POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale e dell'Informazione
Corso di Laurea in Ingegneria Matematica

A Patient-Specific Aortic Valve Model based on Moving Resistive Immersed Surfaces

Supervisor: Prof. Alfio Quarteroni

Assistent Supervisor: Ph.D. Elena Faggiano
External Supervisor: Ph.D. Aymen Laadhari

Candidate: Marco Fedele
matr. 760291

A.A. 2012-2013

Contents

List of Figures

List of Tables

Abstract

The study of hemodynamics in the ascending aorta through computational fluid dynamics simulations is a subject of large interest in medical research for its application in the study of different diseases.
The aim of this work is to enrich the numerical simulation with a moving aortic valve model. Closed and open valve surfaces as well as the lumen aorta are reconstructed directly from medical images using a new specific algorithm, allowing a patient-specific simulation.
The valve surface is inserted in the classic Navier-Stokes equations adding a dissipative term with a resistance that can be interpreted as a penalization parameter enforcing the condition of null velocity on it. Furthermore, we considered also the movement of the valve between its closed and open position using a reduced zero-dimensional model to compute the valvular angle governed by pressure and flow-rate values in the left ventricle and in the aorta.
We describe the valve surface as the zero level of a level set function defined analytically. From a numerical point of view, this strategy avoids the problem of having bi-dimensional finite elements immersed in the three-dimensional domain consistent with the valve surface. Moreover, in this way the mesh does not have to change at each time step with the movement of the valve avoiding remeshing and high-computational costs.
This model can help in a better understanding of the hemodynamics in the ascending aorta, hence it can be used to study aortic and valvular diseases or valvular prosthesis. Furthermore it can be easily extended to the other cardiac valves, helping in the building of a complete heart-integration model.

Sommario

Lo studio dell'emodinamica nell'aorta ascendente attraverso metodi computazionali è un ambito di ricerca di grande interesse clinico per la sua applicazione allo studio di diverse patologie. Nonostante questo interesse, pochi lavori inseriscono all'interno del modello computazionale dell'aorta anche il movimento della valvola. Questo perchè i grandi spostamenti della valvola, i forti gradienti di pressione attraverso i lembi e i costi computazionali legati alla risoluzione di un problema completo che includa l'interazione fluido-struttura della valvola col sangue, rendono questo problema un problema ancora aperto e di grande interesse. Infatti la possibilità di effettuare un'analisi dettagliata della dinamica valvolare accoppiata a quella fluida potrebbe costituire la base per interventi clinici o per la pianificazione di nuove protesi.
Vista l'importanza di questo ambito, lo scopo di questa tesi è quello di arricchire le simulazioni numeriche con un modello di valvola aortica che comprenda il movimento. Come prima cosa in questo lavoro proponiamo un metodo che permetta di ricostruire le superfici della valvola aortica rappresentanti la sua configurazione chiusa e quella aperta e la geometria dell'aorta direttamente dalle immagini mediche, permettendoci di lavorare su geometrie realistiche.
Per quanto riguarda il modello numerico che proponiamo, la valvola è inserita nelle classiche equazioni di Naver-Stokes tramite l'aggiunta di un termine dissipativo caratterizzato da una resistenza che può essere interpretata come un termine di penalizzazione che forza la condizione di velocità del flusso a essere nulla sulla valvola. Inoltre, per inserire il movimento della valvola nella nostra formulazione, interpoliamo le superfici corrispondenti alla posizione chiusa e aperta utilizzando un modello zero-dimensionale che fornisce l'angolo valvolare date le pressioni e le portate nel ventricolo e nell'aorta.
La principale novità del nostro approccio è legata a come rappresentiamo le superfici della valvola: infatti abbiamo deciso di rappresentarle in forma

analitica come livello zero di una funzione level set. Da un punto di vista numerico questa strategia evita il problema di avere una rappresentazione della valvola tramite elementi finiti bidimensionali inseriti in un dominio tridimensionale. Inoltre, in questo modo, si evita la difficoltà di dover muovere la mesh rappresentante la valvola evitando così i problemi di remeshing e di costi computazionali legati ad approcci di questo tipo.
Questo modello può essere di aiuto nella comprensione dell'emodinamica in aorta ascendente ed essere quindi utilizzato per studiare le patologie aortiche e le diverse tecinche di impianto valvolare. Inoltre, può essere facilmente esteso ad altre valvole cardiache, verso la costruzione di un modello completo di cuore.

Introduction

The study of hemodynamics in the ascending aorta through computational fluid dynamics simulations is a subject of large interest in medical research for its application in the study of different diseases. Despite this wide interest in the majority of the works that can be found in literature the valve itself is not modelled and in particular valve movement is often neglected. Heart valve models can be divided into two main groups: lumped parameter and multidimensional models. The former have the disadvantage of being characterized by simplified flow assumptions while the latter remain an extremely challenging task. For these reasons it is not yet established a feasible and accurate method to include valve motion inside aorta fluid-dynamic. Despite these difficulties there is a growing interest in this field due to the clinical need in better understanding valve pathologies and related aorta diseases. In fact detailed numerical analysis of the problem may constitute a theoretical basis for clinical interventions, with the ultimate goal to plan surgical strategies for patients who are susceptible to valve diseases, before they create serious aneurysm formation in the ascending aorta. In addition, the surgery techniques draw more and more advanced aortic valve prosthesis, emphasizing the necessity of fully understanding of the hemodynamics in proximity of the valve.

The aim of this work is to enrich the numerical simulation with a moving aortic valve model: in particular we want to propose a new method which is less expensive and less challenging in terms of meshing problems than a 3D fluid-structure model but more accurate than a standard lumped parameter model. Moreover our aim is also to deal with patient-specific studies and for this reason we develop a full framework which starts from medical images and finishes with the numerical simulations. In details, closed and open valve surfaces as well as the lumen aorta are reconstructed directly from medical images using a new specific algorithm.

The valve surface is inserted in the classic Navier-Stokes equations adding a dissipative term with a resistance that can be interpreted as a penalization parameter enforcing the condition of null velocity on it. Furthermore, to introduce the movement of the valve between its closed and open position we use a reduced zero-dimensional model to compute the valvular angle governed by pressure and flow-rate values in the left ventricle and in the aorta. The novelty of our approach is linked to valve representation. In fact, we decide to describe the valve surface as the zero level of a level set function defined analytically. From a numerical point of view, this strategy avoids the problem of having bi-dimensional finite elements immersed in the three-dimensional domain consistent with the valve surface. Moreover, in this way the mesh does not have to change at each time step with the movement of the valve avoiding remeshing and high-computational costs.
This model can help in a better understanding of the hemodynamics in the ascending aorta, hence it can be used to study aortic and valvular diseases or valvular prosthesis. Another application is related to heart simulations: in fact this model is ready to be coupled with a ventricular model in order to obtain more realistic results of the fluid-dynamic inside the heart and the aorta. Furthermore it can be easily extended to the other cardiac valves, helping in the building of a complete heart-integration model.

The thesis is divided in five chapters:

- In the first chapter we describe the anatomy of the aorta and of the aortic valve and we briefly summarize the literature about valve modelling.
- In the second chapter we illustrate the steps and the procedure necessary to create a patient-specific 3D model of the aorta from clinical images, explaining also the algorithm implemented for this purpose.
- In the third chapter we show the mathematical model adopted to include the leaflets of the valve with their movement in the fluid-dynamics equations.
- In the fourth chapter we describe the three dimensional mesh genera-

tion and the patient-specific valve reconstruction.

- In the fifth chapter the numerical implementation and the numerical results are shown.

Chapter 1

Anatomical Description and Aim

This chapter is dedicated to the description of the anatomical environment we want to study: in particular we describe the aorta, the cardiac valves in general and the aortic valve in particular. A brief description of the common valve pathologies is also presented. Since in this thesis we propose a model for the simulation of moving aortic valve, we recall the state of the art of this topic to end with a description of the aim of the thesis.

1.1 Anatomical Description

Aorta. The aorta is the largest artery of human body, with a mean diameter of $3cm$ and a length of $30 - 40cm$; it originates from the left ventricle of the heart and terminates in the abdomen, where it bifurcates into two smaller arteries (the common iliac arteries). The aorta leads oxygenated blood to all parts of the body through the systemic circulation. It is divided into four tracts:

- Ascending aorta: the ascending aorta originates at the orifice of the aortic valve and has a mean length of $5cm$. At its root, the ascending aorta has three bulges between the cusps of the aortic valve and the vessel wall, named the aortic sinuses or sinuses of Valsalva. The right aortic sinus contains the origin of the right coronary artery while the left sinus contains the origin of the left coronary artery. These two arteries supply the heart with oxygenated blood. The posterior aortic

sinus does not give rise to a coronary artery. For this reason, the right, left and posterior aortic sinuses are also called right-coronary, left-coronary and non-coronary sinuses (see fig. 1.1).

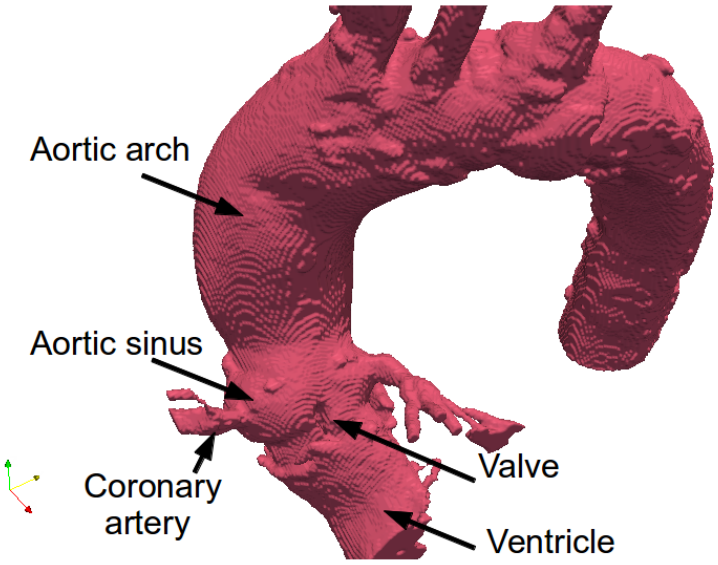


Figure 1.1: An aorta reconstructed from a medical image.

- Aortic arch: the aortic arch originates in the superior mediastinum, and ends at the level of the intervertebral disc between the fourth and fifth thoracic vertebrae. The aortic arch has three major branches: from proximal to distal they are the brachiocephalic trunk, which supplies the right side of the head and neck, as well as the right arm and chest wall, the left common carotid artery, and the left subclavian artery. The latter two together supply the left side of the same regions.

- Thoracic aorta: the thoracic descending aorta gives rise to different arteries which supply bronchi oesophagus, mediastinum, pericardium and diaphragm.

- Abdominal aorta: the abdominal aorta gives rise to arteries which

principally supply kidneys and and bowels. It ends in a bifurcation into the left and right common iliac arteries.

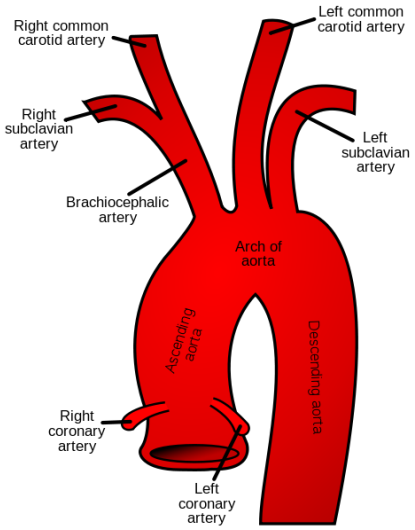


Figure 1.2: Sketch of the aorta and of the arteries starting from it.

Cardiac valves. The cardiac valves are structures that regulates the blood flux inside the heart, allowing the blood to flow in one direction and preventing it to return back. They control the blood flow through the orifices which link the atria with the ventricles and the ventricles with the aorta and the pulmonary artery. Cardiac valves are four and are made of endocardium tissue. Anatomically, they lie all on the same cardiac plane (fig. 1.3). They can be divided in two categories: two atrioventricular valves which are between the atria and the ventricles, and two semilunar valves, which are between the ventricles and the arteries leaving the heart. The atrioventricular valves are the mitral valve (also called the bicuspid valve), and the tricuspid valve. The semilunar valves are the aortic valve and the pulmonary valve.
During the diastolic phase the atrioventricular valves are opened and the blood flows from atria to ventricles. In the systolic phase the ventricles contraction closes the atrioventricular valves and the semilunar valves open

Figure 1.3: Base of ventricles without the atria with the four valves visible.

allowing the blood to flow from ventricles to the arteries (see the Wigger diagram in fig. 1.4 showing the cardiac cycle events occurring in the left ventricle).
In our work we focus our attention to the aortic valve.

Aortic valve. The semilunar aortic valve regulates the blood flow between the left ventricle and the aorta. During the diastolic phase the valve is closed because the pressure in the aorta is bigger then in the left ventricle. During the ventricular systole, the left ventricle pressure quickly rises until reaching the aortic one. In this moment the aortic valve starts its opening allowing blood to exit from the left ventricle into the aorta. The ventricle pressure continues to grow less steeply during the ejection phase, while the aortic valve remains in its full open position. When ventricular systole ends, pressure in the left ventricle rapidly drops and the aortic pressure becomes bigger forcing the aortic valve to close till the next ventricular systole. As we will see in detail in chapter 3, the pressure is not the only factor that affects the moving of the aortic valve. The effects linked to the blood motion, to the action of the vortex downstream of the valve and to the neighbouring tissue resistance must be considered as well.

Concerning the geometry, the aortic valve has a diameter of approxi-

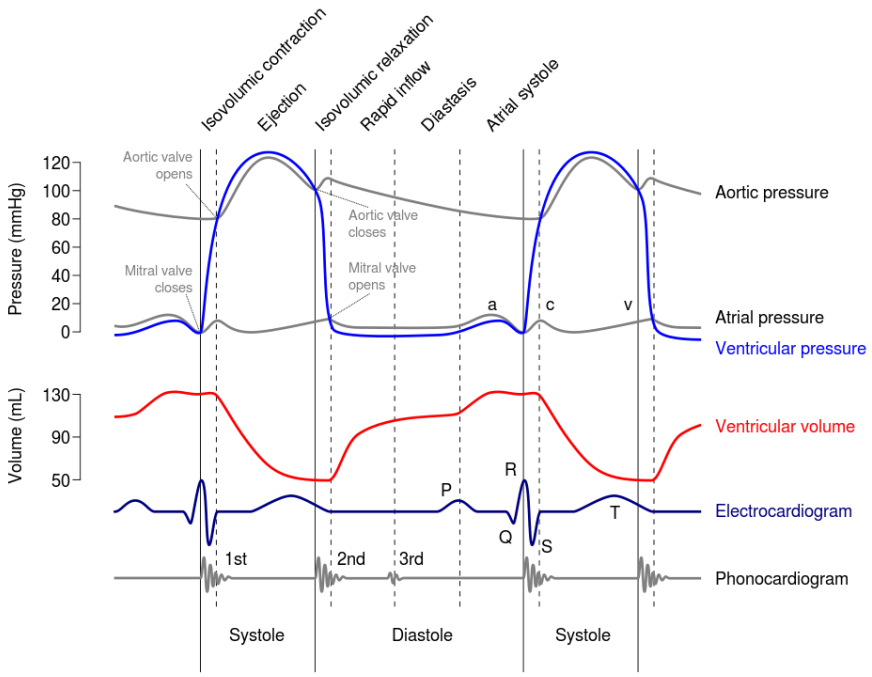


Figure 1.4: Wigger diagram showing the cardiac cycle events occurring in the left ventricle

mately $20mm$ and it is formed by three cusps: left, right, and posterior cusp. The cusps of the heart valves serve to seal the heart valves when closed. The three areas between the cusps and the aortic wall are named aortic sinuses as written before. In fig. 1.6 we highlight how the coronary arteries start from only two of them .

The most common congenital abnormality of the heart is the bicuspid aortic valve. In this condition, instead of three cusps, the aortic valve has two cusps. Besides this congenital disease other pathologies can affect the aortic valve. These pathologies can cause two major effects on the aortic valve: aortic stenosis, in which the valve fails to open fully, thereby obstructing blood flow out from the heart, and aortic insufficiency, also called aortic regurgitation, in which the aortic valve is incompetent and blood flows passively back to the heart in the wrong direction.

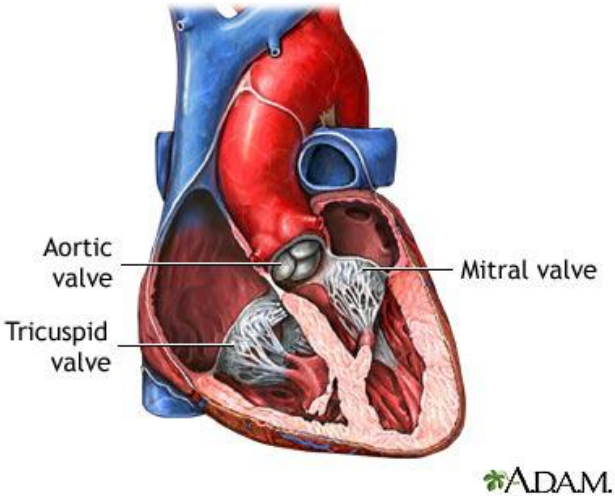


Figure 1.5: Ventricle and aorta with highlighted aortic valve.

Pathological aortic valves compromise cardiovascular regulation and may severely affect quality of life. Because of the high diffusion of valves pathologies in developed countries it is of grate interest the study of heamodynamics inside the aorta and its interplay with valve shape and pathologies.

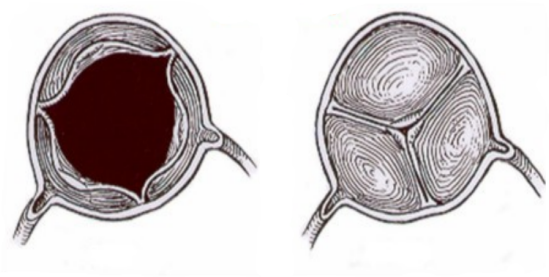


Figure 1.6: Cartoon of aortic valve from the top view (view from the aorta) in open (left) and closed (right) positions.

1.2 Aortic Valve Movement: State of the Art

In the last decades many progresses were accomplished in the field of bio-fluid-mechanical modelling of human cardiovascular system, owing to increased computational resources. Along experimental technique and in-vivo analysis a research field where blood flows are simulated by computational fluid-dynamics (CFD) has been developed. Simulations can be compared to, or sometimes replace altogether real experiments, and shed lights in areas where experiments are difficult to achieve due to many reasons.
In particular, the haemodynamic inside the aorta has been widely studied through CFD analysis but in the majority of the works that can be found in literature the valve itself is not modelled. This is because, in a first approximation, the blood dynamic inside aorta is well described also without valve leaflets inclusion. In this way, pathologies such as aortic aneurysms or aortic coarctation have been widely studied.
Besides this, the inclusion of valve leaflets can improve the numerical solution reliability above all if the interest of the numerical study is the pathology of the valve. For example, in presence of a bicuspid aortic valve, some works [50, 24] has demonstrated the ability of the CFD model without the inclusion of the valve leaflets, to reproduce the correct flow pattern inside the aorta despite it has been highlighted that also the position of the open leaflets can affect the pathological flow in these cases [19, 12, 22]. Moreover, if we are interested in the development of a full integrated model which takes

into account the presence of ventricle and aorta together, the modelling of the aortic valve becomes of primary importance.
For these reasons, there's a growing interest in modelling aortic valve and more specifically its interplay with blood flow inside aorta.
Heart valve models can be divided into two main groups: lumped parameter and multidimensional models. Both of them have limitations. Lumped parameter models are characterized by simplified flow assumptions [33]. Moreover they require the use of artificial boundaries on which the reduced models are defined: the position of these boundaries and the type of boundary condition strongly affect computational results.
As for multidimensional valve models, in spite of the progress made in the last decade, fluid–structure interaction (FSI) simulations remain an extremely challenging task. Among the major computational difficulties we recall the large displacements and the contact of the leaflets which lead to topological changes in the computational domain, the high-pressure jump experienced when the valve is closed and the high computational costs which characterize this type of simulations.
For these reasons in most of the studies the flow equations are solved by employing an Eulerian approach with fixed mesh. Fictitious domain (FD) is a variant of the Eulerian method, where wall boundary conditions are imposed using velocity constraints. De Hart *et al.* [20, 21] used FD combined with an ALE method for the FSI simulation of the aortic valve. However, numerical instabilities forced the authors to use unrealistically low Reynolds numbers. Astorino *et al.* [7] introduced a method for calculating self-contacts of valve leaflets inside the aortic fluid domain using FD formulation. Although they tested the algorithm with an FSI simulation of the aortic valve within a rigid root, they used a very low diastolic transvalvular pressure of $0.1 mmHg$, instead of the physiologic value of approximately $80 mmHg$. Also Peskin's immersed boundary (IB) method [42] was proposed for 3D FSI simulations of the aortic root: Griffith *et al.* [30] used this method to model a healthy aortic valve in a semi-rigid root. However, the authors point out that the spatial resolution may not be fine enough to achieve numerically converged results. Marom *et al.* [37] developed an FSI model that included a compli-

ant root and valve coaptation using physiologically realistic conditions but simulating only the end-closing phase of the valve.
In other works the valve leaflets are included as fixed rigid walls inside the domain [12, 41] or through resistive immersed surface [8], but always in an on/off position without taking into account for their movement.
Beside computational issues, also modeling issues are critical: even if the constitutive law of the valves is assumed to be known, it would be extremely difficult to translates it for a specific patient using the data typically available from clinical examinations. Moreover, in general, the proposed works deals with 2D or 3D aortic models and only few works include patient-specific geometries and data [8, 41, 24].

