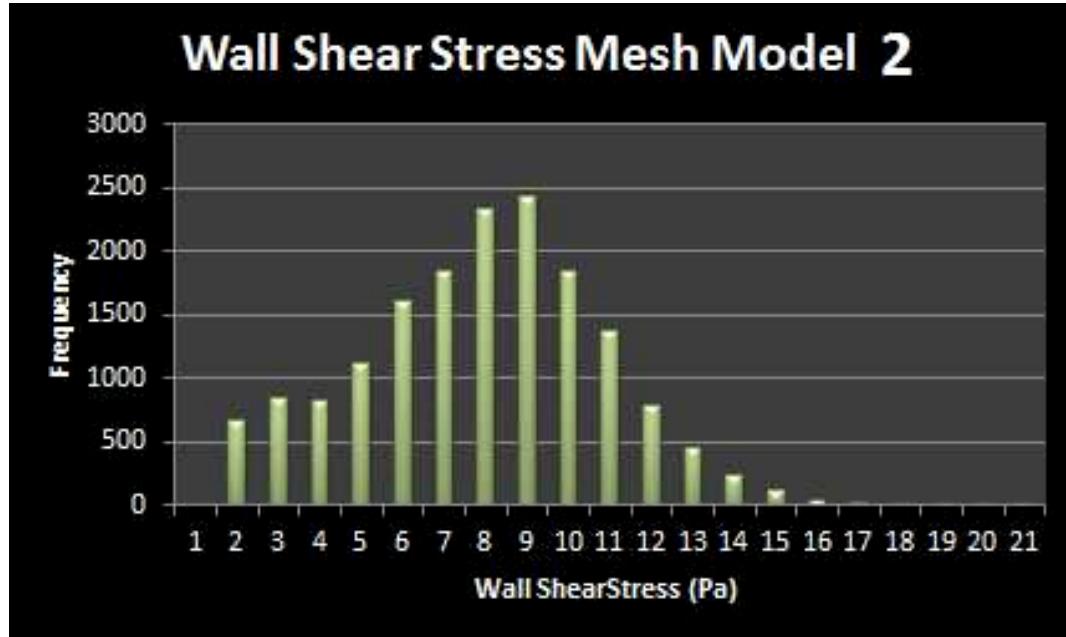


Figure 5.7: Frequency distribution of the wall shear stress obtained with Mesh Model 2

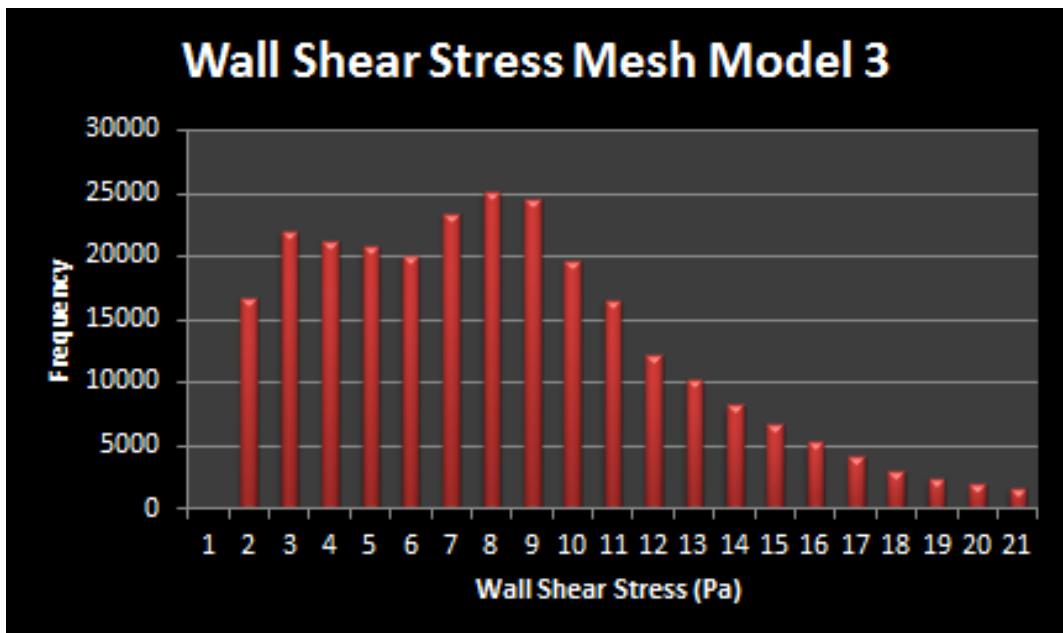


Figure 5.8: Frequency distribution of the wall shear stress obtained with Mesh Model 3

Again, the mean values and the standard deviations have been computed. Also the number of cells of each mesh, as well as the convergence time needed in the simulations are included:

Table 5.2: Means, standard deviations, number of cells and convergence time of each model

|                           | 4D Model | Model 1 | Model 2 | Model 3 |
|---------------------------|----------|---------|---------|---------|
| <b>Mean</b>               | 3.18     | 5.13    | 6.94    | 7.25    |
| <b>Standard deviation</b> | 1.135    | 2.744   | 2.965   | 4.294   |
| <b>Number of cells</b>    | -        | 12870   | 81766   | 1517676 |
| <b>Convergence time</b>   | -        | 64 s    | 101 s   | 3534 s  |

From these figures and tables, and according to the results presented in the whole Chapter 5, it is possible to conclude that:

- The higher the mesh refinement (globally and especially at the aorta walls), the higher the mean values and the standard deviations

- 
- High mesh refinements may contribute through two different ways. The first is related to the resolution of the internal flow. The internal mesh has to adequately solve the whole flow structure, which affects the main flow characteristics and therefore the velocity gradient at the walls. The second is the resolution of the boundary layer, directly involved in the computation of the wall shear stress. It is possible to observe this influence by comparing the results obtained with Mesh Model 3 and the other cases
  - In the supra-aortic branches, there are high flow accelerations and thus high values of wall shear stress. These high values are influenced by the simulation itself, rather than having a relevant physical meaning. Hence, such values have been discarded from the distribution as they are not contained within the range of 0-20 Pa. As a consequence, erroneous flow resolutions at the supra-aortic vessels do not influence the histograms presented in this section
  - Between 4D Model and Mesh Model 1, the shape of the shear stress frequency distribution is similar, although the mean values and the standard deviations differ
  - Between Mesh Model 1 and Mesh Model 2, the main differences come from the fact that the mean value of the frequency distribution shifts to the left and the values are more scattered
  - Between Mesh Model 2 and Mesh Model 3, the same pattern of the previous point is obtained again, with the peculiarity that left frequency values increased significantly. It has been found that this increase is due to the fact that with the mesh of Model 3 very low flow speeds can be computed, coming from the inlet vortices, the flow detachment, etc. The previous models do not adequately solve the flow in these areas and therefore low speed values could not be obtained
  - There is more difference in mean values between Models 1 and 2 than between 2 and 3. However, the opposite happens with the standard deviations. This may be due to the fact that the main distinction between Models 1 and 2 is the internal cell density (difference in internal flow resolution), whereas the main differences between Models 2 and 3 are the mesh refinements at the walls (difference in the wall shear stress computation).