1.3 Aim

The aim of this thesis is to propose a moving aortic valve model which is less expensive and less challenging in terms of meshing problems than a 3D fluid-structure model but more accurate than a standard lumped parameter model.
In this work we started from the reduced model for heart valves called Resistive Immersed Surface (RIS). This method is proposed originally from Fernández *et al.* [27] to study a porous interface immersed in a fluid and it is used from Astorino *et al.* [8] to model the aortic valve. In this approach, the mechanics of the leaflets is neglected and the valve is replaced by two immersed surfaces fixed in space: one describing the 3D shape of the valve during the opened phase, the other during the closed one. The two valve surfaces are inserted in the classic Navier-Stokes equations adding a dissipative term with a resistance that can be interpreted as a penalization parameter enforcing the condition of null velocity on it. In our work we propose to use the same RIS method of Astorino *et al.* [8], but we extend their model representing the open and close valve surfaces through a level set formulation. In this way, the immersed surfaces are described analytically and we avoid the need of inserting this surfaces in the finite element mesh domain.

Moreover, we propose to compute the movement of the valve between its closed and open position using a reduced zero-dimensional model proposed by Korakianitis *et al.* [33] which computes the valvular angle governed by pressure and flow-rate values in the left ventricle and in the aorta. For each time step of the cardiac cycle, the valvular level set surface is then interpolated starting from the open and closed representations without the need of generating a moving mesh.

Finally our aim is also to deal with patient-specific studies: for this reason we develop a full framework which starts from medical images and finishes with the numerical simulations. This is also an important goal toward the development of patient-specific models to help clinicians in studying and understanding diseases.

Chapter 2

Medical Image Segmentation

This chapter is dedicated to the description of the segmentation method used for patient-specific reconstruction of the aorta. Among the variety of segmentation techniques existing in literature, we choose to use the Split-Bregman minimization of the Region-Scalable Fitting Energy (RSFE) method as proposed by Yang et al. [52] because of its properties to work with inhomogeneous intensity images. In a previous work with L. Barbarotta and F. Cremonesi [26], we implemented this algorithm in a C++ library enriching it with some additional tools to make it more suitable to process medical images. After a brief introduction to the concept of image segmentation (section 2.1), we focus on the mathematical derivation of the algorithm (sections 2.2, 2.3 and 2.4) and we provide a description of our numerical implementation (section 2.5). Finally we talk about our customizations focused on medical applications (section 2.6) showing an example of pre-processing and segmentation in the last section (section 2.7). We refer to the project report [26] for more details about code implementation.

2.1 Introduction to Image Segmentation

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse.
More strictly, image segmentation is the process of assigning a label to every

pixel in an image such that pixels with the same label share certain visual characteristics or computed properties (such as colour, intensity, or texture). Hence, the result of image segmentation is a set of subregions such that each region is significantly different with respect to the characteristics of interest. It is typically used to locate objects and boundaries inside images.
Concerning 3D medical images like X-ray Computed Tomography (CT) or Magnetic Resonance Imaging (MRI), the most common way of looking at them is scrolling through the 2D slices in which they are divided. Generating contours inside a 3D medical image using segmentation allows instead the 3D reconstructions of any human organ. This has lots of practical applications [31], like locating tumours, aneurysm and other pathologies. A particularly interesting use of medical image segmentation in Computational Science & Engineering is the 3D reconstruction of a blood vessel, for instance an artery, to generate a computational 3D domain on which a numerical simulation of the blood flow is performed [11, 24]. This kind of simulation allows to study pathologies and subsequent treatments directly on a patient-specific geometry.

The most used methods in image segmentation are the active contours models [17, 25, 32, 18, 16], which can be divided in two typical approaches: edge-based methods and region-based methods.
Edge-based methods [18, 32, 14] use local edge information to move the active contour toward image boundaries. The ability of these methods relies principally on the definition of a good edge detector and their major limitation is the need of strong edges between objects. Region-based methods [16, 52] use certain region descriptor such as for example mean intensity to guide the active contour. Limits of the majority of region-based methods are due to their assumption of intensity homogeneity inside the regions to be segmented. In fact, intensity inhomogeneity often occurs in real images and in particular in medical images from different modalities.
In our work we implement the Split-Bregman algorithm for the minimization of the Region-Scalable Fitting Energy model (SB-RSFE) [52] because of its ability to work also on regions with intensity inhomogeneities.
In this thesis the treated images are always three dimensional grey scale im-

ages. From a mathematical standpoint, a 3D image can be identified with a domain $\Omega \subset \mathbb{R}^3$ (*i.e.* the image domain) and a function $u_0(\mathbf{x})$ standing for the image intensity distribution over Ω.

2.2 The Region Scalable Fitting Energy Functional

The model is based on the RSFE functional minimization [34] and was created with the aim to overcome the limit of the Piecewise Constant (PC) Chan-Vese model [16].
The idea behind the Chan-Vese model is to divide the image in two regions and approximate the intensity of the image inside each region with a constant value representing the average of the intensity inside the region. The regions are obtained minimizing a functional derived from Mumford-Shah functional [16]. This method is conceived in order to segment images consisting of homogeneous regions and fails to provide the correct segmentation in inhomogeneous images as shown in [34].
The RSFE model can be seen as an improvement of this model. In fact the idea behind is similar but it substitutes the constant values with functions that approximate the image intensity in a local region through the use of a Gaussian kernel. The name comes from this: the method is considered scalable because through the use of the Gaussian kernel with a scale parameter it is possible to decide the size of the stencils on which the average of the intensity around a pixel is calculated.
We notice that the RSFE model with a large scale parameter tends to the PC model; from here we can sense the importance of the scale parameter in RSFE for the segmentation of inhomogeneous images.

The aim of the segmentation algorithm is to divide the image in two parts: the object $\Omega_1 \subset \Omega$ and the background $\Omega_2 \subset \Omega$ with $\Omega_1 \cap \Omega_2 = 0$ and $\Omega_1 \cup \Omega_2 = \Omega$. We notice that the interface between the object and the background is the 3D surface or the 2D contour representing the segmented region; we call contour this interface. The RSFE method is based on a level set formulation. This means that the solution of the method (i.e. the searched

contour) is described as the α-level iso-surface $\mathbf{S}(t) = \{\mathbf{x} \in \Omega : \phi(\mathbf{x}, t) = \alpha\}$ of a scalar function $\phi(\mathbf{x}, t) : \Omega \times \mathbb{R}^+ \to [a_0, b_0]$, where $\alpha = (a_0 + b_0)/2$.
The result of the segmentation is obtained minimizing the subsequent functional called the region scalable fitting energy $\mathcal{F}$:

$$\begin{aligned}\mathcal{F}(\phi, f_1(\mathbf{x}), f_2(\mathbf{x})) &= \int_\Omega \left[\sum_{i=1}^{2} \lambda_i \int_\Omega K_\sigma(\mathbf{x}-\mathbf{y}) \big|u_0(\mathbf{y}) - f_i(\mathbf{x})\big|^2 M_i^\varepsilon(\mathbf{y}) d\mathbf{y} \right] d\mathbf{x} \\ &+ \nu \int_\Omega \big|\nabla M_1^\varepsilon(\mathbf{x})\big| \, d\mathbf{x} \\ &+ \mu \int_\Omega \frac{1}{2}\Big(\big|\nabla\phi(\mathbf{x})\big| - 1\Big)^2 d\mathbf{x}, \end{aligned} \tag{2.2.1}$$

The first term in 2.2.1 is the error we make by approximating the image intensity $u_0(\mathbf{x})$ with the functions $f_i(\mathbf{x})$, which represent the average of intensities in the neighbours of $\mathbf{x}$ defined by K_σ, inside the region Ω_i.
The second term is a penalization on the length/area of the contour: the more the length/area of the contour grows, the more the energy rises.
The third term is introduced in [35] and has the purpose of regularizing the contour forcing the levelset to be similar to a distance function. In fact a levelset with a unitary gradient in each point of the domain takes the value of the distance from its zero level.
In 2.2.1 u_0 is the image intensity, ϕ is the level set function that defines the contour and λ_1, λ_2, ν, μ are positive scalar parameters of the model.
K_σ is the Gaussian kernel with scale parameter σ:

$$K_\sigma(\mathbf{x}) = \frac{1}{(\sqrt{2\pi}\,\sigma)^d} \exp\left\{-\frac{|\mathbf{x}|^2}{2\sigma^2}\right\}, \quad \mathbf{x} \in \mathbb{R}^d. \tag{2.2.2}$$

M_1^ε and M_2^ε are two functions that identify the regions using the following smoothed Heaviside function depending on the smooth parameter ε:

$$\mathrm{H}^\varepsilon(\phi(\mathbf{x})) = \frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{\phi(\mathbf{x}) - (a_0 + \alpha)}{\varepsilon|b_0 - a_0|}\right)\right), \tag{2.2.3}$$

$$M_1^\varepsilon(\mathbf{x}) = \mathrm{H}^\varepsilon\left(\phi(\mathbf{x})\right), \qquad M_2^\varepsilon(\mathbf{x}) = 1 - \mathrm{H}^\varepsilon\left(\phi(\mathbf{x})\right). \tag{2.2.4}$$

f_1 and f_2 are two functions that approximate image intensities in this way:

$$f_i(\mathbf{x}) = \frac{K_\sigma * M_i^\varepsilon u_0}{K_\sigma * M_i^\varepsilon}. \tag{2.2.5}$$

The minimization of this equation leads to the next evolution equation [52]:

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= -\delta_\varepsilon(\phi)\ (\lambda_1 e_1 - \lambda_2 e_2) \\ &+ \nu \delta_\varepsilon(\phi)\ \mathrm{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right) \\ &+ \mu\left(\Delta\phi - \mathrm{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right)\right), \end{aligned} \tag{2.2.6}$$

$$\text{where} \quad e_i(\mathbf{x}) = \int_\Omega K_\sigma(\mathbf{x}-\mathbf{y})\,|u_0(\mathbf{x}) - f_i(\mathbf{y})|^2\,d\mathbf{y}. \tag{2.2.7}$$

Functional (2.2.1) is non convex, so standard minimization methods like gradient descent can be easily trapped in local minima and thus the evolution of the contour would be stopped. Furthermore (2.2.6) is non-linear and its resolution can be very expensive computationally. Our purpose is to minimize (2.2.1) in a different way: we want to apply the Global Convex Segmentation (GCS) method [15] to RSFE in order to make it convex and then minimize the resulting functional with the Split-Bregman algorithm [29]. Before doing this in the next section we discuss in detail the Split-Bregman method.

2.3 The Split-Bregman Method

The Split-Bregman method is a fast way to minimize functionals of the form:

$$\min_{v \in \mathbb{R}^n} |\Phi(v)| + H(v) \tag{2.3.1}$$

where $|\Phi(v)|$ and $H(v)$ are convex functions (Φ could be both scalar or vectorial) [29].
The strategy of the method is:

1. passing from the unconstrained minimization problem to a constrained one decoupling the two terms in (2.3.1) through the introduction of a new variable;
2. reinterpreting the problem as an unconstrained problem by adding a penalization term which replaces the constraint;
3. applying the Bregman iteration to the unconstrained problem;
4. splitting the minimization in the two variables.

As shown in [52] this approach can be generalized to functionals defined on a Hilbert space, like the L^2 space which is the functional space of our application:

$$\min_{u \in L^2(\Omega)} \|\Phi(u)\|_{L^1(\Omega)} + H(u) \tag{2.3.2}$$

where H is a functional defined over $L^2(\Omega)$.

From unconstrained to constrained problem: we decouple the problem (2.3.2) introducing the variable d and enforcing it to be equal to $\Phi(u)$.

$$\begin{cases} \min\limits_{u,d \in L^2(\Omega)} \|d\|_{L^1(\Omega)} + H(u) \\ d = \Phi(u) \end{cases} \tag{2.3.3}$$

Return to the unconstrained problem: now we impose the constraint using a quadratic penalization function:

$$\min_{u,d \in L^2(\Omega)} \|d\|_{L^1(\Omega)} + H(u) + \frac{\lambda}{2}\|d - \Phi(u)\|^2_{L^2(\Omega)} \tag{2.3.4}$$

the functional is now in a form on which we can use the Bregman iteration.

The Bregman iteration: First of all we now introduce a general formulation of the Bregman iteration.
We start from a problem of the following type:

$$\min_{u \in L^2(\Omega)} E(u) + \lambda K(u), \tag{2.3.5}$$

where $E(u)$ and $K(u)$ are convex functionals defined over $L^2(\Omega)$ and $\min_{u\in L^2(\Omega)} K(u) = 0$.
Then, we define the concept of "Bregman Distance":

$$D_E^p(u,v) = E(u) - E(v) - \langle p, u-v\rangle_{L^2(\Omega)} \tag{2.3.6}$$

where p is the subgradient of E at v:

$$p \in \partial E(v) \subset L^2(\Omega), \text{ if } \quad E(w) \geq E(v) - \langle p, w-v\rangle_{L^2(\Omega)} \quad \forall w \in L^2(\Omega). \tag{2.3.7}$$

In particular the last equation implies that $D_E^p(u,v) \geq 0$. Beside, thanks to the convexity of E, $D_E^p(u,v) \geq D_E^p(w,v)$ if $w = \alpha u + (1-\alpha)v$ with $\alpha \in [0,1]$. Anyway $D_E^p(u,v)$ is not a distance in the usual sense because in general it is not symmetric.
We assume for simplicity that $K(u)$ is differentiable. Problem (2.3.5) can be solved by iteratively computing:

$$\begin{aligned} u^{k+1} &= \arg\min_{u\in L^2(\Omega)} \; D_E^p(u,u^k) + \lambda K(u) \\ &= \arg\min_{u\in L^2(\Omega)} \; E(u) - \langle p^k, u-u^k\rangle_{L^2(\Omega)} + \lambda K(u) \\ p^{k+1} &= p^k - \nabla K(u^{k+1}) \end{aligned} \tag{2.3.8}$$

Under the conditions set out above, if the problem (2.3.5) has a solution then the sequence written in (2.3.8) converges to the same solution. Further results about the convergence properties can be found in [40].

Split the minimization in the two variables: We apply the Bregman iteration to the (2.3.4) with $E(u,d) = \|d\|_{L^1(\Omega)} + H(u)$ and $K(u,d) = \frac{1}{2}\|d-$

$\Phi(u)\|^2_{L^2(\Omega)}$ and we obtain:

$$\begin{aligned}(u^{k+1}, d^{k+1}) &= \underset{u,d\in L^2(\Omega)}{\arg\min}\ D_E^p(u, u^k, d, d^k) + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2 \\ &= \underset{u,d\in L^2(\Omega)}{\arg\min}\ E(u,d) - \langle p_u^k, u - u^k\rangle_{L^2(\Omega)} - \\ &\qquad \langle p_d^k, d - d^k\rangle_{L^2(\Omega)} + \frac{\lambda}{2}\|d - \Phi(u)\|^2_{L^2(\Omega)} \qquad (2.3.9)\\ p_u^{k+1} &= p_u^k - \lambda(\nabla\Phi)^T(\Phi(u^{k+1}) - d^{k+1}) \\ p_d^{k+1} &= p_d^k - \lambda(d^{k+1} - \Phi(u^{k+1}))\end{aligned}$$

In [53] it is shown how equations (2.3.9) can be put in the following equivalent form, if the operator Φ is linear:

$$\begin{aligned}(u^{k+1}, d^{k+1}) &= \underset{u,d\in L^2(\Omega)}{\arg\min}\ \|d\|_{L^1(\Omega)} + H(u) + \frac{\lambda}{2}\|d - \Phi(u) - b^k\|^2_{L^2(\Omega)} \\ b^{k+1} &= b^k + (\Phi(u^{k+1}) - d^{k+1})\end{aligned} \qquad (2.3.10)$$

Now we have to solve the problem of minimizing the first equation in (2.3.10) but thanks to the splitting the L^1 and the L^2 portions are now decoupled, so the minimization can be performed iteratively: minimizing first with respect to u and then with respect to d:

$$u^{k+1} = \underset{u\in L^2(\Omega)}{\arg\min}\ H(u) + \frac{\lambda}{2}\|d^k - \Phi(u) - b^k\|^2_{L^2(\Omega)} \qquad (2.3.11)$$

$$d^{k+1} = \underset{d\in L^2(\Omega)}{\arg\min}\ \|d\|_{L^1(\Omega)} + \frac{\lambda}{2}\|d - \Phi(u^{k+1}) - b^k\|^2_{L^2(\Omega)} \qquad (2.3.12)$$

All the terms in (2.3.11) are differentiable so we can use a wide variety of optimization techniques to solve the problem, depending on the properties of $H(u)$.

The speed of this approach depends on how fast we can perform the two minimization subproblems. While the speed of the first subproblem depends on the choice of the solver, for the second subproblem we can use shrinkage operator which is extremely fast:

$$d^{k+1} = shrink(\Phi(u^{k+1}) + b^k, 1/\lambda) \tag{2.3.13}$$

where $shrink(x, \gamma)$ is defined as:

$$shrink(x, \gamma) = \frac{x}{|x|} \max(|x| - \gamma, 0) \tag{2.3.14}$$

In [29] it is shown how it is useless performing more steps of minimization of u and d before updating the Split-Bregman parameter b. Indeed, the further accuracy we would obtain will be wasted when we update b. Furthermore it is not necessary to solve u^{k+1} until convergence, few steps of the iterative method chosen are enough.

2.4 Minimize RSFE using Split-Bregman Method

We now apply the method described above to the functional (2.2.1), but before doing that we have to make some changes in order to make the functional convex. We use the GCS method presented in [15] for this purpose. The first step we need to apply GCS is to drop the last term in equation (2.2.6) dealing to:

$$\frac{\partial \phi}{\partial t} = \delta_\varepsilon(\phi) \left[-\lambda_1 e_1 + \lambda_2 e_2 + \nu \operatorname{div}\left(\frac{\nabla \phi}{|\nabla \phi|} \right) \right] \tag{2.4.1}$$

In [15] is proved that the equilibrium of (2.4.1) is equal to the equilibrium of the same equation omitting $\delta_\epsilon(\phi)$:

$$\frac{\partial \phi}{\partial t} = -\lambda_1 e_1 + \lambda_2 e_2 + \nu \operatorname{div}\left(\frac{\nabla \phi}{|\nabla \phi|} \right) \tag{2.4.2}$$

As shown in [52], solving (2.4.2) corresponds to the minimization of the following energy:

$$E(\phi) = \nu \|\nabla\phi\|_{L^1(\Omega)} + \langle \phi, r \rangle_{L^2(\Omega)} \tag{2.4.3}$$

$$\text{where} \quad r = \lambda_1 e_1 - \lambda_2 e_2 \tag{2.4.4}$$

Note that the energy functional (2.4.3) is now convex but not strictly because of the linear term ensuring the existence of the minimum but not its uniqueness if $\phi \in V$ where V is a generic functional space. To ensure also the uniqueness we have to optimize the functional forcing the solution to lie in a finite interval [52], hence we introduce the subspace $\mathcal{U}_{ad} \subset V$ defined as follow:

$$\mathcal{U}_{ad} := \left\{ \phi \in V : \ a_0 \leq \phi(\mathbf{x}) \leq b_0, \quad q.o. \ \ \mathbf{x} \in \Omega \right\} \tag{2.4.5}$$

We are now ready to apply the Split-Bregman method. As seen in the previous section we have to decouple the L^1 term from the other term. This step is done by adding an auxiliary variable and a constraint on it:

$$\begin{cases} \min\limits_{\phi, \vec{d}} \left\{ \nu \|\vec{d}\|_{L^1(\Omega)} + \left\langle \phi, r \right\rangle_{L^2(\Omega)} \right\} \\ \qquad \vec{d} = \nabla\phi \end{cases} \tag{2.4.6}$$

then imposing the constraint through a quadratic penalization:

$$\min_{\phi, \vec{d}} \left\{ \nu \|\vec{d}\|_{L^1(\Omega)} + \left\langle \phi, r \right\rangle_{L^2(\Omega)} + \|\vec{d} - \nabla\phi\|^2_{L^2(\Omega)} \right\} \tag{2.4.7}$$

and finally applying the Bregman iteration like in (2.3.10):

$$(\phi^{k+1}, \vec{d}^{k+1}) = \underset{\phi \in \mathcal{U}_{ad},\ \vec{d}}{\arg\min} \left\{ \nu \|\vec{d}\|_{L^1(\Omega)} + \left\langle \phi, r^k \right\rangle_{L^2(\Omega)} + \frac{\lambda}{2} \left\| \vec{d} - \nabla\phi - \vec{b}^k \right\|^2_{L^2(\Omega)} \right\} \tag{2.4.8}$$

$$\vec{b}^{k+1} = \vec{b}^k + \nabla\phi^{k+1} - \vec{d}^{k+1} \tag{2.4.9}$$

We now introduce a function into the L_1 norm in order to bring information

about the position of the edges. This function is defined as follows:

$$g\big(u_0(\mathbf{x})\big) = \frac{1}{1+\beta\left|\xi\big(u_0(\mathbf{x})\big)\right|^2}, \quad \beta \in \mathbb{R}^+ \tag{2.4.10}$$

$\xi\big(u_0(\mathbf{x})\big)$ is a function that looks for the edges of the image $u_0(\mathbf{x})$, it assumes values close to zero if the pixels of the image is far from an edge and big values when these pixels are edges of the image. A typical choice is setting ξ as the gradient of the intensity function of the image: $\xi\big(u_0(\mathbf{x})\big) = \nabla\big(u_o(\mathbf{x})\big)$. Hence, in order to make less costly for the contour to move in areas where the image has edges we substitute the L_1 norm in (2.4.8) with the g-norm:

$$\|\vec{v}\|_g = \int_\Omega g\big(u_0(\mathbf{x})\big)\,|\vec{v}(\mathbf{x})|\,d\mathbf{x} \tag{2.4.11}$$

The next step in the Split-Bregman algorithm is the splitting of the equation (2.4.8):

$$\phi^{k+1} = \underset{\phi\,\in\,\mathcal{U}_{ad}}{\arg\min}\left\{\langle\phi, r^k\rangle_{L_2(\Omega)} + \frac{\lambda}{2}\left\|\vec{d}^k - \nabla\phi - \vec{b}^k\right\|^2_{L_2(\Omega)}\right\} \tag{2.4.12}$$

$$\vec{d}^{k+1} = \underset{\vec{d}}{\arg\min}\left\{\ \nu\ \|\vec{d}\|_g + \frac{\lambda}{2}\left\|\vec{d} - \nabla\phi - \vec{b}^k\right\|^2_{L_2(\Omega)}\right\} \tag{2.4.13}$$

$$\vec{b}^{k+1} = \vec{b}^k + \nabla\phi^{k+1} - \vec{d}^{k+1}. \tag{2.4.14}$$

As seen in (2.3.13) minimization of the second equation (2.4.13) can be done using the shrinkage operator. We focus now our attention on the minimization of (2.4.12) showing the equivalence with the resolution of a Poisson problem.

We first rearrange the terms as follows (omitting the superscript k to simplify

notation):

$$\begin{aligned}
\langle \phi, r\rangle_{L_2(\Omega)} \; + \; & \frac{\lambda}{2}\left\|\vec{d} - \nabla\phi - \vec{b}\right\|^2_{L_2(\Omega)} \; = \\
= \; & \langle \phi, r\rangle_{L_2(\Omega)} + \frac{\lambda}{2}\left\langle \vec{d} - \nabla\phi - \vec{b},\; \vec{d} - \nabla\phi - \vec{b}\right\rangle_{L_2(\Omega)} \\
= \; & \frac{\lambda}{2}\langle \nabla\phi,\; \nabla\phi\rangle_{L_2(\Omega)} + \\
& \langle r, \phi\rangle_{L_2(\Omega)} \; + \; \lambda\langle \vec{b},\; \nabla\phi\rangle_{L_2(\Omega)} \; - \; \lambda\langle \vec{d},\; \nabla\phi\rangle_{L_2(\Omega)} + \\
& \frac{\lambda}{2}\left\{\langle \vec{d},\; \vec{d}\rangle_{L_2(\Omega)} \; + \; \langle \vec{b},\; \vec{b}\rangle_{L_2(\Omega)} \; - \; 2\langle \vec{d},\; \vec{b}\rangle_{L_2(\Omega)}\right\} \\
= \; & a(\phi, \phi) \; - \; L\phi \; + \; cost, \qquad (2.4.15)
\end{aligned}$$

where we have defined the bilinear form $a(\phi, v)$ and the linear functional Lv as follows:

$$\begin{aligned}
a(\phi, v) \; &= \; \frac{\lambda}{2}\langle \nabla\phi,\; \nabla v\rangle_{L_2(\Omega)} \\
Lv \; &= \; \lambda\langle \vec{d} - \vec{b},\; \nabla v\rangle_{L_2(\Omega)} \; - \; \langle r, v\rangle_{L_2(\Omega)}
\end{aligned}$$

In the following part of this section we refer for instance to these books [44, 45] for more details on the cited theorems.

Since in (2.4.15) the gradient of ϕ appears, a natural choice is to set the functional space V equal to the Sobolev space $H^1(\Omega)$. Nevertheless we first set $V = H^1_0(\Omega)$ to simplify computations, generalizing later to the more general space $H^1(\Omega)$.

In this case $a(\phi, v)$ is a continuous and coercive bilinear form and Lv is a continuous linear functional, hence all terms in (2.4.15) are Fréchet-differentiable [45].

Since the functional is convex, we can differentiate the equation and put it equal to zero in order to find the minimum [45]:

$$D\left[a(\phi, \phi) \; - \; L\phi \; + \; cost\right][v] \; = \; a(\phi, v) \; - \; Lv \; = \; 0. \qquad (2.4.16)$$

Hence, we have just shown that the minimization problem (2.4.12) is equiv-

alent to the resolution of the following weak formulation:

$$? \ \phi \in V \ : \ a(\phi, v) = Lv, \ \forall v \in V = H_0^1(\Omega). \tag{2.4.17}$$

The existence and the uniqueness of the solution of (2.4.17) is guaranteed by Lax-Milgram theorem [44].
Since we will solve this problem numerically using finite difference method, we are interested in the equivalent strong formulation, obtained as follows using Gauss-Green theorem and the fact that all functions in $H_0^1(\Omega)$ have null trace [44]:

$$\begin{aligned} a(\phi, v) \ - \ Lv \ &= \ \frac{\lambda}{2}\langle \nabla\phi, \ \nabla v\rangle_{L_2(\Omega)} \ + \ \lambda\langle \vec{b} - \vec{d}, \ \nabla v\rangle_{L_2(\Omega)} \ + \ \langle r, v\rangle_{L_2(\Omega)} \\ &= \ -\frac{\lambda}{2}\langle \Delta\phi, \ v\rangle_{L_2(\Omega)} \ + \ \langle \partial_{\vec{n}}\phi, \ v\rangle_{L_2(\partial\Omega)} \ + \\ &\quad \lambda\langle div(\vec{d} - \vec{b}), \ v\rangle_{L_2(\Omega)} \ + \ \lambda\langle (\vec{b} - \vec{d}) \cdot \vec{n}, \ v\rangle_{L_2(\partial\Omega)} \ + \\ &\quad \langle r, v\rangle_{L_2(\Omega)} \\ &= \ -\frac{\lambda}{2}\langle \Delta\phi, \ v\rangle_{L_2(\Omega)} \ + \ \lambda\langle div(\vec{d} - \vec{b}), \ v\rangle_{L_2(\Omega)} \ + \ \langle r, v\rangle_{L_2(\Omega)}, \end{aligned}$$

that is equal to the following Dirichlet Poisson Problem:

$$\begin{cases} -\Delta\phi \ = \ \text{div}\left(\vec{b}^k - \vec{d}^k\right) - \dfrac{r^k}{\lambda} & in \ \Omega \\ \quad \phi \ = \ 0 & on \ \partial\Omega \end{cases} \tag{2.4.18}$$

To extend this result to the more general case of $\phi \in H^1(\Omega)$ we can simply define the solution of the minimization problem (2.4.12) as $\phi = \widetilde{\phi} + R_g$ where $\widetilde{\phi} \in H_0^1(\Omega)$ (hence is null on the boundary) and R_g is a selected function in $H_1(\Omega)$ such that is equal to a function g in the boundary of the domain Ω ($g : g = \phi \ on \ \partial\Omega$). Proceeding again as before we obtain the same equation (2.4.18) with $\widetilde{\phi}$ in place of ϕ and with ΔR_g as a new addend in the forcing function. Substituting $\widetilde{\phi} = \phi - R_g$ we arrive at the following Dirichlet Poisson

Problem:

$$\begin{cases} -\Delta\phi &= \text{div}\left(\vec{b}^k - \vec{d}^k\right) - \dfrac{r^k}{\lambda} \qquad in\ \Omega \\ \phi &= g \qquad\qquad\qquad\qquad on\ \partial\Omega \end{cases} \tag{2.4.19}$$

Projecting the solution in the space $\mathcal{U}_{ad} \subset V$ defined in (2.4.5) (with $V = H^1(\Omega)$) we obtain the solution of the problem (2.4.12).
Hence, we can finally summarise as follow the procedure to minimise functional (2.4.3):

$$\begin{cases} \phi^{k+1} &= \text{solution of (2.4.19)} \\ \textbf{project} \quad \phi^{k+1}\ on\ \ \mathcal{U}_{ad} \\ \vec{d}^{k+1} &= shrink\left(\vec{b}^k + \nabla\phi^{k+1},\ \dfrac{\nu}{\lambda}\, g\right) \\ \vec{b}^{k+1} &= \vec{b}^k + \nabla\phi^{k+1} - \vec{d}^{k+1} \end{cases} \tag{2.4.20}$$

Since the rhs of (2.4.19) $\in L^2(\Omega)$ (to ensure this it is sufficient to initialize $\vec{b}^0 = \vec{d}^0 = 0$) and the domain is convex, we can apply the global elliptic regularity theorem [44] stating that $\phi \in H^2(\Omega)$. Thanks to the Sobolev embedding theorem [44], this implies that ϕ is a continuous function either in the 2D or in the 3D case. Hence, it makes sense to speak of point values of ϕ and, in order to solve (2.4.19) with a numerical method, also the finite difference method is allowed (implementation details will be discussed in the next section).
Note finally that, as suggested in [52], in (2.4.20) ν can be set equal to one without loss of generality.

As we will see in the last chapter, it could be useful to see the solution of (2.4.19) as the equilibrium ϕ_* of the corresponding unsteady parabolic problem (2.4.21):

$$\begin{cases} \dfrac{\partial\phi}{\partial t} - \Delta\phi &= \text{div}\left(\vec{b}^k - \vec{d}^k\right) - \dfrac{r^k}{\lambda} \qquad in\ \Omega \times (0,T] \\ \phi(t,\mathbf{x}) &= g \qquad\qquad\qquad\qquad on\ \partial\Omega \times (0,T] \\ \phi(0,\mathbf{x}) &= \phi_0 \qquad\qquad\qquad\qquad in\ \Omega \end{cases} \tag{2.4.21}$$

This has been done because the robustness of the Split-Bregman method allows us not to solve (2.4.21) until equilibrium, but few temporal steps are sufficient [40]. Implementation details will be shown in section 2.5.
In algorithm 1 we show a pseudocode of the whole algorithm.

Algorithm 1 Minimization of RSFE with Split-Bregman

$\phi^0 \leftarrow$ *initial contour* ▷ initialize levelset
$\vec{b}^0 \leftarrow 0$ ▷ initialize Split-Bregman variables
$\vec{d}^0 \leftarrow 0$
while $\|\phi^{k+1} - \phi^k\| > tol$ **do**
  update $M_i^\varepsilon(\mathbf{x})$ ▷ update Ω_i computing $\mathcal{H}^\varepsilon(\phi^k(x))$ (eq. (2.2.4))
  update $e_i^k(\mathbf{x})$ ▷ compute convolutions to update e_i^k (eq. (2.2.7))
  $r^k \leftarrow \lambda_1 e_1^k - \lambda_2 e_2^k$
  $\phi^{k+1} \leftarrow PDE\ solver\left(\phi^k, r^k, \vec{d}^k, \vec{b}^k, \lambda\right)$ ▷ compute new levelset (eq. (2.4.21))
  $a_0 \leq \phi^{k+1} \leq b_0$ ▷ take back ϕ^{k+1} in the range $[a_0, b_0]$
  $\vec{d}^{k+1} \leftarrow shrink\left(\vec{b}^k + \nabla\phi^{k+1}, \frac{g}{\lambda}\right)$ ▷ update Split-Bregman variables
  $\vec{b}^{k+1} \leftarrow \vec{b}^k + \nabla\phi^{k+1} - \vec{d}^{k+1}$
end while

2.5 Numerical Implementation

In our implementation we consider all functions introduced in previous sections as discrete functions defined in a uniform grid. This is a common choice in image processing because pixels naturally define a uniform grid and it is not possible to go over pixels definition without 'inventing information'.
We have implemented the algorithm both for 2D and 3D images. The main difference is that usually two-dimensional images are characterized from square pixels and therefore they generate uniform grid easier to handle as a discrete signal; on the contrary three-dimensional ones have got pixels with different spacing in each direction called voxels and we have to consider this especially in convolution computation. Hence, to implement Algorithm 1 in our discrete space we follow these steps:

1. We simply implement all algebraic operations as pixels-by-pixels operations.

2. We approximate continuous convolution products with discrete convolution evaluating continuous functions in each pixel and solving convolution using Discrete Fourier Transform (DFT) with Fast Fourier Transform (FFT) algorithm to improve performance.

3. We use finite difference to discretize PDE (2.4.21) and every differential operator.

4. We solve the linear system generated by the finite difference discretization of (2.4.21) using Gauss-Seidel iterative method without saving the matrix to prevent the excessive use of memory.

Convolution: solving convolution products is the most expensive thing we have to do at each iteration. In fact, first we have to solve four convolution products to compute f_i as seen in equation (2.2.5) and then we can compute e_i decomposing equation (2.2.7) in other three convolution products as follows:

$$\begin{aligned} e_i(\mathbf{x}) &= \int_\Omega K_\sigma(\mathbf{x}-\mathbf{y})\,|u_0(\mathbf{x}) - f_i(\mathbf{y})|^2\,d\mathbf{y} \\ &= |u_0|^2\,(\mathbb{1}_\Omega * K_\sigma) \;+\; \left(|f_i|^2 * K_\sigma\right) \;-\; 2u_0\,(f_i * K_\sigma) \end{aligned} \tag{2.5.1}$$

Hence we have a total of ten convolution products for each iteration.
In this paragraph at first we will see how to minimize the number of this products and secondly we will show how to solve each convolution in the fastest possible way using Fast Fourier Transform algorithm.

We first define two functions that don't depend on variables changing during the algorithm, so we can compute them before the while cycle starts:

$$\mathcal{K}_{\mathbb{1}_\Omega} \;=\; \mathbb{1}_\Omega * K_\sigma \qquad \mathcal{K}_{u_0} \;=\; u_0 * K_\sigma \tag{2.5.2}$$

We can now halve the convolution products needed to compute f_i at each

iteration:

$$num \;=\; M_1^{\varepsilon} u_0 * K_\sigma, \qquad den \;=\; M_1^{\varepsilon} * K_\sigma, \tag{2.5.3}$$

$$f_1 \;=\; \frac{num}{den}, \qquad f_2 \;=\; \frac{\mathcal{K}_{u_0} - num}{\mathcal{K}_{\mathbb{1}_\Omega} - den}, \tag{2.5.4}$$

where in (2.5.4) we have obtained the expression of f_2 exploiting linearity property of the convolution in equation (2.2.5):

$$f_2 \;=\; \frac{M_2^{\varepsilon} u_0 * K_\sigma}{M_2^{\varepsilon} * K_\sigma} \;=\; \frac{[(1 - M_1^{\varepsilon})u_0] * K_\sigma}{(1 - M_1^{\varepsilon}) * K_\sigma} \;=\; \frac{u_0 * K_\sigma \;-\; M_1^{\varepsilon} u_0 * K_\sigma}{\mathbb{1}_\Omega * K_\sigma \;-\; M_1^{\varepsilon} * K_\sigma}.$$

If we proceed as follows we can reduce the total number of convolution products in each iteration at only four:

$$r_1 \;=\; \lambda_1 |f_1|^2 - \lambda_2 |f_2|^2, \qquad r_2 \;=\; \lambda_1 f_1 - \lambda_2 f_2, \tag{2.5.5}$$

$$\mathcal{K}_{r_1} \;=\; r_1 * K_\sigma, \qquad \mathcal{K}_{r_2} \;=\; r_2 * K_\sigma, \tag{2.5.6}$$

$$r \;=\; (\lambda_1 - \lambda_2)|u_0|^2 \mathcal{K}_{\mathbb{1}_\Omega} \;+\; \mathcal{K}_{r_1} \;-\; 2u_0 \mathcal{K}_{r_2}, \tag{2.5.7}$$

where to derive equation (2.5.7) it is sufficient to do simple algebraic calculations:

$$\begin{aligned} r \;&=\; \lambda_1 e_1 - \lambda_2 e_2 \\ &=\; \lambda_1 \left\{ |u_0|^2 (\mathbb{1}_\Omega * K_\sigma) \;+\; \left(|f_1|^2 * K_\sigma \right) \;-\; 2u_0 (f_1 * K_\sigma) \right\} \\ &\quad -\lambda_2 \left\{ |u_0|^2 (\mathbb{1}_\Omega * K_\sigma) \;+\; \left(|f_2|^2 * K_\sigma \right) \;-\; 2u_0 (f_2 * K_\sigma) \right\} \\ &=\; (\lambda_1 - \lambda_2)|u_0|^2 \mathcal{K}_{\mathbb{1}_\Omega} \;+\; (\lambda_1 |f_1|^2 - \lambda_2 |f_2|^2) * K_\sigma \;-\; 2u_o \left[(\lambda_1 f_1 - \lambda_2 f_2) * K_\sigma \right]. \end{aligned}$$

Hence, as outlined in Algorithm 2, it is sufficient to compute, before starting the cycle, $\mathcal{K}_{\mathbb{1}_\Omega}$ and $\mathcal{K}_{u_0}$ using (2.5.2) and at each iteration *num* and *den* using (2.5.3) and $\mathcal{K}_{r_1}$ and $\mathcal{K}_{r_1}$ using (2.5.6). All the other calculations are algebraic operations.

Algorithm 2 RSFE Split-Bregman: convolution products

1: $\cdots$

2: $\mathcal{K}_{\mathbb{1}_\Omega} \leftarrow \mathbb{1}_\Omega * K_\sigma$ ▷ initialize constant convolution
3: $\mathcal{K}_{u_0} \leftarrow u_0 * K_\sigma$
4: **while** $\|\phi^{k+1} - \phi^k\| > tol$ **do**
5: update $M_i^{\varepsilon,k}$
6: update $num^k,\ den^k$ ▷ convolution using (2.5.3)
7: update f_i^k ▷ eq. (2.5.4)
8: update $\mathcal{K}_{r_i}^k$ ▷ convolution using (2.5.6)
9: update r^k ▷ eq. (2.5.7)

10: $\cdots$

11: **end while**

Let's see now how we can discretize the convolution integral. First of all note that in our convolution products one of the factor is always the Gaussian Kernel K_σ defined in (2.2.2). Therefore we have to solve in a discrete space this equation:

$$g(\mathbf{x}) \;=\; K_\sigma * f \;=\; \int_{\mathbb{R}^n} K_\sigma(\mathbf{x}-\mathbf{y}) f(\mathbf{y})\, d\mathbf{y}, \qquad n = 2, 3. \tag{2.5.8}$$

The only way to use DFT to solve (2.5.8) is to approximate it with the midpoint/rectangle rule in order to transform it in a Discrete Convolution.