## 6. Conclusions

- According to the initial hypothesis, hemodynamics of blood through a thoracic aorta has been solved
- It has been possible to model important and relevant flow patterns, as for instance flow detachment at the aortic arch or generation of vortices at the widening after the inlet
- By smoothing the surface of the aorta using OpenFOAM utilities, it has been possible to smoothen surface imperfections and avoid unphysical stress concentrators
- Different kinds of boundary conditions at the supra-aortic vessels have been tested. When imposing outlet velocities, the simulations did not succeed and when imposing equal pressure values the simulations did not provide physically realistic results. The chosen solution was to impose outlet flow rates, according to experimental data obtained from MRI measures. With these boundary conditions and with a uniform inlet flow rate, the simulations converged and offered suitable results
- Without an OpenFOAM GUI, it became difficult to define new patches on the initial aorta geometry to apply the boundary conditions. In this case, the `topoSet` and `createPatch` utilities were used, plus a personal bash code developed to manage different Linux instructions
- High values of wall refinement have been obtained in the aorta mesh (the distance between the aorta wall and the first nodes are of the order of micrometers). This guarantees that the boundary layer is correctly solved and wall shear stress values are accurately computed
- The OpenFOAM simulation carried out in this work has been compared with similar cases studied with commercial CFD packages. In the majority of cases, the results agreed

- 
- The wall shear stress values computed with the OpenFOAM simulation are maintained within an adequate range and their average value is close to the expected one
  - In contrast with other CFD aorta resolutions, and due to the high mesh refinement which has been obtained, the distribution of wall shear stress computed with the current OpenFOAM simulation is more realistic: the different regions according to each phenomenon are clearly separated and therefore it is possible to analyze medical implications more accurately
  - The numerical solution of those cases of the guide with a well-known analytical solution perfectly matched the results. Very low errors with no significantly large or heavy simulations were obtained. This is a demonstration of the enormous power of the CFD tools and their importance in the future of fluid mechanics
  - By comparing the results of the OpenFOAM simulation with the images of the 4D method, it can be concluded that the main flow patterns are similar, although the average value of shear stress is lower when using the second method
  - As the mesh used for the 4D method is coarser, its wall shear distribution is not as detailed as the one obtained with OpenFOAM. However, with the 4D method, real velocity values are used. There is a compromise between an OpenFOAM simulation with mathematically computed values but with an accurate mesh, and a 4D method, managing velocity values directly measured from real aortas but with low mesh resolution
  - When comparing the wall shear stress frequency distributions, the standard deviations of the results of the 4D method were found similar to the ones obtained with an OpenFOAM simulation with the same spatial discretization. However, the mean values increased from 3.18 Pa to 5.13 Pa
  - The higher the number of cells of the mesh of the OpenFOAM simulations, the higher the mean values of the wall shear stress distribution, as well as the standard deviations
  - When comparing the results of the 4D method with the main OpenFOAM simulation of this work, the difference in standard deviation of the wall shear stress distributions may be indicative of the accuracy of the results. The more refined the mesh, the higher the variety of stress values that can be computed.

Therefore the dispersion of the results increases. The difference in mean values might come from three different aspects: the internal flow resolution, the definition and treatment of the aorta walls during its segmentation and the computation of the wall shear stress according to the values obtained with MRI

### 6.1 Further steps

At the end of this work, interesting points considered by the author to continue the analyses and simulations developed during its realization are discussed.

1. To contrast the numerical results collected in the guide with experimental work in a laboratory, including for instance wind tunnel experiments for the external flow cases
2. To thoroughly focus on turbulence. Only RAS turbulence models have been used, so it would be interesting to study and simulate new cases by using LES or DNS
3. To simulate with OpenFOAM unhealthy aorta geometries (mainly with an aneurysm or with an inlet flow with excentricity)
4. To introduce more accurate boundary conditions and physical properties such as an inlet flow adapted to the inlet valve or suitable turbulence models
5. To simulate a whole cardiac cycle
6. To determine the wall shear stress by using experimental techniques and compare it with the OpenFOAM simulations and the 4D method

---

# A. Appendix

## A.1 surfaceFeatureExtractDict

```
1 lastAortaSmooth10It.stl
2 {
3     // How to obtain raw features (extractFromFile || extractFromSurface)
4     extractionMethod      extractFromSurface;
5
6     extractFromSurfaceCoeffs
7     {
8         // Mark edges whose adjacent surface normals are at an angle less
9         // than includedAngle as features
10        // - 0 : selects no edges
11        // - 180: selects all edges
12        includedAngle    150;
13    }
14
15    subsetFeatures
16    {
17        // Keep nonManifold edges (edges with >2 connected faces)
18        nonManifoldEdges      no;
19
20        // Keep open edges (edges with 1 connected face)
21        openEdges            yes;
22    }
23
24    // Write options
25
26        // Write features to obj format for postprocessing
27        writeObj                yes;
28    }
29
30    // ****//
```

## A.2 blockMeshDict

```
1 /*----- C++ -----*/
2 | =====
3 | \\ / F i e l d          | OpenFOAM: The Open Source CFD Toolbox
4 | \\ / O p e r a t i o n   | Version: 2.2.1
5 | \\ / A n d              | Web:      www.OpenFOAM.org
```

## A. Appendix

---

```
6 |     \\\/      M anipulation   |
7 |*-----*/|
8 FoamFile
9 {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     object        blockMeshDict;
14 }
15 // * * * * *
16 convertToMeters 0.1;
17
18 vertices
19 (
20     (-1 -1 -0.1)
21     (2 -1 -0.1)
22     (2 2 -0.1)
23     (-1 2 -0.1)
24     (-1 -1 3)
25     (2 -1 3)
26     (2 2 3)
27     (-1 2 3)
28 );
29
30 blocks
31 (
32     hex (0 1 2 3 4 5 6 7) (20 20 20) simpleGrading (1 1 1)
33 );
34
35 edges
36 (
37 );
38
39 boundary
40 (
41     frontAndBack
42     {
43         type slip;
44         faces
45         (
46             (3 7 6 2)
47             (1 5 4 0)
48         );
49     }
50     leftWall
51     {
52         type slip;
53         faces
54         (
55             (0 4 7 3)
56         );
57     }
58     rightWall
59     {
60         type slip;
```

---

```

63         faces
64         (
65             (2 6 5 1)
66         );
67     }
68     topWall
69     {
70         type slip;
71         faces
72         (
73             (0 3 2 1)
74         );
75     }
76     bottomWall
77     {
78         type slip;
79         faces
80         (
81             (4 5 6 7)
82         );
83     }
84 );
85
86 // ****

```

### A.3 boundary

```

1  /*-------------------------------------------------------------------- C++ -*-*/
2  | =====
3  | \\ / Field           | OpenFOAM: The Open Source CFD Toolbox
4  | \\ / Operation       | Version: 2.3.0
5  | \\ / And             | Web: www.OpenFOAM.org
6  | \\\ Manipulation    |
7  /*-----------------------------------------------------------------*/
8 FoamFile
9 {
10     version      2.0;
11     format        binary;
12     class         polyBoundaryMesh;
13     location      "constant/polyMesh";
14     object        boundary;
15 }
16 // * * * * *
17
18 5
19 (
20     frontAndBack
21     {
22         type          slip;
23         inGroups     1(slip);
24         nFaces       800;
25         startFace    22800;
26         inGroups     1 ( slip );

```

---

## A. Appendix

```

27      faces          ( ( 3 7 6 2 ) ( 1 5 4 0 ) );
28  }
29  leftWall
30  {
31      type          slip ;
32      inGroups     1(slip) ;
33      nFaces        400;
34      startFace    23600;
35      inGroups     1 ( slip );
36      faces         ( ( 0 4 7 3 ) );
37  }
38  rightWall
39  {
40      type          slip ;
41      inGroups     1(slip) ;
42      nFaces        400;
43      startFace    24000;
44      inGroups     1 ( slip );
45      faces         ( ( 2 6 5 1 ) );
46  }
47  topWall
48  {
49      type          slip ;
50      inGroups     1(slip) ;
51      nFaces        400;
52      startFace    24400;
53      inGroups     1 ( slip );
54      faces         ( ( 0 3 2 1 ) );
55  }
56  bottomWall
57  {
58      type          slip ;
59      inGroups     1(slip) ;
60      nFaces        400;
61      startFace    24800;
62      inGroups     1 ( slip );
63      faces         ( ( 4 5 6 7 ) );
64  }
65  )
66
67 // ****

```

## A.4 snappyHexMeshDict

```
1  /*----- C++ -----*/  
2  |=====|  
3  | \ \ / F ield | OpenFOAM: The Open Source CFD Toolbox  
4  | \ \ / O peration | Version: 2.2.0  
5  | \ \ / A nd | Web: www.OpenFOAM.org  
6  | \ \ / M anipulation |  
7  \*----- */  
8  FoamFile  
9  {
```