We do it explicitly in the three-dimensional case (2D is easier):

$$\begin{aligned} g(\mathbf{x}_n) &= \sum_{i,j,k\,=-\infty}^{+\infty} h_x h_y h_z \; K_\sigma(\mathbf{x}_n - \mathbf{y}_{i,j,k}) \; f(\mathbf{y}_{i,j,k}) \\ &\simeq \sum_{i=-M_x}^{M_x} \sum_{j=-M_y}^{M_y} \sum_{k=-M_z}^{M_z} h_x h_y h_z \; K_\sigma(\mathbf{x}_n - \mathbf{y}_{i,j,k}) \; f(\mathbf{y}_{i,j,k}) \end{aligned} \tag{2.5.9}$$

$$\text{where} \qquad M_* \;=\; \lfloor\, 3 \; h_* \; \sigma \,\rfloor \tag{2.5.10}$$

and h_* are the dimension in each direction of the pixel/voxel. In (2.5.9) we assume the Gaussian kernel null if we are at a distance greater than 3σ from the central value.

We can now apply the Convolution Theorem. Let's see before a complete version of it where we indicate with $\mathcal{F}$ the Fourier Transform, with DTFT the Discrete Time Fourier Transform and with DFT the Discrete Fourier Transform:

Theorem 1. The Fourier transform of a convolution is the point-wise product of Fourier transforms. In other words, convolution in time domain equals point-wise multiplication in the frequency domain as can be seen in following formulas:

$$f(\mathbf{x}) * g(\mathbf{x}) = \mathcal{F}^{-1}\big\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\big\} \tag{2.5.11}$$

$$f(\mathbf{x}_n) * g(\mathbf{x}_n) = \mathrm{DTFT}^{-1}\Big[\mathrm{DTFT}\,\{f(\mathbf{x}_n)\} \cdot \mathrm{DTFT}\,\{g(\mathbf{x}_n)\}\Big] \tag{2.5.12}$$

$$f_T(\mathbf{x}_n) * g(\mathbf{x}_n) = \mathrm{DFT}^{-1}\Big[\mathrm{DFT}\,\{f(\mathbf{x}_n)\} \cdot \mathrm{DFT}\,\{g(\mathbf{x}_n)\}\Big] \tag{2.5.13}$$

In the last equation we denote with f_T the periodic expansion of the function f defined on a finite and discrete interval (e.g. in 1D this means: $f_T(\mathbf{x}_n + mT) = f(\mathbf{x}_n), \forall m \in \mathbb{Z}$). The resulting convolution (2.5.13) is called Circular Convolution. In fact in this product boundary values of opposite edges of original factor f interact each other because of periodization.
Now to use FFT algorithm we need to use DFT, hence we would like to transform equation (2.5.9) in order to apply (2.5.13). We are dealing with

two discrete and finite factors, but we don't need to perform the Circular Convolution but the Discrete one. The solution is the zero padding method. To simplify the procedure, we first assume the two factors f and K_σ as one-dimensional denoting respectively with N and M their number of elements. We can proceed as follows:

1. add $M-1$ zeros at the end of factor f (zero-padding f).
2. add $N-1$ zeros at the end of factor K_σ (zero-padding K_σ), note that both the two factors have got now $N+M-1$ elements.
3. compute DFT of zero-padded f and K_σ.
4. multiply the DFT of the two zero-padded factors as in equation (2.5.13).
5. compute DFT^{-1} of the product just computed.

Adding the correct number of zeros ensures that there are no interactions between values of different bounds. Hence, zero padding the two factors allows to compute Discrete Convolution using DFT.
It is easy to generalize this procedure in the multi-dimensional case. For instance, in case of equation (2.5.9) we have to add $\big[(M_x-1)\times(M_y-1)\times(M_z-1)\big]$ zeros to the factor f and $\big[(N_x-1)\times(N_y-1)\times(N_z-1)\big]$ zeros to the factor K_σ in order to transform both in factors of dimensions $\big[(N_x+M_x-1)\times(N_y+M_y-1)\times(N_z+M_z-1)\big]$. Then it is sufficient to perform a 3D DFT instead of the one-dimensional.
Last thing to underline is that the result of Discrete Convolution is obviously a discrete function with $\big[(N_x+M_x-1)\times(N_y+M_y-1)\times(N_z+M_z-1)\big]$ elements. However we are not interested to go outside the range $\big[N_x\times N_y\times N_z\big]$ because the other functions in the algorithm has got these dimensions. Hence, at the end, we crop the external frame of the resulting image to reduce dimensions to the original ones.

Let's conclude the paragraph analysing performance of the approach just described. Because K_σ does not change during the algorithm, we can compute its DFT at the beginning, hence every convolution costs only a Forward Fourier Transform (of the factor f), a multiplication ($\mathrm{DFT}\{f\}\times\mathrm{DFT}\{K_\sigma\}$)

and the final Inverse Fourier Transform. Considering that the FFT algorithm to compute a single DFT has a complexity of $\Theta\big((N+M)\times\log(N+M)\big)$, we obtain a really good improvement of performance using this approach. In fact computing direct convolution as in equation (2.5.8) costs $\Theta(N\times M)$ and this means a great loss of time considering that in our application we usually have $M\simeq N$. Moreover a more precise quadrature formula in (2.5.8) would increase complexity even more.
Despite DFT approach leads to the midpoint rule (that is the most approximate quadrature rule), it is the only way to solve convolution quickly if you are dealing with a quite big three-dimensional image like in medical applications.

PDE resolution with Finite Difference and Gauss-Seidel: in this paragraph we discuss how to solve in a discrete space the partial differential equation (2.4.21). We first have a look at the finite difference scheme adopted to approximate differential operator and then we discuss the resolution of the linear system with the Gauss-Seidel method.

The use of finite difference method to discretize differential operator is a very natural choice in image processing because, as described at the beginning of this section, images are naturally defined on a uniform grid perfect to compute finite difference.
We start describing the approximation of the spatial differential operator. To approximate the derivatives of the first order we use a second order finite difference scheme. For non-border values, central differences (2.5.14) are used whereas for border values a second order preserving scheme with three points (2.5.15) is implemented:

$$\frac{\partial f(\mathbf{x}_i)}{\partial x} = \frac{f(\mathbf{x}_{i+1})-f(\mathbf{x}_{i-1})}{2h_x} + o(h_x^2), \tag{2.5.14}$$

$$\frac{\partial f(\mathbf{x}_i)}{\partial x} = \frac{-3f(\mathbf{x}_i)+4f(\mathbf{x}_{i+1})-f(\mathbf{x}_{i+2})}{2h_x} + o(h_x^2). \tag{2.5.15}$$

Previous equations refer to the discretization in x direction and we omit indexes j and k because they are constant in each derivative computation.

It is sufficient to replace x with y or z and i with j or k to have the correct formulation of the other two directions.
We now group all explicit terms of (2.4.21) in a unique forcing function:

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} = \Delta\phi(\mathbf{x},t) + F^k(\mathbf{x}) \quad (2.5.16)$$

$$\text{where} \quad F^k(\mathbf{x}) = \nabla \cdot \left(\vec{b}^k(\mathbf{x}) - \vec{d}^k(\mathbf{x})\right) - \frac{r^k(\mathbf{x})}{\lambda}. \quad (2.5.17)$$

Hence, to compute F^k we can simply apply (2.5.14) and (2.5.15) to approximate the divergence operator.
For the laplacian operator in the three-dimensional case (2D is analog) we use the following second order scheme:

$$\begin{aligned}\Delta f(\mathbf{x}_{i,j,k}) = {} & \frac{f(\mathbf{x}_{i+1,j,k}) + f(\mathbf{x}_{i-1,j,k})}{h_x^2} + \\ & \frac{f(\mathbf{x}_{i,j+1,k}) + f(\mathbf{x}_{i,j-1,k})}{h_y^2} + \\ & \frac{f(\mathbf{x}_{i,j,k+1}) + f(\mathbf{x}_{i,j,k-1})}{h_z^2} - \\ & \left(\frac{2}{h_x^2} + \frac{2}{h_y^2} + \frac{2}{h_z^2}\right) f(\mathbf{x}_{i,j,k}) + \quad o(h_{x,y,z}^2). \end{aligned} \quad (2.5.18)$$

We can now proceed with the temporal discretization as follows:

$$\frac{\phi(\mathbf{x}_{i,j,k}, t+1) - \phi(\mathbf{x}_{i,j,k}, t)}{dt} = \Delta\phi(\mathbf{x}_{i,j,k}, t^*) + F(\mathbf{x}_{i,j,k}) \quad (2.5.19)$$

where we omit the Split-Bregman superscript index k to simplify the notation and we consider now t as the discrete time and dt the time step. Now we focus our attention on the discretization of the laplacian operator. Choosing $t^* = t$ we obtain an explicit method, choosing $t^* = t+1$ we obtain an implicit method. Instead Gauss-Seidel is a semi-implicit method, that generates a lower triangular matrix, also sparse for our equation. This is possible by evaluating in equation (2.5.18) all pixels with a $+1$ index ($i+1$ or $j+1$ or $z+1$) in an explicit way ($t^* = t$) and all pixels with a -1 index ($i-1$ or

$j-1$ or $z-1$) in a implicit way ($t^* = t+1$) as shown in (2.5.20):

$$\tilde{H}\ \phi_{i,j,k}^{t+1} = \frac{\phi_{i+1,j,k}^{t} + \phi_{i-1,j,k}^{t+1}}{h_x^2} + \frac{\phi_{i,j+1,k}^{t} + \phi_{i,j-1,k}^{t+1}}{h_y^2} + \frac{\phi_{i,j,k+1}^{t} + \phi_{i,j,k-1}^{t+1}}{h_z^2} + \frac{\phi_{i,j,k}^{t}}{dt} + F_{i,j,k} \tag{2.5.20}$$

$$\text{where} \qquad \tilde{H} = \frac{1}{dt} + \frac{2}{h_x^2} + \frac{2}{h_y^2} + \frac{2}{h_z^2} \tag{2.5.21}$$

Leaving only the explicit terms on the right of the equal we can easily see the discretization as a linear system $Ax = b$:

$$\tilde{H}\ \phi_{i,j,k}^{t+1} - \frac{\phi_{i-1,j,k}^{t+1}}{h_x^2} - \frac{\phi_{i,j-1,k}^{t+1}}{h_y^2} - \frac{\phi_{i,j,k-1}^{t+1}}{h_z^2} =$$

$$= \frac{\phi_{i+1,j,k}^{t}}{h_x^2} + \frac{\phi_{i,j+1,k}^{t}}{h_y^2} + \frac{\phi_{i,j,k+1}^{t}}{h_z^2} + \frac{\phi_{i,j,k}^{t}}{dt} + F_{i,j,k} \tag{2.5.22}$$

On the right side of the equation we have all known terms, instead on the left one we have different coefficients multiplying the unknown ϕ^{t+1} evaluated in different pixels in order to form a lower triangular matrix.
Note that if we choose the homogeneous Dirichlet condition the boundary values are simply strongly imposed to zero.
An advantage related to the use of Gauss-Seidel is that the system can be solved implementing three nested *for-cycles*. See [26] for more details about this implementation.
We conclude this paragraph underlining again that, despite to obtain the equilibrium of this equation we need usually lots of steps of Gauss-Seidel method, considering that this problem is nested in the other iterative problem of Split-Bregman, thanks to the robustness of the last one, it is sufficient to do only few steps in order to ensure the global convergence of the whole algorithm [40]. Furthermore, as shown in [49], no stability condition are needed for the time step dt because the matrix generated from the discretization of the laplacian operator in a uniform grid is positive-definite and in this case the Gauss-Seidel is unconditionally stable.

2.6 Customizations and Additional Tools

In this section we discuss the additional tools that we decide to add to the algorithm in order to make it more suitable to the typical application in medical image segmentation described in section 2.1. In fact the use of this algorithm leads to two typical problems:

1. We are usually interested in segmenting only an object inside an image (typical a human organ in medical images) and not all the objects inside of it. However this is a global algorithm which segments all the objects it founds in the analysed image. This can not be a problem if objects are far from each other (unusual in medical images), instead, if they are near, with the progress of the iterations of the algorithm they tend to stick to each other.

2. The algorithm divides the image in only two regions (Ω_1 and $Omegs_2$) and to do that it analyses the intensity of the image u_0. Hence, if the object we would like to segment does not have a good contrast with the background, the algorithm typically considers the object in which we are interested joined to the background without segmenting it. For instance this happens in medical images if the organ is grey on a slightly darker background and in the image there are also parts very clear (white) and parts very dark (black).

To solve the first problem we implemented an algorithm to extract a connected component and we allowed its application to the levelset evolution method. To solve the second problem we implemented a set of pre-processing tools.

Connected Component (CC): the extraction of a Connected Component (CC) of an image is a morphological algorithm that works in black and white images as follow [46]:

1. let the user choose a pixel as a starting point;

2. starting from this pixel, the algorithm looks at its neighbours and only if they are of the same colour of the starting point it marks them as pixels of the connected component;

3. algorithm proceeds iteratively and it ends only if current iteration has not made any changes to the previous connected component.

To apply the algorithm to the levelset ϕ we first transform it in a black and white image using the threshold α. This means that if a pixel $\mathbf{x}$ is such that $\phi(\mathbf{x}) \in [\alpha, b_0]$ we mark it as white, otherwise we mark it as black ($\phi(\mathbf{x}) \in [a_0, \alpha]$).
We can now allow the user (after choosing the initial pixel) to extract a connected component from the current levelset at each iteration he wants. After showing the result of the extraction, the user can also decide if he wants to reinitialize the algorithm setting the connected component just shown as the initial levelset. This makes the algorithm more interactive but also slower because the user, each time he wants to perform an extraction, has to set the initial pixel and choose if he wants to reinitialize the algorithm or not. To avoid these slowdowns we add also a fully automatic mode of extraction. This means that the user has only to set before the beginning of the algorithm the initial pixel $\mathbf{x}$ inside the object he would like to segment and the frequency n with which he would like to perform the connected component extraction during execution. When the algorithm starts, it evaluates at each iteration the value of $\phi(\mathbf{x})$, if it is white ($\phi(\mathbf{x}) \in [\alpha, b_0]$) it proceeds with the automatic extraction, it reinitializes the algorithm setting the connected component just extracted as the initial levelset and it restarts the algorithm from here executing other n iterations before performing a connected component extraction again.
This procedure generally improves the results of the algorithm and allows it to be more local segmenting only a selected element of an image. As we will show in section 2.7 the connected component extraction method could help also during the preprocessing step or in some images of very good quality could represent itself a quick method to obtain a partial segmentation. When used as a preprocessing method, an essential parameter with whom user can

play is the percentage threshold used by the method to firstly transform the image in a black & white image. This will became clearer in the example shown in subsection 2.7.1.

Additional Tools: the additional tools we have implemented have the aim of doing a pre-processing on the image to avoid the second problem described at the beginning of this section or, more in general, to improve the quality of the image we want to segment. In this paragraph we just describe quickly the most important ones implemented.

1. **Crop** allows user to quickly crop an image in a smaller one. This is the first step to do before starting the segmentation algorithm because the more an image is big, the more the algorithm is slow and it needs more memory. Hence, first of all it is really suggested to crop the image you would like to segment in order to obtain the smallest image as possible, containing the object of interest.

2. **Change Resolution** allows to decrease resolution of an image or to increase it linearly interpolating the original values. This can be useful if, also after a crop operation, the image is too big and accordingly the algorithm is too slow. In this case a good strategy can consist in the execution of the algorithm on the image with a lower resolution obtaining a less precise solution, following by the rerunning of it on the original image initializing the levelset with the interpolated solution of the step with lower resolution.

3. **Select Range of Intensity** allows the user to select only those pixels $\mathbf{x}$ such that $u_o(\mathbf{x}) \in [c, d]$ where $[c, d]$ is the desired range. Pixel values are then changed as follows: $u_o(\mathbf{x}) = c$ if $u_o(\mathbf{x}) < c$ and $u_o(\mathbf{x}) = d$ if $u_o(\mathbf{x}) > d$. Optionally external values can also all be set to the value c or d (for example $u_o(\mathbf{x}) = c$ if $d < u_o(\mathbf{x}) < c$). This is the simplest way to try to solve the problem of low contrast in the region of interest. For instance if you are dealing with a medical image with values in the range $[0, 1000]$ and you want to segment an object with values

in a range of $[300, 350]$ in a background characterized from a range $[250, 300]$ it is quite sure that our algorithm does not work. Instead if you select from the original image the range of intensity $[250, 350]$ killing all others pixels our algorithm will surely work better.

4. **Histogram Equalization** is a more advanced algorithm to improve the contrast of an image, useful to be applied for example after the selection of a range of intensity.

5. **Median Filter** is instead a non linear filter that helps to cut outliers values from an image. This allows to remove noise like salt & pepper [48]. It is suggested to apply it before the algorithm starts if the original image is pimpled or with a clear presence of noise.

6. **Linear Filters** are all implemented using the convolution product in the way described in section 2.5. The output of this kind of filters strictly depends on the filter chosen as first factor of convolution [48]. For instance the convolution with a Gaussian kernel is generally called Gaussian Blur because produces as output a blurred image, with the entity of the blur depending on the value of the variance σ of the kernel. Other implemented methods are a Laplacian mask or a Sharpness filter, both conceived to highlight edges of the input image.

2.7 Application to Medical Image

In this section we see an example of application of our code to perform both pre-processing and segmentation. As already discussed in the introduction, the purpose of this master's thesis is to perform, in a patient-specific aorta geometry, a numerical simulation that includes the valve movement. For that purpose we want also to reconstruct the leaflets of the valve from the 3D medical image.
Since generally leaflets are totally or partially not visible in medical images, a crucial factor is the quality of the medical data. Despite in Magnetic Resonance Images (MRI) with contrast agent usually the blood vessel of

interest appears really clear, leaflets are usually not visible in this kind of images. On the contrary High-Resolution Contrast-Enhanced Computed Tomography (CT) produces an output a little more noisy but in which the leaflets are at least partially observable. We will talk about the possibility of reconstructing the leaflets also if they are only partially visible in chapter 4, focusing now only on the pre-processing to make the aorta clearly visible on this kind of images and on the application of the segmentation algorithm for the extraction of the aorta domain.

Our medical data were provided by the hospital *Ospedale Sacco* of Milan thanks to the collaboration with Dr. Roberto Scrofani (Divisione di Cardiochirurgia) and Dr. Sonia Ippolito (Divisione di Radiologia). The data relate to patients with an artificial heart valve implanted and clinicians are interested in the comparison of the hemodynamic of two kind of valvular prosthesis: stentless and stented. In the master's thesis of E. Orso [39] a preliminar study is done performing Fluid–Structure Interaction (FSI) simulations in a patient-specific geometry. The valve geometry was not modeled in this study and the two kind of prosthesis were modeled simply changing the wall elasticity. On the contrary, in our work we not perform any FSI simulation yet, starting to consider only the case of rigid wall and fluid simulation. Nevertheless, a natural development of this work is the inclusion of the structure model to perform an FSI simulation and to obtain new results on the study of the two kind of prosthesis.

More in detail, the image on which we have focused our attention concerns a patient with a stentless valve, implanted perfectly and that moves like that of an healthy person. The High-Resolution CT is made of ten 3D images taken at different percentages of the cardiac cycle: two of them during the systole with the open valve visible, the others during the diastole in which the closed valve is visible in almost all.
Since now we focus on the 3D reconstruction of the aorta vessel, we consider the image taken at the 80% of the cardiac cycle during the diastole. In fact, it is usual to reconstruct the aorta from a diastolic image because in this phase the blood is almost stationary in the artery. Hence the vessel is not subject to elastic deformation or lateral displacements and the resulting

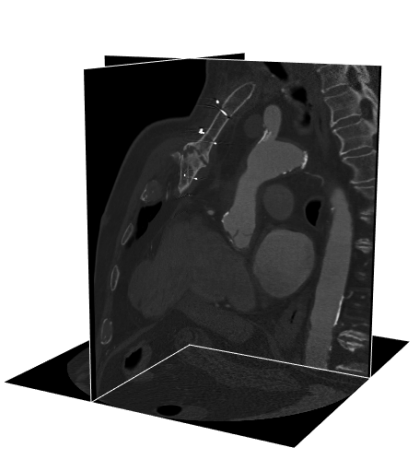
image is more clear, representing a more faithful description of the aorta. However the systolic images will become necessary for our purpose when we will reconstruct the open valve geometry.