## A. Appendix

---

```
66 // Note that this is the number of cells before removing the part which
67 // is not 'visible' from the keepPoint. The final number of cells might
68 // actually be a lot less.
69 maxGlobalCells 20000000;
70
71 // The surface refinement loop might spend lots of iterations refining just a
72 // few cells. This setting will cause refinement to stop if <= minimumRefine
73 // are selected for refinement. Note: it will at least do one iteration
74 // (unless the number of cells to refine is 0)
75 minRefinementCells 0;
76
77 // Allow a certain level of imbalance during refining
78 // (since balancing is quite expensive)
79 // Expressed as fraction of perfect balance (= overall number of cells /
80 // nProcs). 0=balance always.
81 maxLoadUnbalance 0.1;
82
83 // Number of buffer layers between different levels.
84 // 1 means normal 2:1 refinement restriction , larger means slower
85 // refinement.
86 nCellsBetweenLevels 3;
87
88
89 // Explicit feature edge refinement
90 // ~~~~~
91
92 // Specifies a level for any cell intersected by explicitly provided
93 // edges.
94 // This is a featureEdgeMesh , read from constant/triSurface for now.
95 // Specify 'levels' in the same way as the 'distance' mode in the
96 // refinementRegions (see below). The old specification
97 //      level 2;
98 // is equivalent to
99 //      levels ((0 2));
100
101 features
102 (
103     {
104         file "lastAortaSmooth10It.eMesh";
105         levels ((4 4));
106     }
107 );
108
109
110 // Surface based refinement
111 // ~~~~~
112
113 // Specifies two levels for every surface. The first is the minimum level ,
114 // every cell intersecting a surface gets refined up to the minimum level .
115 // The second level is the maximum level. Cells that 'see' multiple
116 // intersections where the intersections make an
117 // angle > resolveFeatureAngle get refined up to the maximum level .
118
119 refinementSurfaces
120 {
121
122     lastAortaSmooth10It.stl //STL filename where all the regions are added
```

---

```

123      {
124      level (6 6);
125      regions
126      {
127          /*zone0 //Named region in the STL file
128          {
129              // Surface-wise min and max refinement level
130              level (3 3);
131                  // Optional specification of patch type (default is wall). No
132                  // constraint types (cyclic, symmetry) etc. are allowed.
133                  patchInfo
134                  {
135                      type patch;
136                      inGroups (meshedPatches);
137                  }
138          }/*
139      }
140  }
141  }
142
143 // Feature angle:
144 // - used if min and max refinement level of a surface differ
145 // - used if feature snapping (see snapControls below) is used
146 resolveFeatureAngle 30;
147
148
149 // Region-wise refinement
150 // ~~~~~
151
152 // Specifies refinement level for cells in relation to a surface. One of
153 // three modes
154 // - distance. 'levels' specifies per distance to the surface the
155 // wanted refinement level. The distances need to be specified in
156 // increasing order.
157 // - inside. 'levels' is only one entry and only the level is used. All
158 // cells inside the surface get refined up to the level. The surface
159 // needs to be closed for this to be possible.
160 // - outside. Same but cells outside.
161
162 refinementRegions
163 {
164     refinementCylinder
165     {
166         mode inside;
167         levels ((1E15 7));
168     }
169 }
170
171 // Mesh selection
172 // ~~~~~
173
174 // After refinement patches get added for all refinementSurfaces and
175 // all cells intersecting the surfaces get put into these patches. The
176 // section reachable from the locationInMesh is kept.
177 // NOTE: This point should never be on a face, always inside a cell, even
178 // after refinement.
179 locationInMesh (0.05 0.03 0.175);

```

---

## A. Appendix

---

```
180
181     // Whether any faceZones (as specified in the refinementSurfaces)
182     // are only on the boundary of corresponding cellZones or also allow
183     // free-standing zone faces. Not used if there are no faceZones.
184     allowFreeStandingZoneFaces true;
185 }
186
187 // Settings for the snapping.
188 snapControls
189 {
190     // Number of patch smoothing iterations before finding correspondence
191     // to surface
192     nSmoothPatch 3;
193
194     // Maximum relative distance for points to be attracted by surface.
195     // True distance is this factor times local maximum edge length.
196     // Note: changed(corrected) w.r.t 17x! (17x used 2* tolerance)
197     tolerance 1.0;
198
199     // Number of mesh displacement relaxation iterations.
200     nSolveIter 30;
201
202     // Maximum number of snapping relaxation iterations. Should stop
203     // before upon reaching a correct mesh.
204     nRelaxIter 5;
205
206     // Feature snapping
207
208         // Number of feature edge snapping iterations.
209         // Leave out altogether to disable.
210         nFeatureSnapIter 10;
211
212         // Detect (geometric only) features by sampling the surface
213         // (default=false).
214         implicitFeatureSnap false;
215
216         // Use castellatedMeshControls::features (default = true)
217         explicitFeatureSnap true;
218
219         // Detect features between multiple surfaces
220         // (only for explicitFeatureSnap , default = false)
221         multiRegionFeatureSnap false;
222 }
223
224 // Settings for the layer addition.
225 addLayersControls
226 {
227     // Are the thickness parameters below relative to the undistorted
228     // size of the refined cell outside layer (true) or absolute sizes (false).
229     relativeSizes true;
230
231     // Layer thickness specification. This can be specified in one of four ways
232     // - expansionRatio and finalLayerThickness (cell nearest internal mesh)
233     // - expansionRatio and firstLayerThickness (cell on surface)
234     // - overall thickness and firstLayerThickness
235     // - overall thickness and finalLayerThickness
236
```

---

```

237     // Expansion factor for layer mesh
238     expansionRatio 2;
239
240     // Wanted thickness of the layer furthest away from the wall.
241     // If relativeSizes this is relative to undistorted size of cell
242     // outside layer.
243     finalLayerThickness 0.4;
244
245     // Wanted thickness of the layer next to the wall.
246     // If relativeSizes this is relative to undistorted size of cell
247     // outside layer.
248     //firstLayerThickness 0.3;
249
250     // Wanted overall thickness of layers.
251     // If relativeSizes this is relative to undistorted size of cell
252     // outside layer.
253     //thickness 0.5
254
255
256     // Minimum overall thickness of total layers. If for any reason layer
257     // cannot be above minThickness do not add layer.
258     // If relativeSizes this is relative to undistorted size of cell
259     // outside layer..
260     minThickness 0.1;
261
262
263     // Per final patch (so not geometry!) the layer information
264     // Note: This behaviour changed after 21x. Any non-mentioned patches
265     //       now slide unless:
266     //           – nSurfaceLayers is explicitly mentioned to be 0.
267     //           – angle to nearest surface < slipFeatureAngle (see below)
268     layers
269     {
270         aortaWall
271         {
272             nSurfaceLayers 2;
273
274         }
275         maxY
276         {
277             nSurfaceLayers 2;
278             // Per patch layer data
279             expansionRatio      2;
280             finalLayerThickness 0.4;
281             minThickness        0.1;
282         }
283
284         // Disable any mesh shrinking and layer addition on any point of
285         // a patch by setting nSurfaceLayers to 0
286         frozenPatches
287         {
288             nSurfaceLayers 0;
289         }
290     }
291
292     // If points get not extruded do nGrow layers of connected faces that are
293     // also not grown. This helps convergence of the layer addition process

```

---

## A. Appendix

---

```
294     // close to features.  
295     // Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)  
296     nGrow 0;  
297  
298     // Advanced settings  
299  
300     // When not to extrude surface. 0 is flat surface, 90 is when two faces  
301     // are perpendicular  
302     featureAngle 60;  
303  
304     // At non-patched sides allow mesh to slip if extrusion direction makes  
305     // angle larger than slipFeatureAngle.  
306     slipFeatureAngle 30;  
307  
308     // Maximum number of snapping relaxation iterations. Should stop  
309     // before upon reaching a correct mesh.  
310     nRelaxIter 5;  
311  
312     // Number of smoothing iterations of surface normals  
313     nSmoothSurfaceNormals 1;  
314  
315     // Number of smoothing iterations of interior mesh movement direction  
316     nSmoothNormals 3;  
317  
318     // Smooth layer thickness over surface patches  
319     nSmoothThickness 10;  
320  
321     // Stop layer growth on highly warped cells  
322     maxFaceThicknessRatio 0.5;  
323  
324     // Reduce layer growth where ratio thickness to medial  
325     // distance is large  
326     maxThicknessToMedialRatio 0.3;  
327  
328     // Angle used to pick up medial axis points  
329     // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.  
330     minMedianAxisAngle 90;  
331  
332     // Create buffer region for new layer terminations  
333     nBufferCellsNoExtrude 0;  
334  
335     // Overall max number of layer addition iterations. The mesher will exit  
336     // if it reaches this number of iterations; possibly with an illegal  
337     // mesh.  
338     nLayerIter 50;  
339  
340     // Max number of iterations after which relaxed meshQuality controls  
341     // get used. Up to nRelaxIter it uses the settings in meshQualityControls,  
342     // after nRelaxIter it uses the values in meshQualityControls::relaxed.  
343     nRelaxedIter 20;  
344  
345     // Additional reporting: if there are just a few faces where there  
346     // are mesh errors (after adding the layers) print their face centres.  
347     // This helps in tracking down problematic mesh areas.  
348     //additionalReporting true;  
349 }  
350
```