2.7.1 Preprocessing

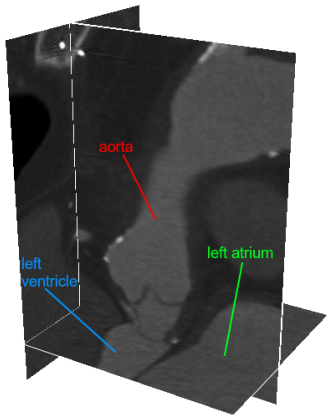
Crop. In fig. 2.1a is represented the original medical image in which are visible most of the bones and the organs of the chest. This image is taken at the 80% of the cardiac cycle during the diastolic phase.
As explained in the previous section, cropping near the region of interest is always the first step of pre-processing considering also that this is the best way to improve algorithm performances minimizing memory burden. In fig. 2.1b we show the cropped image near the aortic root, in which also the above part of left ventricle and of the left atrium are highlighted. Indeed, since we are interested in performing simulation including the aortic valve, our input surface has to be the last part of the left ventricle. Hence, we cannot crop the image above. As a consequence, we have to deal with the problem of the left atrium closeness, because we don't want to include the atrium in the output of the segmentation and on the other hand the atrium has the same gray level of the ventricle and aorta.

Select Range of Intensity and Median Filter. Before dealing with the atrium problem, we want first increase the contrast of the aorta with its background and decrease the salt & pepper noise. In fig. 2.1c the output of the select range of intensity method is shown. The original image takes values in the range $[-900, 3000]$. The lowest values represent the zones full of air like lungs and bronchi, while the greatest represent the bones. Selecting instead a range of $[-100, 1175]$ we increase the contrast of the aorta in the easiest possible way. At this point we can apply the median filter; as shown in fig. 2.1d the output image is a little less pimpled despite the original image was already of a good quality. The whitest part of the aorta notable in both the last two figures are calcifications due to the older age of the patient. Since we are interested in the lumen surface of the aorta, these calcifications

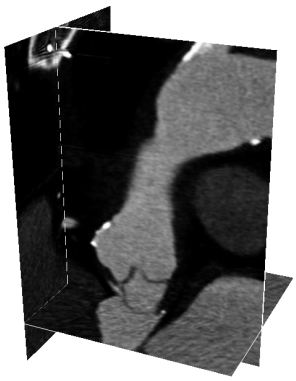
can be easily deleted from the image applying once again the select range of intensity method with a lower upper-bound and with the option of putting all the values outside the range at the minimum.



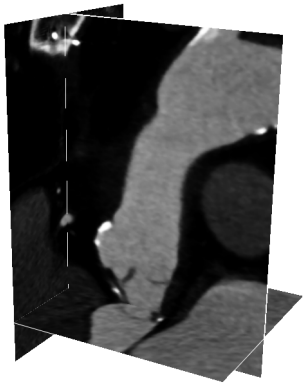
(a) the original 3D Image.



(b) the cropped image near the aortic root.



(c) the range of intensity output.



(d) the median filter output.

Figure 2.1: Illustration of the first steps of pre-processing.

Atrium Deletion Procedure. The algorithm implemented to delete atrium uses essentially the connected component extraction, the gaussian blur and simple algebraic operation on the images.

1. Step1: CC aorta extraction. We first extract the aorta using a big threshold in order to be sure that the first step of the connected component algorithm transforms the image in a black & white image where atrium and aorta are not connected. In this way, as shown in fig. 2.2a, the output of the connected component represents a sort of brutal segmentation of the aorta full of holes. Note that output of this method is a binary image made only of ones and zeros and this is useful in the third step.

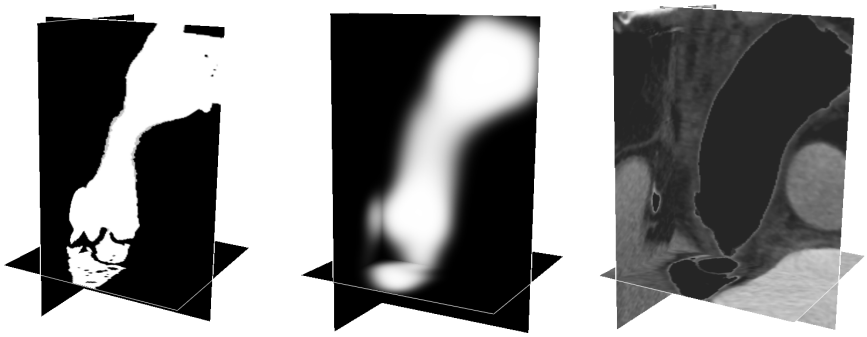
2. Step2: Gaussian Blur. The next step is enlarging the resulting binary image through the Gaussian Blur operation with $\sigma = 3.5$mm. The result is shown in fig. 2.2b. This is an effective way to fill the holes in the binary image. For doing this also some morphological operator could be used in a more efficient way (the resulting image in that case is still binary) [46], but they are not implemented in our library and the final purpose can be obtained also using our linear filtering method.

3. Step 3: Aorta deletion. At this point we combine through a multiplication the outputs of steps 1 and 2 (fig. 2.2a and fig. 2.2b)) and we generate the complementary image of the combined one: in this way we obtain an image that has zero or almost zero values near the aorta and one otherwise. This image is then multiplied for the original image in fig. 2.1d to obtain an image with zeros inside the aorta and the original values otherwise (fig. 2.2c): in this image the atrium has not got any neighbors with similar intensity values.

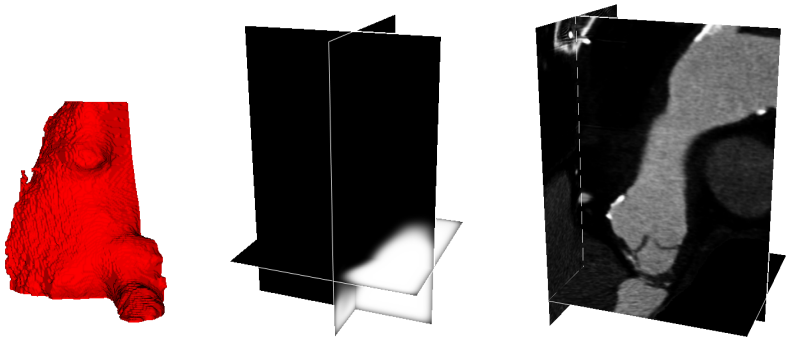
4. Step4: Atrium deletion. Now we can proceed in the same way of the first two steps (Step 4a and 4b) but starting from the output of the last step (fig. 2.2c). Note that for the atrium CC extraction now we can use a lower threshold than the one used in step 1 for the aorta: in this way we are sure to extract all the atrium region. Results are shown in fig. 2.2d and fig. 2.2e. Once again it is sufficient to multiply the complementary of this two images to the original image in figure fig. 2.1d to obtain as final result the same image but without the disturbing

proximity of the atrium (Step 4c).

We highlight that step 1 cannot be directly applied to the atrium because, if we use a low threshold aorta and atrium results connected while, if we use a high threshold the resulting CC of the atrium is not well defined and if we use this to subtract the atrium from the original image we still have some spurious pixels related to it. For this reason the described double-stage procedure is needed.



(a) Step 1. CC aorta extraction. (b) Step 2. Gaussian Blur. (c) Step 3. Aorta deletion.



(d) Step 4a. CC atrium extraction. (e) Step 4b. Gaussian Blur. (f) Step 4c. Final atrium deletion.

Figure 2.2: Algorithm to delete atrium illustrated with partial steps.

The image in fig. 2.2f is now ready to be segmented. We finally underline that all the operations described in this subsection could be performed in an interactive way using the executable of our library or in a more automatic

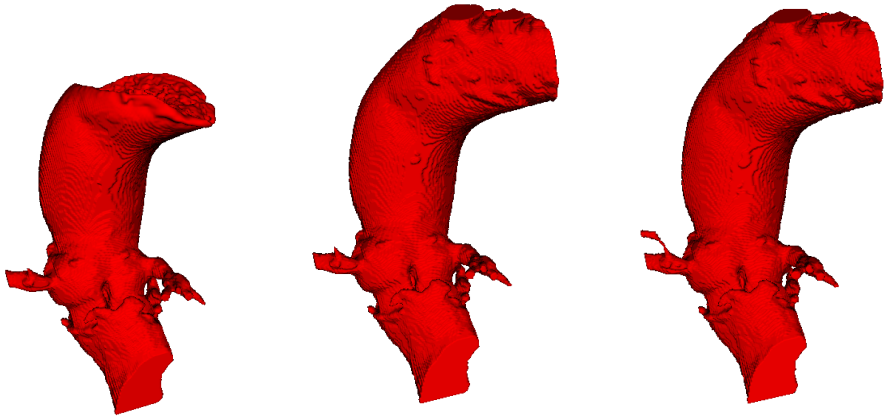
way writing directly the C++ code to use each method. In our case, after the setting of the correct parameters of the connected component extraction, we develop a code that automatically produces in few seconds an output similar to that one shown in fig. 2.2f also when executed in the images at the other percentages of the cardiac cycle.

2.7.2 Segmentation

After this kind of preprocessing the segmentation algorithm is fast and we can obtain a good result in few iterations as shown in fig. 2.3. The time of computation is less than 30 second using a quad-core laptop and the open-mp parallelization implemented in the library. To obtain the results shown, we started from a little cube inside aortic root as initial contour and we used parameters shown in table 2.1. They are the default values for our algorithm and they allows to segment images only in the case of an input image with a well distinguished region of interest as it is the output of our pre-processing procedure. Furthermore, in this case, it is not necessary to set the extraction of the connected component during the execution of the algorithm because there is not any risk of merging with other vessels. Instead, in case of a medical image in which is more difficult to delete near vessels, the best strategy to adopt is to slow down the evolution of the contour varying some parameters (especially increasing λ) and inserting the connected component extraction every few iterations. This means that the algorithm slows down and needs more iterations to converge but is still able to obtain good results also with more inhomogeneous or noisy images. For more details about this strategy and, in general, about our suggested settings of the parameters we refer to the report [26].

We conclude this chapter underlining that the output of the segmentation must be processed: first of all the α-level of the algorithm solution which represents the lumen interface must be extracted (see for example fig. 2.3) and a triangular surface mesh must be generated. Secondly, different post-processing steps on the resulting triangular surface are needed before the tetrahedral mesh for the numerical simulations can be generated. We will

see these steps in chapter 4 where we will focus also on the patient-specific valve reconstruction.



(a) 3^{rd} iteration. (b) 6^{th} iteration. (c) 12^{th} iteration.

Figure 2.3: Aorta surface evolution during segmentation algorithm.

σ	λ	λ_1	λ_2	ε	β	dt	ls_steps	a_0	b_0
5	10^{-3}	10^{-5}	10^{-5}	10^{-3}	100	1	5	0	1

Table 2.1: Algorithm's parameters adopted to obtain result in fig. 2.3.

Chapter 3

Moving Aortic Valve Model

Before showing in chapter 4 the method used to generate the patient-specific aortic valve, we dedicate this chapter to show the mathematical model adopted to include the leaflets of the valve with their movement in the fluid-dynamics equations. In this way also the final output of the algorithm for the valve reconstruction shown in section 4.4 will appear more clear to the reader.
In section 3.1 we start from the classical and weak formulations of the Navier-Stokes equation showing how to modify both of them to include inside the domain the immersed boundary surface that in our case represents the valve. Then, in section 3.2 we describe the level sets method used to model an open surface inside a three-dimensional domain and in section 3.3 we show how to include it in the weak formulation using a smooth Dirac function that allows to describe the valve surface directly in the three-dimensional domain. Finally, section 3.4 is dedicated to the detailed description of the reduced valve model adopted to describe in time the angular position of the leaflets [33].
Details of numerical implementation are shown instead in chapter 5 before the final results. We limit now to underline the importance for our method in the description of an immersed surface directly in a three dimensional space using the combination of the level set and the smooth Dirac function. Indeed, this avoid the problem of building a mesh with immersed bi-dimensional element consistent with the valve surface. Furthermore, since we want to describe also the movement of the leaflets, with a mesh approach, the mesh

itself should be updated in time causing lots of implementation problems linked, for instance, to re-meshing in a consistent way at each time step and to the interpolation of the solution between different meshes.

3.1 Navier-Stokes Equation with Resistive Immersed Surface

We now start from the Navier-Stokes (NS) equations for a generic incompressible homogeneous Newtonian fluid showing how to model an immersed surface inside a generic regular domain and concluding the section setting everything to model the hemodynamics in the aorta.

Given a fixed domain $\Omega^f \subset \mathbb{R}^d$ with $d = 2, 3$ the NS equations in their eulerian formulation write the following: find the fluid velocity $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t) : \Omega^f \times \mathbb{R}^+ \to \mathbb{R}^d$ and the fluid pressure $p = p(\boldsymbol{x}, t) : \Omega^f \times \mathbb{R}^+ \to \mathbb{R}$ such that

$$\begin{cases} \rho\left(\dfrac{\partial \boldsymbol{u}}{\partial t} - \dfrac{2\mu}{\rho}\nabla \cdot \mathbb{D}(\boldsymbol{u}) + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p\right) = \boldsymbol{f}, & \boldsymbol{x} \in \Omega^f,\ t > 0 \\ \nabla \cdot \boldsymbol{u} = 0, & \boldsymbol{x} \in \Omega^f,\ t > 0 \end{cases} \tag{3.1.1}$$

where ρ is the density of the fluid, μ is its viscosity and $\mathbb{D}$ is the strain rate tensor defined for Newtonian fluids as follows:

$$\mathbb{D}(\boldsymbol{u}) = \frac{\nabla \boldsymbol{u} + \nabla^T \boldsymbol{u}}{2}, \tag{3.1.2}$$

and linked to the Cauchy stress tensor with the relation:

$$\boldsymbol{\tau}(\boldsymbol{u}, p) = -p\mathbb{I} + 2\mu\mathbb{D} \tag{3.1.3}$$

where $\mathbb{I}$ is the identity tensor. For the sake of simplicity, we also assume that the boundary of the domain is divided in two regions $\partial\Omega^f = \partial\Omega^f_D \cup \partial\Omega^f_N$ where respectively Dirichlet and Neumann boundary conditions are assigned

$$\boldsymbol{u}(\boldsymbol{x},t) = \mathbf{g}(\boldsymbol{x},t) \qquad \forall \boldsymbol{x} \in \partial\Omega_D^f, \qquad t > 0 \tag{3.1.4a}$$

$$\boldsymbol{\tau} \cdot \boldsymbol{n}(\boldsymbol{x},t) = \mathbf{h}(\boldsymbol{x},t) \qquad \forall \boldsymbol{x} \in \partial\Omega_N^f, \qquad t > 0 \tag{3.1.4b}$$

and we close the problem with the initial condition

$$\boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0(\boldsymbol{x}), \ \forall \boldsymbol{x} \in \Omega^f, \tag{3.1.5}$$

where $\mathbf{g}$, $\mathbf{h}$ and $\boldsymbol{u}_0$ are known functions.

The Resistive Immersed Surface (RIS) method [27, 8] is based on the idea to model another surface $\Gamma \subset \mathbb{R}^{d-1}$ inside the domain Ω^f without considering it as a boundary of the fluid domain. Hence, the resulting domain is $\Omega = \Omega^f \cup \Gamma$ and the purpose is obtained adding at (3.1.1) a dissipative surface term to the conservation of momentum equation as follows:

$$\begin{cases} \rho\left(\dfrac{\partial \boldsymbol{u}}{\partial t} - \dfrac{2\mu}{\rho}\nabla \cdot \mathbb{D}(\boldsymbol{u}) + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p\right) + \boldsymbol{R}_\Gamma \ \boldsymbol{u} \ \delta_\Gamma = \boldsymbol{f}, & \boldsymbol{x} \in \Omega, \ t > 0 \\ \nabla \cdot \boldsymbol{u} = 0, & \boldsymbol{x} \in \Omega, \ t > 0 \end{cases} \tag{3.1.6}$$

$\boldsymbol{R}_\Gamma$ is a symmetric and positive tensor representing the dissipation due to the immersed surface and δ_Γ is the Dirac measure on the immersed surface that means:

$$\langle \boldsymbol{R}_\Gamma \ \boldsymbol{u} \ \delta_\Gamma, \boldsymbol{v} \rangle = \int_\Gamma \boldsymbol{R}_\Gamma \ \boldsymbol{u} \cdot \boldsymbol{v} \ d\Gamma, \quad \forall \boldsymbol{v} \in [H^1(\Omega)]^d. \tag{3.1.7}$$

Setting in the correct way the tensor $\boldsymbol{R}_\Gamma$ allows to study different problems, especially in medical application. For instance, in [27] the RIS method is used to model a medical device, called stent, consisting on an immersed porous surface designed for the treatment of cerebral aneurysm. Hence, in this case the tensor $\boldsymbol{R}_\Gamma$ needs to be modelled in order to allow a few of flow passage through the porous surface.

Instead, in our application the tensor has to model a surface not allowing flow passage and this is obtained replacing the tensor with a positive scalar

resistance R_Γ and setting its value with an high number.
RIS is used to model aortic valve in this way also in [8], but without modelling the valve movement. In [8] both the open and closed valve geometry are included in the domain and each surface is associated with a different resistance value during systole and diastole: closed valve surface switches from zero to an high-value when the valve closes and vice versa when the valve opens whilst open valve surface has the opposite behavior.
In our work we adopt a different approach: we include only an immersed surface in the domain, localizing it with a level set function and moving it using a reduced zero-dimensional model for the valvular angle. The difference between our work and [8] will become more clear at the end of this chapter after having discussed in detail all these aspects of our method.

We now focus on the weak formulation of equations (3.1.6). Let $\boldsymbol{v} \in V_0 : \Omega \to \mathbb{R}^3$ and $q \in Q : \Omega \to \mathbb{R}$ be the test functions for velocity and pressure, where the three functional space V, V_0 and Q are set as the following Sobolev space:

$$V = [H^1(\Omega)]^3, \qquad V_0 = [H^1_{\Gamma_D}(\Omega)]^3 = \{\boldsymbol{v} \in V : \boldsymbol{v}|_{\Gamma_D} = 0\}, \qquad Q = L^2(\Omega). \tag{3.1.8}$$

Thus applying the Gauss-Green Theorem and integrating by parts we obtain the following:

3.1.1. **Weak Formulation.** Find $\boldsymbol{u} \in L^2(\mathbb{R}^+; V) \cap C^0(\mathbb{R}^+; [L^2(\Omega)]^3)$, $p \in L^2(\mathbb{R}^+; Q)$, $\boldsymbol{u}|_{\Gamma_D} = \mathbf{g}$ such that:

$$\begin{cases} \left(\rho \dfrac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{v}\right) + a(\boldsymbol{u}, \boldsymbol{v}) + c(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) = F(\boldsymbol{v}), & \forall \boldsymbol{v} \in V_0 \\ b(\boldsymbol{u}, q) = 0, & \forall q \in Q \end{cases} \tag{3.1.9}$$

where we denote with $(\cdot, \cdot)$ the scalar product in $L^2(\Omega)$ and we define the

following bilinear forms and linear functional:

$$\begin{aligned}
a : V \times V \to \mathbb{R}, \qquad & a(\boldsymbol{u}, \boldsymbol{v}) = 2\mu \int_\Omega \mathbb{D}(\boldsymbol{u}) : \mathbb{D}(\boldsymbol{v})\ d\boldsymbol{x} + \int_\Gamma R_\Gamma\ \boldsymbol{u} \cdot \boldsymbol{v}\ d\gamma \\
b : V \times Q \to \mathbb{R}, \qquad & b(\boldsymbol{v}, p) = -\rho \int_\Omega p \nabla \cdot \boldsymbol{v}\ d\boldsymbol{x} \\
c : V \times V \times V \to \mathbb{R}, \quad & c(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \big((\boldsymbol{w} \cdot \nabla)\boldsymbol{u}\big) \cdot \boldsymbol{v}\ d\boldsymbol{x} \\
F : V \to \mathbb{R}, \qquad & F(\boldsymbol{v}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v}\ d\boldsymbol{x} + \int_{\Gamma_N} \mathbf{h} \cdot \boldsymbol{v}\ d\gamma.
\end{aligned} \tag{3.1.10}$$

Note that, regarding the resistive term, we simply have used the previous definition (3.1.7). Hence, in a variational framework, the dissipative resistance R_Γ can be interpreted as a penalization parameter that enforce the condition $\boldsymbol{u} = \mathbf{0}$ on the surface Γ. Furthermore, thanks to the positivity of the resistance R_Γ, the bilinear form $a(\boldsymbol{u}, \boldsymbol{v})$ is still coercive. Thus, this new term doesn't have negative effects on the usual analysis of the NS weak solution.
Since with this choice of functional space, $b(\boldsymbol{v}, p)$ satisfies the inf-sup condition stating that $\exists \beta > 0$ such that

$$\inf_{q \in Q} \sup_{\boldsymbol{v} \in V} \frac{b(\boldsymbol{v}, q)}{\|\nabla \boldsymbol{v}\|\ \|q\|} \geq \beta, \tag{3.1.11}$$

theoretical results prove the existence and the uniqueness of solutions for this problem with $d = 3$. For an in-depth analysis of this topic see [43, 45, 44, 28].

We conclude this section underlining the choice of modelling blood as a Newtonian fluid, despite blood is in general a non-Newtonian fluid for its heterogeneous composition. Indeed, it is made of blood cells (erythrocytes, leukocytes, thrombocytes) for 45% by volume, suspended in blood plasma which is composed mostly by water (92% by volume). Thus, blood rheology greatly depends on erythrocytes and their aggregation and elastic properties. Hence, especially in small vessels where erythrocytes and the diameter of the vessel are of the same order of magnitude, blood can not be seen as a continuum and the viscoelastic behaviour can not be neglected. Instead,

considering blood a Newtonian fluid is a common choice in large arteries like aorta. More details about blood rheology can be found in [9].

In (3.1.6) and (3.1.9) we set the viscosity μ equal to 0.0035 $g/(mm\ s)$ and the density equal to 0.001 g/mm^3.
The domain Ω matches the patient-specific aorta segmented in section 2.7 after the post-processing. As we will see in section 4.1, the boundary domain $\partial\Omega$ will be divided into three regions: the wall of the vessel $\partial\Omega_{wall}$, the inlet surface $\partial\Omega_{in}$ and the outlet surface $\partial\Omega_{out}$. Since we approximate the aorta as a rigid vessel without considering its elasticity and deformation, we impose homogeneous Dirichlet condition on $\partial\Omega_{wall}$; in the outlet and in the inlet surfaces we impose physiological pressures using the Neumann condition (3.1.4b); in some simulation we impose instead a physiological flow rate in the inlet during the systole using Dirichlet boundary condition for the velocity (3.1.4a). In section 5.3 we will describe in a more accurate way the imposed boundary conditions.
Concerning the resistance R_Γ a value of about 10^5 $g/(mm^2\ s)$ it has proven to be sufficient to avoid flow passage during the diastole, while during the systole with the open valve we can decrease this value of an order of magnitude.

3.2 Level-Sets Method to describe Open Surfaces

Before seeing the Galerkin discretization and its Finite Element (FE) formulation in section 5.1, we now see how to describe the valve surface using level sets functions.

A level set of a real-valued function of n real variables f is a set of the form:

$$L_c(f) = \{(x_1, \cdots, x_n) \mid f(x_1, \cdots, x_n) = c\}, \quad f : \mathbb{R}^n \to \mathbb{R}, \tag{3.2.1}$$

that is a set where the function takes on a given constant value c. When the number of variables is two, a level set is generically a curve, called a level curve, contour line, or isoline. When $n = 3$, a level set is called a level

surface or isosurface. Hence a level curve and a level surface are respectively the set of all real-valued roots of an equation in two or in three variables. If the function f is defined only to deal with its level sets, it is generally called level set function.
For our purpose, we want that the level set function is also a signed distance function (the reason will became clearer in section 3.3). This means that, giving a domain $\Theta \subset \mathbb{R}^n$, the function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as follows:

$$\begin{aligned} f(x) &= \begin{cases} d(x, \Theta^c) & \text{if } x \in \Theta \\ -d(x, \Theta) & \text{if } x \in \Theta^c \end{cases} \\ d(x, \Theta) &= \inf_{y \in \Theta} d(x, y) \end{aligned} \tag{3.2.2}$$

Hence, a distance function is a function that in each point of its domain takes the value of the distance of that point to the boundary of another immersed domain Θ. The boundary of Θ coincides with the zero level set of the distance function. A visual example in case of $\Theta \subset \mathbb{R}^2$ is shown in fig. 3.1.

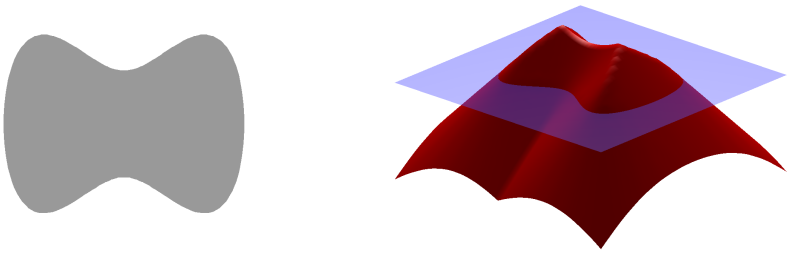


Figure 3.1: The domain Θ with its signed distance function.

Concerning our application, the domain of the distance function is the three-dimensional domain Ω represented by the aorta volume and we want to trace an immersed domain Θ such that $\partial\Theta = \Gamma$ where Γ represent the aortic valve surface.
We deal with another problem: the valve is an open surface while a distance function can only represent the distance between points in space and a closed

curve (2D) or surface (3D) matching the boundary of an immersed domain. To avoid this problem, the general strategy is to define a first level set function f to represent a closed surface such that part of it coincides with the interested open surface, and a second level set function g to cut the first one in order to describe the desired open surface as the combination of the two functions. In a more formal way, we represent the open surface Γ such that:

$$\Gamma \;=\; \{\boldsymbol{x} \in \Omega : f(\boldsymbol{x}) = 0 \;\wedge\; g(\boldsymbol{x}) \leq 0\}\,. \tag{3.2.3}$$

A visual example of the bi-dimensional case is shown in fig. 3.2.

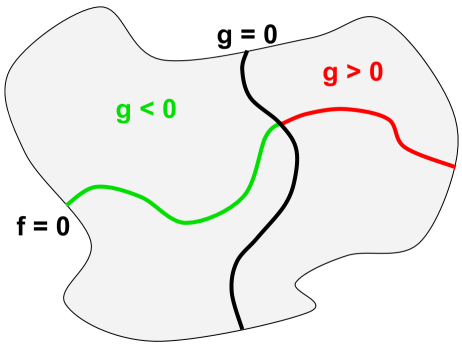


Figure 3.2: Representing an open curve (in green) in a 2D domain using two level set functions f and g.

Calling $\varphi : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$ the first level set function partially coincident with the valve surface in its zero level and $\psi : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$ the second one used to cut the extra surface, exploiting (3.2.3) we can rewrite the bilinear form a in (3.1.10) of the variation formulation (3.1.9) using the following definition:

$$\int_\Gamma R_\Gamma\; \boldsymbol{u}\cdot\boldsymbol{v}\; d\gamma \;=\; \int_{\varphi=0} \big(1-\mathcal{H}(\psi)\big)\; R_\Gamma\; \boldsymbol{u}\cdot\boldsymbol{v}\; d\gamma, \tag{3.2.4}$$

where $\mathcal{H}$ is the Heaviside function used to select only the domain where the

function ψ is negative i.e. :

$$\mathcal{H}(\psi) \;=\; \begin{cases} 1 & \text{if } \psi(x) > 0 \\ 0 & \text{if } \psi(x) <= 0 \end{cases}$$

Exploiting the Dirac measure definition (3.1.7) this means:

$$\int_{\varphi=0} \big(1-\mathcal{H}(\psi)\big)\; R_\Gamma\; \boldsymbol{u}\cdot\boldsymbol{v}\; d\gamma \;=\; \big(1-\mathcal{H}(\psi)\big)\; \langle R_\Gamma\; \boldsymbol{u}\; \delta(\varphi),\; \boldsymbol{v}\rangle \qquad (3.2.5)$$

Despite at this moment this method could seem only a formal way to describe the valve surface Γ, in the next section the advantages of this description of an open surface will appear clearer, especially from a numerical point of view.

3.3 Smooth Dirac Function to Include Level Sets

Next step in our method is to replace the Dirac function in (3.2.5) with a smooth approximation of it as follow:

$$\begin{aligned} \delta_\varepsilon(\varphi) \;&=\; \begin{cases} \big(1+cos(\pi\varphi/\varepsilon)\big)/2\varepsilon, & \text{if } \; |\varphi| \le \varepsilon \\ 0, & \text{if } \; |\varphi| > \varepsilon \end{cases} \\ \text{such that } \int_{-\infty}^{+\infty} \delta_\varepsilon(\varphi)\; d\varphi \;&=\; \int_{-\varepsilon}^{+\varepsilon} \delta_\varepsilon(\varphi)\; d\varphi \;=\; 1. \end{aligned} \qquad (3.3.1)$$

This kind of approximation assures that the property of the Dirac function of measuring the surface Γ is still verified:

$$\text{measure}(\Gamma) \;=\; \int_\Omega \delta_\Gamma\; d\boldsymbol{x} \;=\; \int_\Omega \big(1-\mathcal{H}(\psi)\big)\; \delta_\varepsilon(\varphi)\; d\boldsymbol{x} \qquad (3.3.2)$$

Note that this is true if and only if the level set function φ is also a distance function. Indeed, only in this case we can be sure to integrate a non null $\delta_\varepsilon(\varphi)$ exactly in a radius of 2ε around the level $\varphi = 0$, which coincides with the support of δ_ε. Hence now it is clear the reason of why we have defined it in such way.

Replacing in (3.1.6) the Dirac function with the smooth one, once moved on the weak formulation we obtain a volume integral instead of the surface one on Γ. More precisely, we can replace in (3.1.10) the surface integral as follow:

$$\int_{\Gamma} R_{\Gamma}\ \boldsymbol{u}\cdot\boldsymbol{v}\ d\gamma\ \rightarrow\ \int_{\Omega}\ \big(1-\mathcal{H}(\psi)\big)\ \delta_{\varepsilon}(\varphi)\ R_{\Gamma}\ \boldsymbol{u}\cdot\boldsymbol{v}\ d\boldsymbol{x}. \tag{3.3.3}$$

Replacing the surface integral with a volume integral gives us many advantages from a numerical point of view. As we will see in section 5.1 this avoids the problem of building a mesh with immersed bi-dimensional finite elements consistent with the valve surface and that has to move at each time step with it. In fact, in our approach, we define analytically the level set functions at each time step and we use them to evaluate the leaflets position directly in the three-dimensional finite elements. Moreover, the analytical description of the level set functions comes naturally from our patient-specific valve surface reconstruction technique (see section 4.4) combined with the reduced model we use for the computation of the valvular angle (see section 3.4) as we will see later. The approximation we are making with this approach is linked to the replacement of a surface without thickness with a volume of 2ε width. Hence, one has to set ε as small as possible. As we will see in section 5.2 its value is linked to some numerical factors like the dimension of the used finite elements of the mesh and the quadrature rule adopted. We finally underline again the importance of using a distance function for φ, otherwise not only the equality (3.3.3) is not true, but we loose also the geometric sense of ε linked with the leaflets thickness causing a valve surface with a space varying thickness.

3.4 Reduced Angle Valve Model

We now focus on the model used to decide the position of the leaflets. This model has two aims: decide when the valve has to be fully closed or fully opened and decide the opening angle while valve is switching between the two positions. Since, as we will see in chapter 4, the open and closed position can be reconstructed directly from the medical images and each one is associated

with a level set equation, we will exploit the opening angle found at each time step to interpolate in a smart way the open and closed level set equations in order to reproduce a valve that correctly simulate the angle computed from the reduced model.

The model that we used refer to the work of Korakianitis and Shi [33]. This paper presents a new concentrated parameter model for cardiovascular dynamics that includes an innovative model of heart valve dynamics, which is embedded in the overall model of the four chambers of the heart and the systemic and pulmonary circulation loops. Since the model is a totally zero dimensional model, the authors demonstrate that their results in terms of evolution in time of flow-rate and pressure in each heart chambers and at the beginning of each blood vessel that starts from them agree well with results illustrated in cardiology textbooks. In our work we take inspiration from the valve model proposed by the authors, importing it to make three-dimensional simulation on the aortic root.

The first step is to describe basic pressure–flow relation in the aortic valve:

$$Q_{ao} \;=\; CQ_{ao} \cdot AR_{ao} \cdot \sqrt{|P_{lv} \;-\; P_{ao}|} \tag{3.4.1}$$

where we have indicated with P_{lv} the pressure in the left-ventricle, with P/Q_{ao} the pressure/flow-rate in the aortic root and with CQ_{ao} a flow coefficient characteristic of the aorta vessel. AR_{ao} (acronym of Area Resistance) is the valve opening coefficient, linked to the percentage of overture of the orifice area and consequently linked to the angle position of the leaflets.
If an on/off model for the valve is considered (like in the already cited work of Astorino et al. [8]), the AR_{ao} is usually described simply switching between 0 and 1 depending on which side of valve has a higher pressure:

$$AR_{ao} \;=\; \begin{cases} 1, & \text{if } P_{lv} \geq P_{ao} \\ 0, & \text{if } P_{lv} < P_{ao} \end{cases} \tag{3.4.2}$$

Instead, in the new moving valve model presented in [33] the valve opening

coefficient is decided by the angular position of the valve leaflets:

$$AR_{ao} = \frac{(1 - cos\vartheta)^2}{(1 - cos\vartheta_{max})^2} \tag{3.4.3}$$

while the leaflet angular position ϑ is computed by considering the various factors that affect the leaflet motion. More in details, as illustrated in fig. 3.3, these include the blood-flow effects of pressure difference across the valve, the frictional effect from neighbouring tissue resistance, the dynamic motion effect of the blood acting on the leaflet, the action of the vortex downstream of the valve, the shear stress on the leaflet. Hence the valve

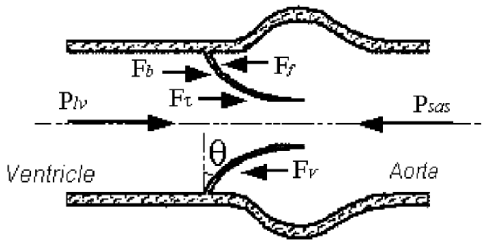


Figure 3.3: Illustration of typical forces acting on a heart valve leaflets, imported from [33]-fig.3.

motion is governed by the following differential equation:

$$I_{ao}\frac{d^2\vartheta}{dt^2} = F_{pr} - F_{fr} + F_{bm} - F_{vo} + F_{ss}, \tag{3.4.4}$$

where I_{ao} indicates the inertial moment of rotating and we define the various

forces as follows [33]:

$$F_{pr} = k_{p,ao}\,(P_{lv} - P_{ao})\,cos(\vartheta) \tag{3.4.5a}$$

$$F_{fr} = k_{f,ao}\,\frac{d\vartheta}{dt} \tag{3.4.5b}$$

$$F_{bm} = k_{b,ao}\,Q_{ao}\,cos(\vartheta) \tag{3.4.5c}$$

$$F_{ss} \approx 0 \tag{3.4.5d}$$

$$F_{vo} = \begin{cases} k_{v,ao}\,Q_{ao}\,sin(2\vartheta), & \text{if } Q_{ao} \geq 0 \\ 0, & \text{if } Q_{ao} < 0 \end{cases} \tag{3.4.5e}$$

More in detail the following physiological phenomena are linked to these forces:

- The pressure effect F_{pr} is considerer proportional to the normal projection on the valve leaflet surfaces of the pressure difference (considering $P_{lv} - P_{ao}$ as a vector with the direction illustrated in fig. 3.3).
- The frictional effect F_{fr} due to tissue resistance at the valve root is assumed to be proportional to the leaflet angular velocity.
- The blood motion effect F_{bm} is assumed to be proportional to the flow rate projected once again in the normal direction of the leaflet surface.
- The shear stress force F_{ss}, according for instance to [47], is very small and can be neglected.
- The vortex effect F_{vo} is still ongoing investigations. In some works, such as [51, 10], is discussed the importance of the vortex effect underlining that neglecting it will cause overestimation of the reverse flow during valve closure. The vortex effect should be modelled by the local vorticity distribution, which is possible with the distributed parameter studies using 2D or 3D CFD, and focusing on local flow features. However numerical simulation made in the past (see for instance [47]) failed to provide a clear understanding of this effect on valve motion. Thus, following [33], we model the vortex effect basing on experimental

and clinical observations proofing that the vortex intensity is closely related to the fluid flow rate across the valve and the leaflet angular position. More precisely, in early systole, with the contraction of the left ventricle the forward flow across the aortic valve is ever-increasing as jet flow, the valve is in the opening process, and there are vortices developing in the aortic sinuses. This trend lasts until valve leaflets open to, approximately, the valve half-open position. After this, the aortic flow is still increasing and the valve is still in the opening process, but the motion of valve leaflets reduces the allowable space for the vortex in the aortic sinuses, so that after the valve half-open position vortex intensity is decreasing. This situation continues until valve leaflets reach the fully open position, and the forward aortic flow becomes weak, in late systole. In late systole and early diastole, the aortic flow changes from weak forward flow to weak backward flow. When the backward aortic flow develops, vortices become merged into the strong backward flow and lose the effect, thus no vortex effect needs to be considered then. Based on this understanding, the term F_{vo} is constructed in the form of the product of the flow rate and the effect of the leaflet angular position. It is understood that the bigger the flow rate in the valve, the stronger the vortex intensity near the valve. Also, it is assumed that vortex intensity reaches the maximum value when the leaflet is at about 45 opening angle. Below 45, the smaller the valve opening angle, the more likely for the flow to pass the valve as jet flow. Similarly, above 45, the bigger the valve opening angle, the more likely the leaflets will cover the valve root area and restrict the vortex formation.

Defining $K = k/I_{ao}$ for each force, we obtain the final equation to find the

angle ϑ of the aortic valve:

$$\frac{d^2\vartheta}{dt^2} = \begin{cases} k_{p,ao}\ (P_{lv} - P_{ao})\ cos(\vartheta)\ -\ k_{f,ao}\ \frac{d\vartheta}{dt}\ + \\ \ + k_{b,ao}\ Q_{ao}\ cos(\vartheta)\ -\ k_{v,ao}\ Q_{ao}\ sin(2\vartheta), & \text{if } Q_{ao} \geq 0 \\ \\ k_{p,ao}\ (P_{lv} - P_{ao})\ cos(\vartheta)\ -\ k_{f,ao}\ \frac{d\vartheta}{dt}\ + \\ \ + k_{b,ao}\ Q_{ao}\ cos(\vartheta), & \text{if } Q_{ao} < 0 \end{cases} \tag{3.4.6}$$

Since the aortic valve has got a fixed angle while is totally open or closed, a ϑ_{min} and a ϑ_{max} are defined too. This means that if the ϑ computed with (3.4.6) is greater than ϑ_{max}, it is forced to be equal to ϑ_{max} meaning that the opened position is reached and the leaflets have not to move for a while; on the contrary (if it is less than ϑ_{min}) it is set equal to ϑ_{min} till the valve remains in the closed position.

We finally underline that (3.4.6) can be generalised at the other cardiac valves simply setting in the correct way each parameters, since the physic phenomena that influence the movement are the same [33]. This makes the whole work of this thesis easily generalisable to the other cardiac valves or usable for a more complex numerical simulation that includes all the four chambers of the heart. In table 3.1 are summarized the parameters used in case of the aortic valve, taken from [33].

CQ_{ao}	$K_{p,ao}$	$K_{f,ao}$	$K_{b,ao}$	$K_{v,ao}$	ϑ_{min}	ϑ_{max}
350 $\frac{ml}{s\ mmHg^{0.5}}$	5500 $\frac{rad}{s^2\ mmHg}$	50 s^{-1}	2 $\frac{rad}{s\ m}$	7 $\frac{rad}{s\ m}$	5°	75°

Table 3.1: Valve angle model parameters from [33].

One of the most important improvement due to this detailed description of valve dynamics is that it helps to describe the regurgitant flow in the valve in a more physiological way. Indeed, using an on/off model the regurgitant flow is usually totally omitted (because the valve closed when $P_{lv} < P_{ao}$ or when Q_{ao} becomes null) while using a less detailed model that ignores the

vortex effect is generally overestimated.