---

```

351 // Generic mesh quality settings. At any undoable phase these determine
352 // where to undo.
353 meshQualityControls
354 {
355     // Maximum non-orthogonality allowed. Set to 180 to disable.
356     maxNonOrtho 45;
357
358     // Max skewness allowed. Set to <0 to disable.
359     maxBoundarySkewness 20;
360     maxInternalSkewness 4;
361
362     // Max concaveness allowed. Is angle (in degrees) below which concavity
363     // is allowed. 0 is straight face, <0 would be convex face.
364     // Set to 180 to disable.
365     maxConcave 80;
366
367     // Minimum pyramid volume. Is absolute volume of cell pyramid.
368     // Set to a sensible fraction of the smallest cell volume expected.
369     // Set to very negative number (e.g. -1E30) to disable.
370     minVol 1e-13;
371
372     // Minimum quality of the tet formed by the face-centre
373     // and variable base point minimum decomposition triangles and
374     // the cell centre. This has to be a positive number for tracking
375     // to work. Set to very negative number (e.g. -1E30) to
376     // disable.
377     //    <0 = inside out tet ,
378     //    0 = flat tet
379     //    1 = regular tet
380     minTetQuality 1e-9;
381
382     // Minimum face area. Set to <0 to disable.
383     minArea -1;
384
385     // Minimum face twist. Set to <-1 to disable. dot product of face normal
386     // and face centre triangles normal
387     minTwist 0.05;
388
389     // minimum normalised cell determinant
390     // 1 = hex, <= 0 = folded or flattened illegal cell
391     minDeterminant 0.001;
392
393     // minFaceWeight (0 -> 0.5)
394     minFaceWeight 0.05;
395
396     // minVolRatio (0 -> 1)
397     minVolRatio 0.01;
398
399     // must be >0 for Fluent compatibility
400     minTriangleTwist -1;
401
402     //-- if >0 : preserve single cells with all points on the surface if the
403     // resulting volume after snapping (by approximation) is larger than
404     // minVolCollapseRatio times old volume (i.e. not collapsed to flat cell).
405     // If <0 : delete always.
406     //minVolCollapseRatio 0.5;
407

```

---

## A. Appendix

---

```
408      // Advanced
409
410      // Number of error distribution iterations
411      nSmoothScale 4;
412      // amount to scale back displacement at error points
413      errorReduction 0.75;
414
415      // Optional : some meshing phases allow usage of relaxed rules.
416      // See e.g. addLayersControls::nRelaxedIter.
417      relaxed
418      {
419          //-- Maximum non-orthogonality allowed. Set to 180 to disable.
420          maxNonOrtho 45;
421      }
422  }
423
424      // Advanced
425
426      // Flags for optional output
427      // 0 : only write final meshes
428      // 1 : write intermediate meshes
429      // 2 : write volScalarField with cellLevel for postprocessing
430      // 4 : write current intersections as .obj files
431      debug 0;
432
433      // Merge tolerance. Is fraction of overall bounding box of initial mesh.
434      // Note: the write tolerance needs to be higher than this.
435      mergeTolerance 1e-6;
436
437      // **** //
```

## A.5 checkMesh

```
1  /*-----*/
2  | ====== |
3  | \ \ / F ield           | OpenFOAM: The Open Source CFD Toolbox
4  | \ \ / O peration        | Version: 2.3.0
5  | \ \ / A nd              | Web:     www.OpenFOAM.org
6  | \ \ / M anipulation    |
7  /*-----*/
8  Build  : 2.3.0-f5222ca19ce6
9  Exec   : checkMesh
10 Date   : May 23 2014
11 Time   : 17:48:31
12 Host   : "H25109"
13 PID    : 7022
14 Case   : /home/jordi/Escriptori/AortaFoam
15 nProcs : 1
16 sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
17 fileModificationChecking : Monitoring run-time modified files using timeStampMaster
18 allowSystemOperations : Disallowing user-supplied system call operations
19
20 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

---

```

21 Create time
22
23 Create polyMesh for time = 1
24
25 Time = 1
26
27 Mesh stats
28     points:          1493135
29     faces:           3749654
30     internal faces: 3348304
31     cells:           1128520
32     faces per cell: 6.28962
33     boundary patches: 6
34     point zones:    0
35     face zones:     0
36     cell zones:     0
37
38 Overall number of cells of each type:
39     hexahedra:      1035405
40     prisms:          0
41     wedges:          0
42     pyramids:        0
43     tet wedges:      0
44     tetrahedra:      0
45     polyhedra:       93115
46 Breakdown of polyhedra by number of faces:
47     faces   number of cells
48         6      28221
49         9      32886
50         12     20459
51         15     11054
52         18      495
53
54 Checking topology ...
55     Boundary definition OK.
56     Cell to face addressing OK.
57     Point usage OK.
58     Upper triangular ordering OK.
59     Face vertices OK.
60     Number of regions: 1 (OK).
61
62 Checking patch topology for multiply connected surfaces ...
63             Patch   Faces   Points   Surface topology
64             frontAndBack   0       0       ok (empty)
65             leftWall     0       0       ok (empty)
66             rightWall    0       0       ok (empty)
67             topWall      0       0       ok (empty)
68             bottomWall   0       0       ok (empty)
69             aortaWall    401350  401352  ok (closed singly connected)
70
71 Checking geometry ...
72     Overall domain bounding box (0.0389844 -0.0442188 0.149359) (0.11375 0.131328
73     0.183023)
74     Mesh (non-empty, non-wedge) directions (1 1 1)
75     Mesh (non-empty) directions (1 1 1)
76     Boundary openness (4.90475e-16 -1.88181e-17 -2.36364e-16) OK.
77     Max cell openness = 3.05442e-16 OK.

```

---

## A. Appendix

---

```
77     Max aspect ratio = 1.03333 OK.
78     Minimum face area = 5.49316e-08. Maximum face area = 9.08203e-07. Face area
79     magnitudes OK.
80     Min volume = 1.33038e-11. Max volume = 8.5144e-10. Total volume =
81     6.96518e-05. Cell volumes OK.
82     Mesh non-orthogonality Max: 25.6096 average: 9.01125
83     Non-orthogonality check OK.
84     Face pyramids OK.
85     Max skewness = 0.333367 OK.
86     Coupled point location match (average 0) OK.
87
88     Mesh OK.
89
90     Time = 2
91
92     Mesh stats
93         points: 1377941
94         faces: 3596786
95         internal faces: 3348304
96         cells: 1128520
97         faces per cell: 6.15416
98         boundary patches: 6
99         point zones: 0
100        face zones: 0
101        cell zones: 0
102
103        Overall number of cells of each type:
104            hexahedra: 921527
105            prisms: 52632
106            wedges: 0
107            pyramids: 0
108            tet wedges: 463
109            tetrahedra: 1
110            polyhedra: 153897
111            Breakdown of polyhedra by number of faces:
112                faces    number of cells
113                    4      38526
114                    5      22256
115                    6      28221
116                    9      32886
117                   12      20459
118                   15      11054
119                   18       495
120
121        Checking topology...
122            Boundary definition OK.
123            Cell to face addressing OK.
124            Point usage OK.
125            Upper triangular ordering OK.
126            Face vertices OK.
127            Number of regions: 1 (OK).
128
129            Checking patch topology for multiply connected surfaces ...
130                Patch    Faces    Points           Surface topology
131                    frontAndBack      0      0           ok (empty)
132                    leftWall        0      0           ok (empty)
133                    rightWall       0      0           ok (empty)
```

---

```

132           topWall      0      0          ok (empty)
133           bottomWall   0      0          ok (empty)
134           aortaWall   248482  286158      ok (closed singly connected)
135
136     Checking geometry...
137       Overall domain bounding box (0.0389332 -0.0443906 0.149293) (0.113807 0.131309
138           0.183016)
139       Mesh (non-empty, non-wedge) directions (1 1 1)
140       Mesh (non-empty) directions (1 1 1)
141       Boundary openness (-1.43363e-16 1.58962e-17 2.76816e-16) OK.
142       Max cell openness = 3.59876e-16 OK.
143       Max aspect ratio = 5.54205 OK.
144       Minimum face area = 3.08657e-09. Maximum face area = 9.68777e-07. Face area
145           magnitudes OK.
146       Min volume = 1.5113e-12. Max volume = 9.13292e-10. Total volume =
147           6.96441e-05. Cell volumes OK.
148       Mesh non-orthogonality Max: 44.994 average: 10.5987
149       Non-orthogonality check OK.
150       Face pyramids OK.
151       Max skewness = 2.92068 OK.
152       Coupled point location match (average 0) OK.
153
154     Mesh OK.
155
156     —> FOAM Warning :
157       From function polyMesh::readUpdateState polyMesh::readUpdate()
158       in file meshes/polyMesh/polyMeshIO.C at line 207
159       Number of patches has changed. This may have unexpected consequences.
160       Proceed with care.
161
162     Time = 3
163
164     Mesh stats
165       points: 1882949
166       faces: 4987227
167       internal faces: 4739360
168       cells: 1571329
169       faces per cell: 6.19004
170       boundary patches: 6
171       point zones: 0
172       face zones: 0
173       cell zones: 0
174
175     Overall number of cells of each type:
176       hexahedra: 1261492
177       prisms: 52712
178       wedges: 1
179       pyramids: 0
180       tet wedges: 550
181       tetrahedra: 2
182       polyhedra: 256572
183
184     Breakdown of polyhedra by number of faces:
185       faces   number of cells
186           4      39626
187           5      22129
188           6      28566
189           7      76408
190           8      24811

```

---

## A. Appendix

---

```
185          9    32867
186         10    123
187         12   20459
188         15   11056
189         18    495
190         21    29
191         24     3
192
193 Checking topology ...
194     Boundary definition OK.
195     Cell to face addressing OK.
196     Point usage OK.
197     Upper triangular ordering OK.
198     Face vertices OK.
199     Number of regions: 1 (OK).
200
201 Checking patch topology for multiply connected surfaces ...
202             Patch      Faces      Points      Surface topology
203             frontAndBack      0          0      ok (empty)
204             leftWall        0          0      ok (empty)
205             rightWall       0          0      ok (empty)
206             lowerWall       0          0      ok (empty)
207             upperWall       0          0      ok (empty)
208             aortaWall     247867    285740      ok (closed singly connected)
209
210 Checking geometry ...
211     Overall domain bounding box (0.0389332 -0.0443906 0.149293) (0.113807 0.131309
212           0.183016)
213     Mesh (non-empty, non-wedge) directions (1 1 1)
214     Mesh (non-empty) directions (1 1 1)
215     Boundary openness (-1.91619e-16 -3.42491e-17 1.49812e-16) OK.
216     Max cell openness = 4.03527e-16 OK.
217     Max aspect ratio = 13.4184 OK.
218     Minimum face area = 2.63761e-09. Maximum face area = 9.76263e-07. Face area
219           magnitudes OK.
220     Min volume = 4.52487e-13. Max volume = 9.14821e-10. Total volume =
221           6.96434e-05. Cell volumes OK.
222     Mesh non-orthogonality Max: 66.7028 average: 9.79509
223     Non-orthogonality check OK.
224     Face pyramids OK.
225     Max skewness = 1.94323 OK.
226     Coupled point location match (average 0) OK.
227     Mesh OK.
228
229 End
```

## A.6 topoSetDict

```
1 /*----- C++ -----*/
2 | =====
3 | \\ / Field | OpenFOAM: The Open Source CFD Toolbox
4 | \\ / Operation | Version: 2.2.1
```

```

5 | \\ / A nd | Web: www.OpenFOAM.org
6 | \\\ Manipulation |
7 *-
8 FoamFile
9 {
10     version      2.0;
11     format       ascii;
12     class        dictionary;
13     object       topoSetDict;
14 }
15
16 // * * * * *
17
18 actions
19 (
20     {
21         name      allPatchSet;
22         type      faceSet;
23         action    new;
24         source   boundaryToFace;
25         sourceInfo
26         {
27         }
28     }
29
30     {
31         name      c1;           // Inlet Set
32         type      faceSet;
33         action    new;
34         source   boxToFace;
35         sourceInfo
36         {
37             box (0.0425 0.0558 0.153) (0.0675 0.0708 0.177);
38         }
39     }
40
41     {
42         name      boxOutletSet;
43         type      faceSet;
44         action    new;
45         source   boxToFace;
46         sourceInfo
47         {
48             box (0.0825 0.129 0.16) (0.108 0.144 0.184);
49         }
50     }
51
52     {
53         name      s1;           // S01 Set
54         type      faceSet;
55         action    new;
56         source   boxToFace;
57         sourceInfo
58         {
59             box (0.04 -0.048 0.172) (0.05 -0.038 0.182);
60         }
61     }

```

## A. Appendix

---

```
62
63     {
64         name      boxS02Set ;
65         type      faceSet ;
66         action    new ;
67         source    boxToFace ;
68         sourceInfo
69         {
70             box ( 0.06 -0.0405 0.15 ) ( 0.07 -0.0305 0.16 ) ;
71         }
72     }
73
74     {
75         name      boxS03Set ;
76         type      faceSet ;
77         action    new ;
78         source    boxToFace ;
79         sourceInfo
80         {
81             box ( 0.071 -0.0515 0.15 ) ( 0.081 -0.0415 0.16 ) ;
82         }
83     }
84
85 );
86
87 // ****//
```

## A.7 createPatchDict

```
1  /*----- C++ -----*/
2  | _____ |
3  | \\ / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \\ / O peration   | Version: 2.2.1 |
5  | \\ / A nd        | Web:      www.OpenFOAM.org |
6  | \\/ M anipulation | |
7  \*-----*/
8
9  FoamFile
10 {
11     version    2.0;
12     format     ascii;
13     class      dictionary;
14     object     createPatchDict;
15 }
16 // * * * * *
17
18 // This application/dictionary controls:
19 // - optional: create new patches from boundary faces (either given as
20 //   a set of patches or as a faceSet)
21 // - always: order faces on coupled patches such that they are opposite. This
22 //   is done for all coupled faces, not just for any patches created.
23 // - optional: synchronise points on coupled patches.
24
```

---

```

25 // 1. Create cyclic:
26 // - specify where the faces should come from
27 // - specify the type of cyclic. If a rotational specify the rotationAxis
28 // and centre to make matching easier
29 // - always create both halves in one invocation with correct 'neighbourPatch'
30 // setting.
31 // - optionally pointSync true to guarantee points to line up.
32
33 // 2. Correct incorrect cyclic:
34 // This will usually fail upon loading:
35 // "face 0 area does not match neighbour 2 by 0.0100005%"
36 // -- possible face ordering problem."
37 // - in polyMesh/boundary file:
38 //      - loosen matchTolerance of all cyclics to get case to load
39 //      - or change patch type from 'cyclic' to 'patch'
40 //          and regenerate cyclic as above
41
42
43 // Do a synchronisation of coupled points after creation of any patches.
44 // Note: this does not work with points that are on multiple coupled patches
45 //       with transformations (i.e. cyclics).
46 pointSync false;
47
48 // Patches to create.
49 patches
50 (
51     {
52         // Name of new patch
53         name inlet;
54
55         // Type of new patch
56         patchInfo
57         {
58             type patch;
59         }
60
61         // How to construct: either from 'patches' or 'set'
62         constructFrom set;
63
64         // If constructFrom = patches : names of patches. Wildcards allowed.
65         patches (aortaWall);
66
67         // If constructFrom = set : name of faceSet
68         set definitiveInletSet;
69     }
70
71     {
72         // Name of new patch
73         name outlet;
74
75         // Type of new patch
76         patchInfo
77         {
78             type patch;
79         }
80
81         // How to construct: either from 'patches' or 'set'

```

---

## A. Appendix

---

```
82     constructFrom set ;
83
84     // If constructFrom = patches : names of patches. Wildcards allowed .
85     patches (aortaWall) ;
86
87     // If constructFrom = set : name of faceSet
88     set definitiveOutletSet ;
89 }
90
91 {
92     // Name of new patch
93     name S01 ;
94
95     // Type of new patch
96     patchInfo
97     {
98         type patch ;
99     }
100
101    // How to construct: either from 'patches' or 'set'
102    constructFrom set ;
103
104    // If constructFrom = patches : names of patches. Wildcards allowed .
105    patches (aortaWall) ;
106
107    // If constructFrom = set : name of faceSet
108    set definitiveS01Set ;
109 }
110
111 {
112     // Name of new patch
113     name S02 ;
114
115     // Type of new patch
116     patchInfo
117     {
118         type patch ;
119     }
120
121     // How to construct: either from 'patches' or 'set'
122     constructFrom set ;
123
124     // If constructFrom = patches : names of patches. Wildcards allowed .
125     patches (aortaWall) ;
126
127     // If constructFrom = set : name of faceSet
128     set definitiveS02Set ;
129 }
130
131 {
132     // Name of new patch
133     name S03 ;
134
135     // Type of new patch
136     patchInfo
137     {
138         type patch ;
```

---