We conclude this chapter underlining that in our application the input parameters P_{lv}, P_{ao} and Q_{ao} of the valve equation are directly computed from the numerical simulation in the three-dimensional domain, ϑ is instead used to interpolate the open and closed level set functions representing the open and closed valve. Details about the numerical implementation will be shown in chapter 5.

Chapter 4

Patient-Specific Aortic Valve

In this chapter we focus on the mesh generation and on the patient-specific valve reconstruction.
We start, in section 4.1, from the post-processing of the output of the segmentation algorithm necessary to make the surface ready to generate the mesh.
Then we describe the algorithm to reconstruct the valve: in section 4.2 we focus on the rotation to make the images and the surface orthogonal to the valvular plane, in section 4.3 we describe the method adopted to take the control points of the valve leaflets from the medical images, in section 4.4 we interpolate the control points to finally generate the level set equations adopting different strategies for the open and the closed position.
We conclude the chapter in section 4.5 describing the method used to build a mesh more dense near the valve.
All steps of the algorithm are implemented as Python scripts using the Visualization Toolkit (VTK) library [4] and the Vascular Modeling Toolkit (vmtk) [3].

4.1 Post-Processing on the Segmented Surface

We have concluded chapter 2 with a triangular surface mesh generated from the output of the segmentation algorithm (fig. 4.1a). This is made through the marching cubes algorithm [36] using the corresponding vmtk script as follows:

```
vmtkmarchingcubes -ifile inputimage.mhd -l alpha
    -connectivity 1 -ofile outputsurface.vtp
```

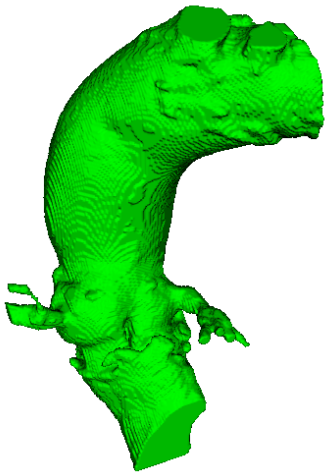
where the *-connectivity* option is used to output only the largest connected region of the isosurface. In this way we can exclude the other isosurfaces generated by the segmentation algorithm.
The surface generated in this way needs some post processing steps as illustrated in fig. 4.1, before it can be used to generate the computational mesh:

- First of all we cut off the coronaries using the vmtk script *vmtkmeshadjust* implemented by Ph.D. Elena Faggiano. This script allows to interact with the surface drawing a closed curve inside. Then it cuts the surface capping the generated hole using a thin plate sline interpolator. After the capping step the script performs a remeshing of the whole surface that makes it also a little smoother. The output of this process is shown in fig. 4.1b.

- Then, we open the outlet and the inlet surface fig. 4.1c using vmtk script *vmtksurfaceclipper*. This script allows user interaction using a deformable and movable cube in order to decide where to open the surface with a clip.

- We finally smooth the surface.
  Firstly, this is made using once again the script *vmtkmeshadjust* to cut the bumps of the ventricle in the bottom part of the surface.
  Secondly, it is used the script *vmtksurfacesmoothing* to obtain the result shown in fig. 4.1d.

The surface is now ready to build a mesh. Before seeing how to generate it in section 4.5, we skip to the algorithm we implemented to reconstruct the patient-specific open and closed aortic valve surfaces. Indeed, we need these surfaces to generate an adaptive finer mesh in the valvular zone.



(a) output of segmentation algorithm.

(b) cut the coronaries.

(c) open inlet and outlet boundary.

(d) final smoothing of the surface.

Figure 4.1: Illustration of the post-preprocessing steps on the surface.

4.2 Valvular Plane Highlighting

The first step of the algorithm used to reconstruct the valve consists of an opportune rotation in order to make the valvular plane more visible. In radiology the valvular plane is the plane orthogonal to the valve surface in which it is more comfortable to see the leaflets. In general coordinates of a medical image have not an axis orthogonal to this plane. We want to avoid this problem, implementing an algorithm that can rotate the reference system in a consistent way with the valvular plane exploiting information about the aorta surface. Once this result is obtained we can simply scrolling a plane orthogonal to the new z-axis in order to clearly see the three leaflets at each level z.

Showing the medical image used to segment the aorta surface as an example, we proceed as follows:

- We first compute the centerline of the aorta (the line that intuitively pass in the middle of the vessel) [6, 5]. Looking at fig. 4.2a we note that the centerline (in red) is not parallel to the z-axis near the valvular zone. With these coordinates the leaflets are less recognizable. On the contrary, manually rotating the plane in order to be orthogonal to the centerline (fig. 4.2b), we see perfectly the valvular plane. Hence, we want that the new z-axis takes the direction of the centerline at the level of the valvular plane.

- At this point, we simply ask to the user to select a point where the leaflets cross each-other (the red-point of fig. 4.2b). We exploit this point to find the nearest point on the centerline and to compute its direction. Then we find the correct rotation matrix needed to rotate coordinates and we apply it at the aorta surface, at the centerline itself and at all the images loaded by user. We also set the point selected by the user as the new offset of all the objects. Indeed, it can be useful to locate the centre of the valvular plane with the point $\boldsymbol{x} = (0, 0, 0)$ of the system of reference.
  In our example we consider two systolic images with the open valve

partially visible and one with the closed valve perfectly visible. The outputs of the procedure are shown in fig. 4.2c, 4.2d, 4.2e. The first two figures refer to the closed valve image. We notice the quality of the visibility of the three leaflets, both in the top part in which they are attached each other and in the bottom part. The one concerning the open valve is instead less clear. Indeed, we decide to load both the available images in systolic phase to use all the information we have about the open leaflets.

- Looking again at fig. 4.2d, we can notice another problem. The medical image is shifted horizontally in respect to the aorta surface. The reason is that we have generated the segmented surface from the image with the closed valve and a little movement between two different instants of the cardiac cycle is physiological. Since in our fluid simulation we instead consider, as usual, the aorta surface as rigid wall, we want to make the open-valve image more coherent with the surface. To obtain this result, we proceed approximately as before. We allow user to select two points in the image and we translate horizontally the image in the direction joining the two points computing again the corresponding transform matrix. Output of the translation is shown in fig. 4.2f.

We now see more in detail the used and implemented python script.
To generate the centerline we use the corresponding vmtk script:

```
vmtkcenterlines -ifile inputsurface.vtp -endpoints 1
    -ofile centerline.vtp
```

where the option *-endpoints* makes the centerline continue till the center of the open boundaries.
The rotation algorithm is instead implemented in a vmtk-compatible script that can be used as follow:

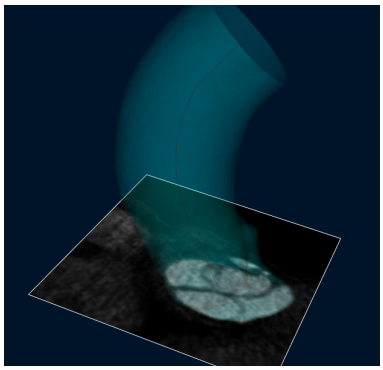
```
vmtkimagerotation -interactive 1 -axis z
    -ifile inputimage.mhd [-i2file inputimage2.mhd ...]
    -icenterlinefile inputcenterline.vtp
    -isurfacefile inputsurface.vtp
    -ofile outputimage.mhd [-o2file outputimage2.mhd ...]
```

```
    -ocenterlinefile outputcenterline.vtp
    -osurfacefile outputsurface.vtp
```

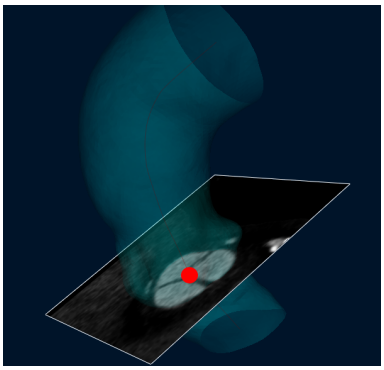
This interactive version, as described above, simply asks to the user the point where the leaflets cross each others and gives as output all the rotated objects. A not interactive way of the script is also allowed, taking as input parameters the direction of new z-axis and the new offset. Here is shown with the parameters used to rotate objects in fig. 4.2

```
vmtkimagerotation -interactive 0 -ifile ... -ofile ...
    -newzdirection -0.62 -0.002 0.79
    -newoffset 37.64 26.31 23.64
```

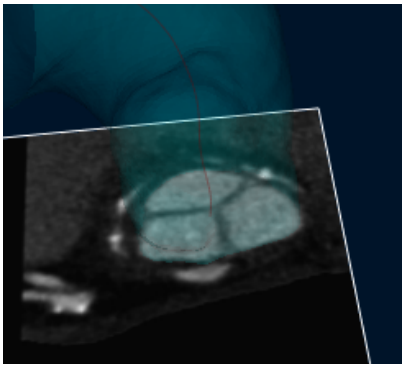
Concerning the horizontal translation, we have implemented a similar script called *vmtkimagetranslator* working with same inputs. The only difference consists on the selection of two points on the image instead of one, to obtain a vector usable for the translation (fig. 4.2e). In its not interactive version, we add only the parameter *-translationvector*.
We finally underline that inside the scripts, we largely use some *VTK*classes like *vtkPointLocator*, *vtkTransform*, *vtkTransformFilter*, *vtkImageReslice*.



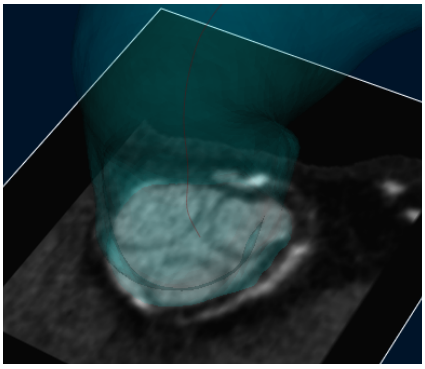
(a) plane orthogonal at z-axis before rotation.



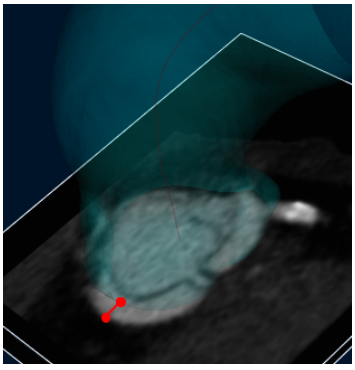
(b) selection of new origin interactively.



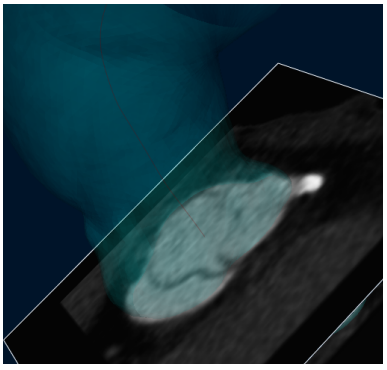
(c) plane orthogonal to new z-axis after rotation showing the top part of the closed valve.



(d) the same plane showing the bottom part of the medical image with the closed valve.



(e) open valve image not coherent with the surface after rotation.



(f) open valve image after the translation, coherent with the surface.

Figure 4.2: Illustration of the effect of the rotation and translation on the visibility and coherency of the valvular plane

4.3 Leaflets Control Points Picking

Once we have each images and surface correctly rotated and translated, we are in the best possible situation to start leaflets reconstruction.
As we have seen in section 3.2, our purpose is to define analitycally two level set functions φ and ψ. In particular, we want to reach this aim to describe both closed and open valve surfaces. The general idea of our algorithm consists in taking the greatest number of control points from the medical images available and interpolating them with a polynomial fitting.
We will talk more in detail about the interpolations in the next section, focusing now on the way of taking control-points from the image. We remind that the valve surface has to be described with the level set function φ and that the auxiliar level set function ψ is used only to cut the extra surface defined by the zero level of φ. Since is clearly not possible to define a unique polynomial to describe the whole valve surface, we take different sets of control-points in different areas.

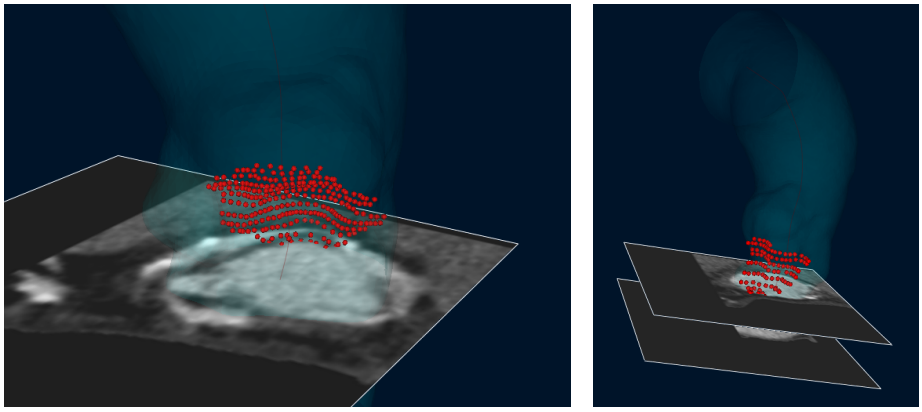


Figure 4.3: Taking control-points from the images in case of two different open leaflets

More in detail, concerning the level set function φ, we proceed as follows:

- **top closed valve**: the top part of the closed valve is characterized by the three leaflets stick each other. From a top view, as visible in fig. 4.2c, we see like three lines. We decide to take a different set of

control points for each of these lines. Looking at each set, it appears natural to interpolate each one with a plane.

- **bottom closed valve**: the bottom part is instead characterized by three separated leaflets with a circular form (see fig. 4.2d). These half circles end inside the aorta wall and tend to became smaller in size if we scroll the plane downward. Hence, they remind the typical level curves of a three-dimensional parabola. Once again we decide to take separate control-point sets for each leaflet.

- **open valve** Concerning the open valve, the form is different to be interpreted. Taking control-points in different sets for each leaflet we obtain a cloud of points as shown in figure fig. 4.3. We noticed that, using the coordinates of the image, a polynomial fitting failed to work good. Hence, we decided to change the coordinates for each control-point set, in order to obtain a reference system suitable for the polynomial interpolation. In particular, the new z-axis is chosen in order to cross the cloud of point in a direction parallel to the valvular plane. With new coordinates, we can describe the leaflets with a fifth degree polynomial.

Instead, concerning the level set ψ, we can take only a set of control points for the open valve and another for the closed one. Indeed, since all the leaflets end at about the same value of z, we can interpolate the control points all together defining a unique polynomial to cut φ in a direction almost parallel to the valvular plane.

The implementation of this part of the algorithm take inspiration from the vmtk script *vmtkimageseeder*. The main differences concern the visualization, the input and the output.
Instead of visualize the image with the classic three planes, we show only that one orthogonal to the z-axis because we are interested in a visualization parallel to the valvular plane.
We talk about the inputs and the outputs after showing an example of usage:

```
vmtkimagecontrolpointswriter -activerotation 1
```

```
-ifile inputimage.mhd [-i2file inputimage2.mhd ...]
[-isurfacefile inputsurface.vtp]
-ofile PhiRightOpen.dat
```

The first difference is linked to the fact that we allow more than one image in input, as shown in fig. 4.3. This allows, for instance, to take control points for the same leaflet from two different images. If the *-activerotation* flag is active when the control points are generated user can rotate them in new coordinates. This is done, like in the previous scripts, selecting two points in the image representing the new z direction. All the control points (both original and rotated ones) are saved in a text file readable via MatLab®.

4.4 Interpolation of the Level Set Equation

At this point of the algorithm we can simply use the function *polyfit* of MatLab to interpolate each set of control points and to generate each equation. We can finally put togheter all the analitycal equations using maximum and minimum operator to generate the equation of φ.

To better understand what is the output of the whole algorithm described in the last two sections, we now show as an example the complete equation of the level set φ concerning the leaflet of the righ-coronary sinus in both open and closed position.

$$\textbf{\textit{closed}} \begin{cases} \varphi_{rn}(x,y,z) &= z - c_{rn,00} - c_{rn,10}\cdot x - c_{rn,01}\cdot y \\ \varphi_{rl}(x,y,z) &= z - c_{rl,00} - c_{rl,10}\cdot x - c_{rl,01}\cdot y \\ \varphi_{r}(x,y,z) &= z - p_{r,00} - p_{r,10}\cdot x - p_{r,01}\cdot y - \\ & \quad\; p_{r,20}\cdot x^2 - p_{r,11}\cdot xy - p_{r,02}\cdot y^2 \\ \varphi_{right}(\boldsymbol{x}) &= min\left\{\varphi_{rn}(\boldsymbol{x}),\ \varphi_{rl}(\boldsymbol{x}),\ \varphi_{r}(\boldsymbol{x})\right\}. \end{cases} \tag{4.4.1}$$

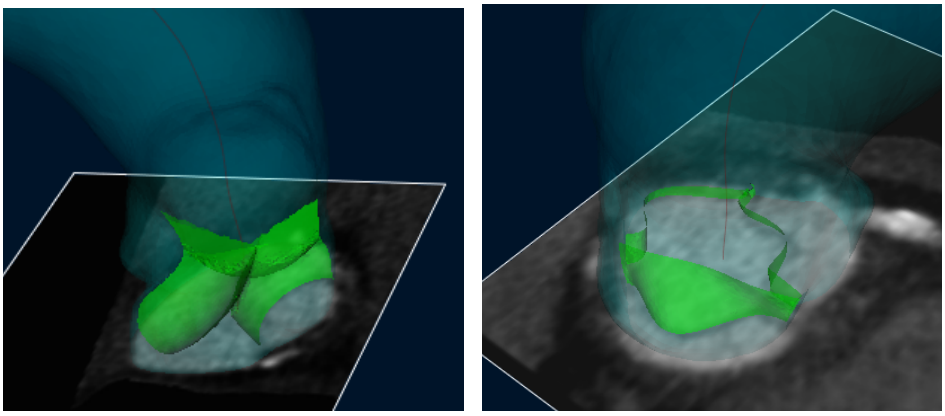
$$\textbf{\textit{open}} \begin{cases} X &= c_{11} \cdot x + c_{12} \cdot y + c_{13} \cdot z \\ Y &= c_{21} \cdot x + c_{22} \cdot y + c_{23} \cdot z \\ Z &= c_{31} \cdot x + c_{32} \cdot y + c_{33} \cdot z \\ \varphi_r(x,y,z) &= Z - \sum_{i,j=0\ldots5} p_{r,ij} \cdot X^i Y^j \\ \varphi_{right}(\boldsymbol{x}) &= min\left\{\varphi_r(\boldsymbol{x}),\ \varphi_{right,closed}(\boldsymbol{x})\right\}. \end{cases} \tag{4.4.2}$$

Since also for the other two leaflets we define the function negative inside the valve surface and positive otherwise, in order to describe the three leaflets together in a unique level set function φ, we can simply apply the maximum operator as follows:

$$\varphi(\boldsymbol{x}) \ = \ max\left\{\varphi_{right}(\boldsymbol{x}),\ \varphi_{left}(\boldsymbol{x}),\ \varphi_{non}(\boldsymbol{x})\right\}. \tag{4.4.3}$$

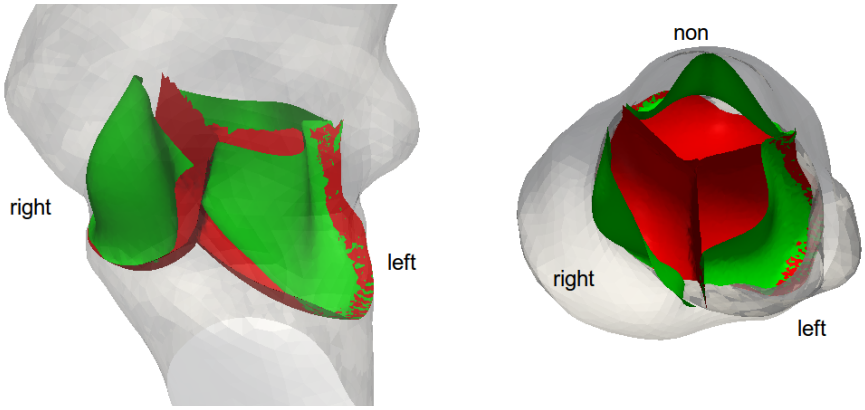
We finally underline that in this way the level set function is not a distance function. We will see in section 5.2 how to regolarize it in order to obtain the desired distance function introduced in section 3.2.
We now have the analytical expression of the closed and open valve. We will use the angle computed from the reduced model shown in section 3.4 to interpolate this two equations in order to generate the surface of the moving valve. Details are postponed to chapter 5.