```

139         }
140
141         // How to construct: either from 'patches' or 'set'
142         constructFrom set;
143
144         // If constructFrom = patches : names of patches. Wildcards allowed.
145         patches (aortaWall);
146
147         // If constructFrom = set : name of faceSet
148         set definitiveS03Set;
149     }
150
151 );
152
153 // ****

```

## A.8 Script for patch definition

```

1 #!/bin/bash
2
3 #Requirements:
4
5 # -The 2 directories (createPatchScript and createPatchTemplates) must be at
6 #   Desktop containing templates of the OpenFOAM files according to the
7 #   instructions shown in the script
8 # -The AortaFoam case directory must be at Desktop
9 # -In the templates sheets there must be a space of 3 lines between the heading
#   and the first "("
10 # -The last directory created by snappyHexMesh must be named "3" (if not, change
#   it manually)
11 cp AortaFoam/3/polyMesh/sets/c1 createPatchScript/c1Intermediate #Copy data from
#   topoSet
12
13 sed -i 1,+19d createPatchScript/c1Intermediate #Erase header
14
15 sed -i '$d' createPatchScript/c1Intermediate
16
17 sed -i '$d' createPatchScript/c1Intermediate
18
19 sed -i '$d' createPatchScript/c1Intermediate
20
21
22 cp AortaFoam/3/polyMesh/sets/allPatchSet createPatchScript/allPatchSetIntermediate
#   Copy data from topoSet
23
24 sed -i 1,+19d createPatchScript/allPatchSetIntermediate #Erase header
25
26 sed -i '$d' createPatchScript/allPatchSetIntermediate
27
28 sed -i '$d' createPatchScript/allPatchSetIntermediate
29

```

## A. Appendix

---

```
30    sed -i '$d' createPatchScript/allPatchSetIntermediate
31
32    sort createPatchScript/c1Intermediate > createPatchScript/c1Intermediate.sort
33        #Sort (necessary for the comm instruction)
34    sort createPatchScript/allPatchSetIntermediate >
35        createPatchScript/allPatchSetIntermediate.sort
36
37    comm -12 createPatchScript/c1Intermediate.sort
38        createPatchScript/allPatchSetIntermediate.sort >
39        createPatchScript/definitiveInletSetIntermediate #Compare to find common faces
40        shearing the condition of external face and being inside the box
41
42    wc -w < createPatchScript/definitiveInletSetIntermediate >
43        createPatchScript/WordCounter #Count the number of faces of the resulting file
44
45    ed -s createPatchScript/definitiveInletSet <<< $'20r
46        createPatchScript/definitiveInletSetIntermediate\nw'
47
48    ed -s createPatchScript/definitiveInletSet <<< $'18r
49        createPatchScript/WordCounter\nw'
50
51    > createPatchScript/c1Intermediate #Reinitialize to 0 all the files which were used
52    > createPatchScript/allPatchSetIntermediate
53    > createPatchScript/WordCounter
54    > createPatchScript/c1Intermediate.sort
55    > createPatchScript/allPatchSetIntermediate.sort
56    > createPatchScript/definitiveInletSetIntermediate
57
58    cp createPatchScript/definitiveInletSet
59        AortaFoam/3/polyMesh/sets/definitiveInletSet
60    cp createPatchTemplates/definitiveInletSetTemplate
61        createPatchScript/definitiveInletSet
62
63    #####
64
65    cp AortaFoam/3/polyMesh/sets/boxOutletSet
66        createPatchScript/boxOutletSetIntermediate
67
68    sed -i 1,+19d createPatchScript/boxOutletSetIntermediate
69
70    sed -i '$d' createPatchScript/boxOutletSetIntermediate
71
72    sed -i '$d' createPatchScript/boxOutletSetIntermediate
73
74    sed -i '$d' createPatchScript/boxOutletSetIntermediate
75
```

---

```
76 sort createPatchScript/boxOutletSetIntermediate >
    createPatchScript/boxOutletSetIntermediate.sort
77 sort createPatchScript/allPatchSetIntermediate >
    createPatchScript/allPatchSetIntermediate.sort
78
79 comm -12 createPatchScript/boxOutletSetIntermediate.sort
    createPatchScript/allPatchSetIntermediate.sort >
    createPatchScript/definitiveOutletSetIntermediate
80
81 wc -w < createPatchScript/definitiveOutletSetIntermediate >
    createPatchScript/WordCounter
82
83 ed -s createPatchScript/definitiveOutletSet <<< $'20r
    createPatchScript/definitiveOutletSetIntermediate\nw'
84
85 ed -s createPatchScript/definitiveOutletSet <<< $'18r
    createPatchScript/WordCounter\nw'
86
87 > createPatchScript/boxOutletSetIntermediate
88 > createPatchScript/allPatchSetIntermediate
89 > createPatchScript/WordCounter
90 > createPatchScript/boxOutletSetIntermediate.sort
91 > createPatchScript/allPatchSetIntermediate.sort
92 > createPatchScript/definitiveOutletSetIntermediate
93
94 cp createPatchScript/definitiveOutletSet
    AortaFoam/3/polyMesh/sets/definitiveOutletSet
95 cp createPatchTemplates/definitiveOutletSetTemplate
    createPatchScript/definitiveOutletSet
96
97 #####
98
99 cp AortaFoam/3/polyMesh/sets/s1 createPatchScript/s1Intermediate
100
101 sed -i 1,+19d createPatchScript/s1Intermediate
102
103 sed -i '$d' createPatchScript/s1Intermediate
104
105 sed -i '$d' createPatchScript/s1Intermediate
106
107 sed -i '$d' createPatchScript/s1Intermediate
108
109
110 cp AortaFoam/3/polyMesh/sets/allPatchSet createPatchScript/allPatchSetIntermediate
111
112 sed -i 1,+19d createPatchScript/allPatchSetIntermediate
113
114 sed -i '$d' createPatchScript/allPatchSetIntermediate
115
116 sed -i '$d' createPatchScript/allPatchSetIntermediate
117
118 sed -i '$d' createPatchScript/allPatchSetIntermediate
119
120 sort createPatchScript/s1Intermediate > createPatchScript/s1Intermediate.sort
121 sort createPatchScript/allPatchSetIntermediate >
    createPatchScript/allPatchSetIntermediate.sort
122
```

## A. Appendix

---

```
123 comm -12 createPatchScript/s1Intermediate.sort
      createPatchScript/allPatchSetIntermediate.sort >
      createPatchScript/definitiveS01SetIntermediate
124
125 wc -w < createPatchScript/definitiveS01SetIntermediate >
      createPatchScript/WordCounter
126
127 ed -s createPatchScript/definitiveS01Set <<< $'20r
      createPatchScript/definitiveS01SetIntermediate\nw'
128
129 ed -s createPatchScript/definitiveS01Set <<< $'18r
      createPatchScript/WordCounter\nw'
130
131 > createPatchScript/s1Intermediate
132 > createPatchScript/allPatchSetIntermediate
133 > createPatchScript/WordCounter
134 > createPatchScript/s1Intermediate.sort
135 > createPatchScript/allPatchSetIntermediate.sort
136 > createPatchScript/definitiveS01SetIntermediate
137
138 cp createPatchScript/definitiveS01Set AortaFoam/3/polyMesh/sets/definitiveS01Set
139 cp createPatchTemplates/definitiveS01SetTemplate createPatchScript/definitiveS01Set
140
141 #####
142
143 cp AortaFoam/3/polyMesh/sets/boxS02Set createPatchScript/boxS02SetIntermediate
144
145 sed -i 1,+19d createPatchScript/boxS02SetIntermediate
146
147 sed -i '$d' createPatchScript/boxS02SetIntermediate
148
149 sed -i '$d' createPatchScript/boxS02SetIntermediate
150
151 sed -i '$d' createPatchScript/boxS02SetIntermediate
152
153
154 cp AortaFoam/3/polyMesh/sets/allPatchSet createPatchScript/allPatchSetIntermediate
155
156 sed -i 1,+19d createPatchScript/allPatchSetIntermediate
157
158 sed -i '$d' createPatchScript/allPatchSetIntermediate
159
160 sed -i '$d' createPatchScript/allPatchSetIntermediate
161
162 sed -i '$d' createPatchScript/allPatchSetIntermediate
163
164 sort createPatchScript/boxS02SetIntermediate >
      createPatchScript/boxS02SetIntermediate.sort
165 sort createPatchScript/allPatchSetIntermediate >
      createPatchScript/allPatchSetIntermediate.sort
166
167 comm -12 createPatchScript/boxS02SetIntermediate.sort
      createPatchScript/allPatchSetIntermediate.sort >
      createPatchScript/definitiveS02SetIntermediate
168
169 wc -w < createPatchScript/definitiveS02SetIntermediate >
      createPatchScript/WordCounter
```

---

```
170
171 ed -s createPatchScript/definitiveS02Set <<< $'20r
172         createPatchScript/definitiveS02SetIntermediate\nw'
173
174 ed -s createPatchScript/definitiveS02Set <<< $'18r
175         createPatchScript/WordCounter\nw'
176
177 > createPatchScript/boxS02SetIntermediate
178 > createPatchScript/allPatchSetIntermediate
179 > createPatchScript/WordCounter
180 > createPatchScript/boxS02SetIntermediate.sort
181 > createPatchScript/allPatchSetIntermediate.sort
182 > createPatchScript/definitiveS02SetIntermediate
183
184 cp createPatchScript/definitiveS02Set AortaFoam/3/polyMesh/sets/definitiveS02Set
185 cp createPatchTemplates/definitiveS02SetTemplate createPatchScript/definitiveS02Set
186
187 #####
```

```
188
189 cp AortaFoam/3/polyMesh/sets/boxS03Set createPatchScript/boxS03SetIntermediate
190
191 sed -i 1,+19d createPatchScript/boxS03SetIntermediate
192
193 sed -i '$d' createPatchScript/boxS03SetIntermediate
194
195 sed -i '$d' createPatchScript/boxS03SetIntermediate
196
197
198 cp AortaFoam/3/polyMesh/sets/allPatchSet createPatchScript/allPatchSetIntermediate
199
200 sed -i 1,+19d createPatchScript/allPatchSetIntermediate
201
202 sed -i '$d' createPatchScript/allPatchSetIntermediate
203
204 sed -i '$d' createPatchScript/allPatchSetIntermediate
205
206 sed -i '$d' createPatchScript/allPatchSetIntermediate
207
208 sort createPatchScript/boxS03SetIntermediate >
209         createPatchScript/boxS03SetIntermediate.sort
210 sort createPatchScript/allPatchSetIntermediate >
211         createPatchScript/allPatchSetIntermediate.sort
212
213 comm -12 createPatchScript/boxS03SetIntermediate.sort
214         createPatchScript/allPatchSetIntermediate.sort >
215         createPatchScript/definitiveS03SetIntermediate
216
217 wc -w < createPatchScript/definitiveS03SetIntermediate >
218         createPatchScript/WordCounter
219
220 ed -s createPatchScript/definitiveS03Set <<< $'20r
221         createPatchScript/definitiveS03SetIntermediate\nw'
222
223 ed -s createPatchScript/definitiveS03Set <<< $'18r
224         createPatchScript/WordCounter\nw'
```

## A. Appendix

---

```
218 > createPatchScript/boxS03SetIntermediate
219 > createPatchScript/allPatchSetIntermediate
220 > createPatchScript/WordCounter
221 > createPatchScript/boxS03SetIntermediate.sort
222 > createPatchScript/allPatchSetIntermediate.sort
223 > createPatchScript/definitiveS03SetIntermediate
224
225
226 cp createPatchScript/definitiveS03Set AortaFoam/3/polyMesh/sets/definitiveS03Set
227 cp createPatchTemplates/definitiveS03SetTemplate createPatchScript/definitiveS03Set
```

## A.9 p

```
1 /*----- C++ -----*/
2 | =====
3 | \\ / Field | OpenFOAM: The Open Source CFD Toolbox
4 | \\ / Operation | Version: 2.2.1
5 | \\ / And | Web: www.OpenFOAM.org
6 | \\\ Manipulation |
7 */
8 FoamFile
9 {
10     version      2.0;
11     format       ascii;
12     class        volScalarField;
13     object       p;
14 }
15 // * * * * *
16
17 dimensions      [0 2 -2 0 0 0];
18
19 internalField   uniform 0;
20
21 boundaryField
22 {
23
24     inlet
25     {
26         type          zeroGradient;
27     }
28
29     outlet
30     {
31         type          fixedValue;
32         value        uniform 0;
33     }
34
35     S01
36     {
37         type          zeroGradient;
38     }
39
40     S02
```

---

```

41      {
42          type          zeroGradient;
43      }
44
45      S03
46      {
47          type          zeroGradient;
48      }
49
50      aortaWall
51      {
52          type          zeroGradient;
53      }
54
55      frontAndBack
56      {
57          type          slip;
58      }
59
60      leftWall
61      {
62          type          slip;
63      }
64
65      rightWall
66      {
67          type          slip;
68      }
69
70      lowerWall
71      {
72          type          slip;
73      }
74
75      upperWall
76      {
77          type          slip;
78      }
79  }
80
81 // ****

```

## A.10 U

```

1  /*----- C++ -----*/
2  | =====
3  | \\ / F ield      | OpenFOAM: The Open Source CFD Toolbox
4  | \\ / O peration   | Version: 2.2.1
5  | \\ / A nd         | Web: www.OpenFOAM.org
6  | \\\/ M anipulation |
7  */
8  FoamFile
9  {

```

## A. Appendix

```

10    version      2.0;
11    format       ascii;
12    class        volVectorField;
13    location     "0";
14    object       U;
15 }
16 // * * * * *
17
18 dimensions      [0 1 -1 0 0 0];
19
20 internalField   uniform (0 0 0);
21
22 boundaryField
23 {
24     inlet
25     {
26         type          fixedValue;
27         value         uniform (-2.3407e-03 -0.6853 -9.674e-04);
28     }
29
30     outlet
31     {
32         type          inletOutlet;
33         inletValue   uniform (0 0 0);
34         value        uniform (0 0 0);
35     }
36
37     S01
38     {
39         type          flowRateInletVelocity;
40         volumetricFlowRate -2.713e-05; //Negative because outgoing
41         value         uniform (0 0 0);
42     }
43
44     S02
45     {
46         type          flowRateInletVelocity;
47         volumetricFlowRate -1.015e-05; //Negative because outgoing
48         value         uniform (0 0 0);
49     }
50
51     S03
52     {
53         type          flowRateInletVelocity;
54         volumetricFlowRate -1.8e-05; //Negative because outgoing
55         value         uniform (0 0 0);
56     }
57
58     aortaWall
59     {
60         type          fixedValue;
61         value         uniform (0 0 0);
62     }
63
64     frontAndBack
65     {
66         type          slip;

```

---

```

67     }
68
69     leftWall
70     {
71         type          slip ;
72     }
73
74     rightWall
75     {
76         type          slip ;
77     }
78
79     lowerWall
80     {
81         type          slip ;
82     }
83
84     upperWall
85     {
86         type          slip ;
87     }
88 }
89 // ****

```

## A.11 transportProperties

```

1  /*----- C++ -----*/
2  | =====
3  | \\ / Field      | OpenFOAM: The Open Source CFD Toolbox
4  | \\ / Operation  | Version: 2.2.1
5  | \\ / And        | Web: www.OpenFOAM.org
6  | \\\ Manipulation |
7 */
8 FoamFile
9 {
10    version    2.0;
11    format     ascii;
12    class      dictionary;
13    object     transportProperties;
14 }
15 // *
16
17 transportModel Newtonian;
18
19 nu           nu [0 2 -1 0 0 0] 3.365e-06;
20
21 // ****

```

## A. Appendix

---

### A.12 RASProperties

```
1  /*----- C++ -----*/\n2  |\n3  | \\\\ / F ield      | OpenFOAM: The Open Source CFD Toolbox\n4  | \\\\ / O peration   | Version: 2.2.1\n5  | \\\\ / A nd         | Web:      www.OpenFOAM.org\n6  | \\\\ / M anipulation |\n7  */\n8  FoamFile\n9 {\n10    version      2.0;\n11    format       ascii;\n12    class        dictionary;\n13    object        RASProperties;\n14 }\n15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\n16\n17  RASModel          laminar;\n18\n19  turbulence        off;\n20\n21  printCoeffs       off;\n22\n23 // ****\n
```

### A.13 controlDict

```
1  /*----- C++ -----*/\n2  |\n3  | \\\\ / F ield      | OpenFOAM: The Open Source CFD Toolbox\n4  | \\\\ / O peration   | Version: 2.2.1\n5  | \\\\ / A nd         | Web:      www.OpenFOAM.org\n6  | \\\\ / M anipulation |\n7  */\n8  FoamFile\n9 {\n10    version      2.0;\n11    format       ascii;\n12    class        dictionary;\n13    object        controlDict;\n14 }\n15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\n16\n17  application     simpleFoam;\n18\n19  startFrom       latestTime;\n20\n21  startTime        0;\n22\n23  stopAt          endTime;\n24
```

---

```

25 endTime         500;
26
27 deltaT          1;
28
29 writeControl    timeStep;
30
31 writeInterval   10;
32
33 purgeWrite      0;
34
35 writeFormat     binary;
36
37 writePrecision  6;
38
39 writeCompression uncompressed;
40
41 timeFormat      general;
42
43 timePrecision   6;
44
45 runTimeModifiable true;
46
47 // ****

```

## A.14 fvSchemes

```

1  /*----- C++ -----*/
2  | ===== |
3  | \ \ / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O peration   | Version: 2.2.1 |
5  | \ \ / A nd        | Web: www.OpenFOAM.org |
6  | \ \ \ M anipulation | |
7  */
8 FoamFile
9 {
10     version    2.0;
11     format     ascii;
12     class      dictionary;
13     location   "system";
14     object     fvSchemes;
15 }
16 // * * * * *
17
18 ddtSchemes
19 {
20     default    steadyState;
21 }
22
23 gradSchemes
24 {
25     default    Gauss linear;
26     grad(U)    cellLimited Gauss linear 1;
27 }

```

## A. Appendix

---

```
28
29     divSchemes
30     {
31         default      none;
32         div(phi,U)    bounded Gauss linearUpwindV grad(U);
33         div((nuEff*dev(T(grad(U))))) Gauss linear;
34     }
35
36     laplacianSchemes
37     {
38         default      Gauss linear corrected;
39     }
40
41     interpolationSchemes
42     {
43         default      linear;
44     }
45
46     snGradSchemes
47     {
48         default      corrected;
49     }
50
51     fluxRequired
52     {
53         default      no;
54         p          ;
55     }
56
57 // **** //
```

## A.15 fvSolution

```
1  /*----- C++ -----*/
2  | ===== |
3  | \\ / F ield      | OpenFOAM: The Open Source CFD Toolbox
4  | \\ / O peration   | Version: 2.2.1
5  | \\ / A nd        | Web: www.OpenFOAM.org
6  | \\\ M anipulation | |
7  \*----- */
8
9  FoamFile
10 {
11     version    2.0;
12     format     ascii;
13     class      dictionary;
14     location   "constant";
15     object     fvSolution;
16 }
17
18 solvers
19 {
20     p
```

---

```

21      {
22          solver          GAMG;
23          tolerance      1e-7;
24          relTol         0.01;
25          smoother        GaussSeidel;
26          nPreSweeps     0;
27          nPostSweeps    2;
28          cacheAgglomeration on;
29          agglomerator    faceAreaPair;
30          nCellsInCoarsestLevel 10;
31          mergeLevels     1;
32      }
33
34      U
35      {
36          solver          smoothSolver;
37          smoother        GaussSeidel;
38          tolerance      1e-8;
39          relTol         0.1;
40          nSweeps        1;
41      }
42  }
43
44 SIMPLE
45 {
46     nNonOrthogonalCorrectors 2;
47 }
48
49 potentialFlow
50 {
51     nNonOrthogonalCorrectors 10;
52 }
53
54 relaxationFactors
55 {
56     fields
57     {
58         p              0.3;
59     }
60     equations
61     {
62         U             0.7;
63     }
64 }
65
66 cache
67 {
68     grad(U);
69 }
70
71 // **** //
```

---

# Bibliography

- [1] Mohamed Gad-el-Hak. Fluid mechanics from the beginning to the third millennium. *Int. J. Engng Ed.*, 1998.
- [2] Josdo Parra Viol. Estudio e implementaci nuevas funcionalidades de deformaci malla en software de mecca de fluidos computacional. Technical report, ETSEIAT, 2011.
- [3] *OpenFOAM, the open source CFD toolbox User Guide*. OpenFOAM, 2013.
- [4] Jonas Lantz. On aortic blood flow simulations. Technical report, Linkping University, 2012.
- [5] Wikipedia. Ascending aorta. [http://en.wikipedia.org/wiki/Ascending\\_aorta](http://en.wikipedia.org/wiki/Ascending_aorta), 2014.
- [6] David Elad and Shmuel Einav. *Physical and flow properties of blood*. McGraw-Hill, 2004.
- [7] Jones, Noble, Eichmann. What determines blood vessel structure? genetic prespecification vs. hemodynamics. <http://physiologyonline.physiology.org/content/21/6/388>, 2006.
- [8] J Renner. Towards subject specific aortic wall shear stress. Technical report, Linkping University, 2011.
- [9] P R Hoskins and D Hardman. Three-dimensional imaging and computational modelling for estimation of wall stresses in arteries. *The British Journal of Radiology*, 82, 2009.
- [10] Eduardo Soudah, Jorge S. Ronda, Marcelino Rodriguez, Herkel Hervilla, Francesc Carreras and Eugenio O. Using 4d phase-contrast mri to determinate blood flow patterns in the aortic diseases. *2nd International Conference on Mathematical and Computational Biomedical Engineering - CMBE2011*, 2011.

- 
- [11] Wikipedia. Openfoam. [http://openfoamwiki.net/index.php/OpenFOAM#cite\\_note-1](http://openfoamwiki.net/index.php/OpenFOAM#cite_note-1), 2010.
  - [12] Eugene de Villiers. The potential of large eddy simulation for the modeling of wall bounded flows. Technical report, Imperial College of Science, 2006.
  - [13] Aleksandar Karac. Drop impact of fluid-filled polyethylene containers. Technical report, Imperial College of Science, 2003.
  - [14] Jennifer Katy Hutchings. On modelling the mass of arctic sea ice. Technical report, University College London, 2003.
  - [15] David Paul Hill. The computer simulation of dispersed two-phase flows. Technical report, Imperial College of Science, 1998.
  - [16] Ruth D Blum. Numerical simulations of aortic blood flow with a bicuspid aortic valve. Technical report, Emory University, 2010.
  - [17] Umberto Morbiducci. Visualization and quantification of blood flow in the human aorta. from in vivo 4d phase contrast mri to subject-specific computational hemodynamics. Technical report, Politecnico di Torino, 2011.
  - [18] Jonas Lantz, Johan Renner and Matts Karlsson. Wall shear stress in a subject specific human aorta. influence of fluid-structure interaction. *Internal Journal of Applied Mechanics*, (3), 4, 759-778, 2011.
  - [19] Eduardo Soudah, E. Y. K. Ng, T. H. Loong, Maurizio Bordone , Uei Pua and Sriram Narayanan. Cfd modelling of abdominal aortic aneurysm on hemodynamic loads using a realistic geometry with ct. *Journal of Computational and Mathematical Methods in Medicine*, 472564, DOI:10.1155/2013/472564, 2013.
  - [20] Auren F. Stalder, Alex Frydrychowicz, Max F. Russe, Jan G. Korvink, Jrgen Hennig, Kuncheng Li and Michael Markl. Assessment of flow instabilities in the healthy aorta using flow-sensitive mri. *Journal of Magnetic Resonance Imaging*, 2011.
  - [21] Eduardo Soudah, Riccardo Rossi, Sergio Idelsohn, and Eugenio O. A reduced order model based on coupled 1d/3d finite element simulations for an efficient analysis of hemodynamics problems. *Journal of Computational Mechanics*, 54, 1013-1022, 2014.
  - [22] Umberto Morbiducci. Visualization and quantification of blood flow in the human aorta. from in vivo 4d phase contrast mri to subject-specific computational hemodynamics. Technical report, Politecnico di Torino.