We conclude this section showing the valve surfaces generated from this algorithm in fig. 4.4, where the quality of the output is evident.
The first two figures 4.4a and 4.4b show the reconstructed valve surface compared with the medical image from which the control points are taken. In both cases the reconstructed leaflets coincide well with the images.
In fig. 4.4c and 4.4d we show two different views of the open and closed valve together. We notice that in the two coronary sinuses the space between the leaflets and the lumen wall is clearly bigger then in the non-coronary one. This fact is linked to the blood flow toward the coronaries and will be clearer in chapter 5 after we will show our numerical results.



(a) closed valve with its image.

(b) open valvewith its image.



(c) open and closed valve lateral view.

(d) open and closed valve top view.

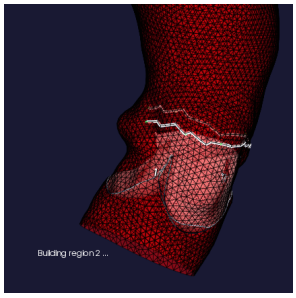
Figure 4.4: Output of the algorithm to reconstruct the valve.

4.5 Adaptive Mesh Generation

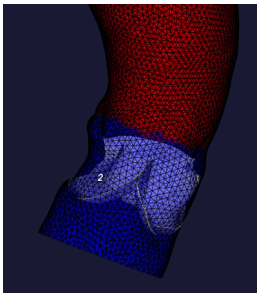
We can now finally generate the tetrahedral mesh proceeding as follows:

- We use an apposite script to tag the surface in different regions as shown in fig. 4.5a and fig. 4.5b. This script, called *vmtksurfaceregiontagger*, was written in collaboration with Ph.D. E. Faggiano and it used to produce a surface with different tags in each zone in which we desire tetrahedra of different dimensions.
- Exploiting the output of the previous script, we assigned a local tetrahedra dimension at each tag.
- We use these dimensions to generate the mesh through the script *vmtkmeshgenerator*.

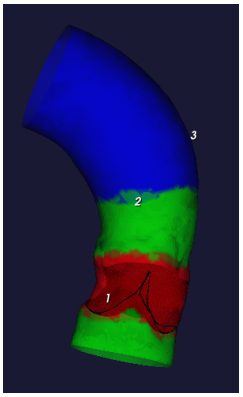
This procedure is thought to generate a mesh refined in the area near the valve for better modeling and capturing the leaflets movement.
As shown in fig. 4.5d and 4.5c, we choose to divide our mesh into three regions. We always set the smallest dimension of tetrahedra in region 1, the biggest in region 3 and an intermediate one in region 2. In particular, the mesh used for the results that we will show in section 5.4 is illustrated in fig. 4.5e.



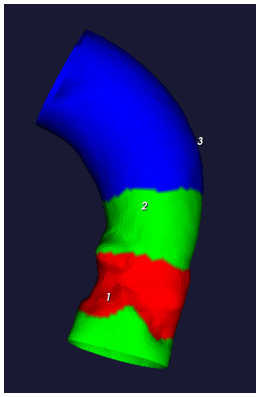
(a) drawing a region interactively.



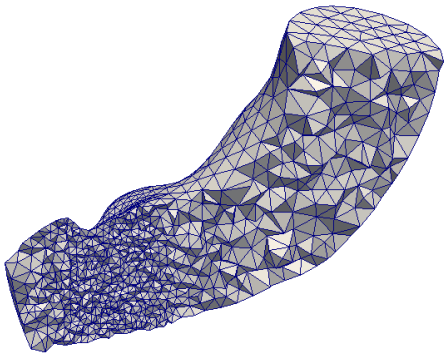
(b) generation of a tag.



(c) tagged surface with valve.



(d) tagged surface.



(e) mesh with elements respectively of 1.85, 3.5 and 5 mm of diameters in the three regions.

Figure 4.5: The procedure to tag the surface and generate the mesh.

Chapter 5

Numerical Implementation and Results

This chapter is dedicated to the numerical implementation and to the illustration of the results. Starting from the weak formulation derived in section 3.1, we formulate the Galerkin problem and its Finite Element (FE) approximation in section 5.1. Then, in section 5.2, we focus on some implementation details linked to the level-set equations like the method adopted to regularise them near the valve and its evaluation in the quadrature rule points. In section 5.3, we talk about the boundary condition imposed and the problems of stabilization that we have at the actual implementation of the code. Finally, we show the results obtained in section 5.4.
Before starting, we first underline that we use LifeV [1] to run our simulations. LifeV is a project started in 2002 developed by a joint collaboration between Politecnico di Milano (MOX department), École Polytechnique Fédérale de Lausanne (group CMCS), INRIA (project REO) and Emory University. More in detail, it is a finite element library written in C++ for the solution of PDEs released under LGPL license and with a strong focus in bio-medical applications.

5.1 Finite Element Formulation

We now see the discretization of the weak formulation 3.1.1, with the biliniar form a updated using the two level-sets ψ and φ following (3.3.3).
We use the couple of finite elements spaces $\mathbb{P}^1_{bubble}$-$\mathbb{P}^1$ for the velocity and pressure discretization. This choice is due to the necessity of using an

$inf - sup$ stable couple to solve our problem because otherwise we need to stabilize our discrete formulation with a specific stabilization method for our problem. Indeed, the already implemented stabilization methods in LifeV (like Interior Penalty (IP) Stabilization) [13] do not work with our problem probably because of the strong pressure gradient that we have across the valve interface that can not be correctly stabilized using methods based on constant coefficients.
More formally, let $\mathcal{T}_h$ be a tetrahedralization of our domain $\Omega \subset \mathbb{R}^3$ representing the aortic root such that $\Omega = \bigcup_{T\in\mathcal{T}_h} T$ and $h = \min(\mathrm{diam}(T))$. We introduce the spaces:

$$\begin{aligned} X_h^r &= \left\{ v_h \in C^0(\bar{\Omega}) : v_h|_T \in \mathbb{P}^r,\ \forall T \in \mathcal{T}_h \right\}, \\ \mathbb{B} &= \left\{ v_B \in V \cap C^0(\bar{\Omega}) : v_B|_T = c_T \cdot b_T,\ b_T|_{\partial T} = 0,\ c_T \in \mathbb{R},\ \forall T \in \mathcal{T}_h \right\}, \end{aligned} \tag{5.1.1}$$

where we denote with $\mathbb{P}^r$ the finite dimensional functional space of polynomial of degree r and with b_T a generic bubble function defined in each tetrahedra. Indicating with V and Q the ∞-dimensional space defined in (3.1.8), we can now set the finite element space $V_h = ([X_h^1]^3 \cap V) \oplus \mathbb{B}$ and $Q_h = X_h^1 \cap Q$. Hence, we can formulate the discretized problem in space as follows:

5.1.1. **Galerkin Formulation.** $\forall t > 0$, find $(\boldsymbol{u}_h(t), p_h(t)) \in V_h \times Q_h$ such that $\boldsymbol{u}_h(0) = \boldsymbol{u}_{h0}$, $\boldsymbol{u}_h = \mathbf{g}_h$ on Γ_D and

$$\begin{cases} \left(\rho \dfrac{\partial \boldsymbol{u}_h}{\partial t}, \boldsymbol{v}_h\right) + a(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) + b(\boldsymbol{v}_h, p_h) = F(\boldsymbol{v}_h), & \forall \boldsymbol{v}_h \in V_h \\ b(\boldsymbol{u}_h, q_h) = 0, & \forall q \in Q \end{cases} \tag{5.1.2}$$

where $\mathbf{g}_h$ and $\boldsymbol{u}_{h0}$ are the approximation on the Galerkin space V_h of the boundary and initial conditions respectively.

Introducing a base $\{\boldsymbol{\varphi}_j\}_{j=1}^{N_u}$ for the spaces V_h and a base $\{\psi_k\}_{k=1}^{N_p}$ for Q_h (with $N_u = \dim(V_h)$ and $N_p = \dim(Q_h)$), we can now obtain an algebraic

formulation. Writing $\boldsymbol{u}_h$ and p_h using the base functions

$$\boldsymbol{u}_h(\boldsymbol{x},t) = \sum_{j=1}^{N_u} u_j(t)\boldsymbol{\varphi}_j(\boldsymbol{x}), \quad p_h(\boldsymbol{x},t) = \sum_{k=1}^{N_p} p_k(t)\psi_k(\boldsymbol{x}) \tag{5.1.3}$$

and taking $\boldsymbol{v}_h = \boldsymbol{\varphi}_j$, $j = 1 \dots N_u$ and $q_h = \psi_k$, $k = 1 \dots N_q$ we obtain the following non linear system of ODEs:

$$\begin{cases} M\dfrac{d\mathbf{u}(t)}{dt} + A\mathbf{u}(t) + C(\mathbf{u}(t))\mathbf{u}(t) + B^T\mathbf{p}(t) = \mathbf{F}(t) \\ B\mathbf{u}(t) = 0 \end{cases} \tag{5.1.4}$$

where $\mathbf{u}(t) = [u_1, ..., u_{N_u}]^T$, $\mathbf{p}(t) = [p_1, ..., p_{N_p}]^T$, $\mathbf{F}(t) = [F(\boldsymbol{\varphi}_1), ..., F(\boldsymbol{\varphi}_{N_u})]^T$ and each matrix is defined using the definition (3.1.10) of the various bilinear form:

$$\begin{aligned} &A \in \mathbb{R}^{N_u \times N_u}: && A_{ij} = a(\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i) \\ &B \in \mathbb{R}^{N_p \times N_u}: && B_{lj} = b(\boldsymbol{\varphi}_i, \psi_l) \\ &M \in \mathbb{R}^{N_u \times N_u}: && M_{ij} = (\rho\boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i) \\ &C(\mathbf{u}) \in \mathbb{R}^{N_u \times N_u}: && C_{ij} = c(\mathbf{u}(t), \boldsymbol{\varphi}_j, \boldsymbol{\varphi}_i). \end{aligned} \tag{5.1.5}$$

We finally introduce the discretization in time. We use an implicit Euler scheme for the momentum equation with a semi-implicit scheme for the convective term.

$$\begin{cases} M\dfrac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + A\mathbf{u}^{n+1} + C(\mathbf{u}^n)\mathbf{u}^{n+1} + B^T\mathbf{p}^{n+1} = \mathbf{F}^{n+1} \\ B\mathbf{u}^{n+1} = 0 \end{cases} \tag{5.1.6}$$

Thus, we have to solve at each time step the following linear system:

$$\begin{bmatrix} \mathcal{N}(\mathbf{u}^n) & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{n+1} \\ \mathbf{p}^{n+1} \end{bmatrix} = \begin{bmatrix} F^{n+1} + \frac{M}{\Delta t}\mathbf{u}^n \\ 0 \end{bmatrix} \tag{5.1.7}$$

where $\mathcal{N} = \frac{M}{\Delta t} + A + C(\mathbf{u}^n)$.

LifeV uses a GMRES algorithm to solve the system (5.1.7), while the

linked algebraic library Trilinos [2] provides preconditioner IFPACK and factorization ILUT.
Concerning the matrix assembling and the various differential operator computations, we use the relatively recent module ETA (Expression Template Assembly) of LifeV because of its better computational performance and flexibility.
We finally underline that we add to the assembled matrix also some terms of the IP stabilization, using the apposite LifeV class. More in detail, we add only the first two terms needed to stabilize the convective term and the incompressibility constraints while dealing with high Reynolds number. Notice that the term linked to the *inf-sup* unstable spaces is not necessary, since we are working with stable FE spaces. For an in depth analysis of IP stabilization see [13]. We postpone to the section 5.3 the discussion about the problems linked to the setting of the IP parameters.

5.2 Implementation Details about the Level-Sets

We now see some details about the correct way of setting the level-set equations in the code implementation. Particularly we will focus on the regularization to make it a distance function, on the setting of the ε parameter (linked to the smooth Dirac function) and on the interpolation between the open and the closed valve.

Regularization. We have seen in section 4.4 that, at the end of algorithm to reconstruct the valve, we have the level-set function φ defined as a combination of polynomials of first and second degree for the closed position (eq. (4.4.1)) and of fifth degree for the open one (eq. (4.4.2)). Thus, φ is not a distance function since it doesn't respect the property of having $|\nabla\varphi| = 1$, $\forall \boldsymbol{x} \in \Omega$. A possible solution is to redefine φ dividing it for its gradient:

$$\varphi(\boldsymbol{x}) \;=\; \frac{\varphi_{old}(\boldsymbol{x})}{|\nabla\varphi_{old}(\boldsymbol{x})|} \tag{5.2.1}$$

This formula ensures that the new level-set function has a unitary gradient only in case of a linear φ_{old}. However, the more we are close to the level zero, the more this approximation is good. Indeed, we can observe that near the level zero the term of the polynomial equation that gives the greatest contribution is the one of first degree. Hence, we are sure that the formula works good in the zone that we need for our Navier-Stokes formulation. Since we have the analytical expression of φ, we can easily apply (5.2.1) and define the new regularized level-set function. In fig. 5.1 is shown how the result is effectively very similar to a distance function.

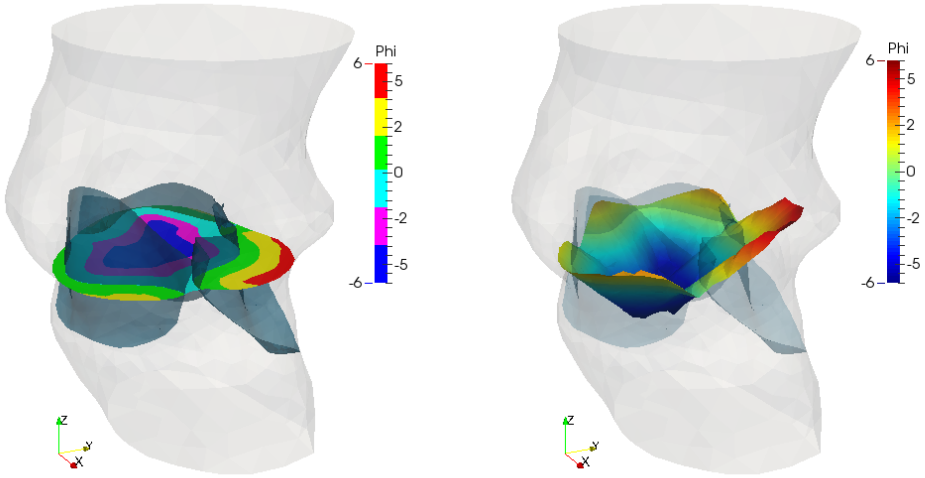


Figure 5.1: Regularized φ shown on the plane $z = 0$ with two different visualizations.

Setting of ε parameter. As we have already discussed in section 3.3, this parameter defines a radius around the zero-level of function φ in which the resistance R_Γ is not null thanks to the action of the smooth Dirac function. Thus, in an intuitive way, 2ε represents the thickness of the valve in the numerical simulation. Since the real valve can be considered as a surface with a null thickness, it is very important to set this value as small as possible. On

the other hand, from a numerical point of view, we have to pay attention in decreasing it too much, because the risk is to have a too thin Dirac function not seen by our space discretization.

A general rule to make working a discrete Dirac function in a discrete domain consists of setting its support wide at least $2h$ (where h is the dimension of the tetrahedral mesh) [23]. This is a conservative rule necessary when the δ_ε is applied to unknown functions of our algebraic problem (i.e. $\mathbf{u}$ or $\mathbf{p}$), because we can evaluate them only at the degrees of freedom (dof) of each finite element (four points in case of $\mathbb{P}^1_{bubble}$).

On the contrary, since we have our level set function always defined analytically, we can go further the number of dof and evaluate $\delta_\varepsilon(\varphi(\boldsymbol{x}))$ directly at the quadrature nodes used to compute the integrals inside the matrix (see eq. (5.1.5)).

In LifeV, the most accurate quadrature rule (that is the default one for the space $\mathbb{P}^1_{bubble}$) considers 64 nodes per element. Hence we can evaluate our smooth Dirac function in all these points for each finite element (using once again the flexibility of the expression template). For this reason we decide to choose ε equal to $h/4$ without the risk of ignoring the presence of the resistance. In fig. 5.2 we show an example of ε equal to $0.5mm$.

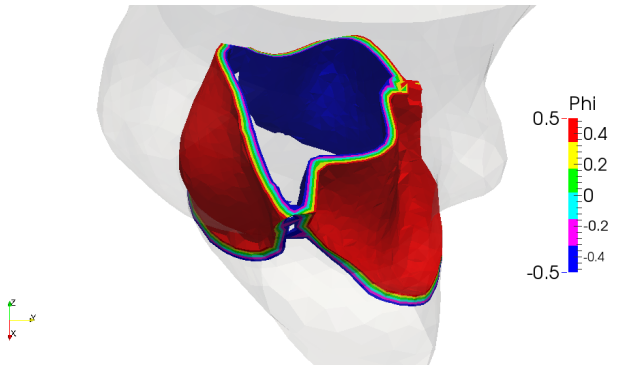


Figure 5.2: Thickness of the valve leaflets in case of $\varepsilon = 0.5mm$ and mesh size $h = 2mm$ in the valve area.

Interpolation of Open and Closed Valve surfaces. As described in detail in section 3.4, we used a reduced zero dimensional model proposed in [33] to compute the angle of the valve at each time step.
Then, the computed angle is transformed in the coefficient AR that represents the percentage of overture of the valve leaflets (see eq. (3.4.3)). In our C++ code implementation, we want to obtain the correct orifice area at each time step (and, as a consequence, the correct AR) linearly interpolating the open and closed level set using a coefficient $\mathcal{K}$ as follows:

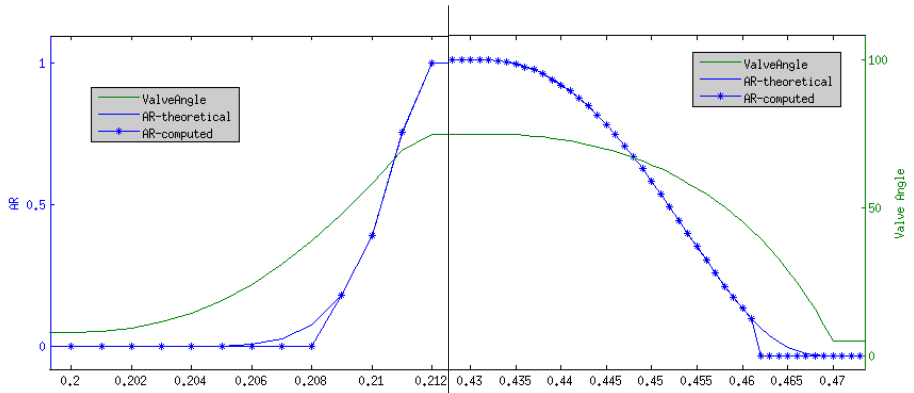
$$\varphi(\boldsymbol{x}, t) \;=\; \mathcal{K}(t) \cdot \varphi_{open}(\boldsymbol{x}) \;+\; (1 - \mathcal{K}(t)) \cdot \varphi_{closed}(\boldsymbol{x}). \tag{5.2.2}$$

We have noticed that setting $\mathcal{K} = AR$ we obtained an orifice area similar to the correct one but with a not negligible error especially in case of small angles. Hence, we proceed as follow to correct the coefficient $\mathcal{K}$:

- We first compute the maximum orifice area (OA_{max}) setting $\mathcal{K} = 1$ in order to make φ equal to the open level set. This is done computing the area of the subset of the plane $z = 0$ in which ϕ is less then ε (see fig. 5.1). Hence, this is the part in which the blood can flow freely.
- Then, we set $\mathcal{K} = AR$ and the desired orifice area $OA_{desired} = AR \cdot OA_{max}$.
- Given the new $\mathcal{K}$, we interpolate $\varphi(\boldsymbol{x}, t)$ using (5.2.2) and we compute the current orifice area OA.
- If $OA < OA_{desired}$, then we increase $\mathcal{K}$.
  If $OA > OA_{desired}$, then we decrease it.
- The procedure continues iteratively since $OA = OA_{desired} \pm tol$. The increment of $\mathcal{K}$ is adapted during the procedure in order to make only few iterations of the algorithm to obtain the final result.

In fig. 5.3 the computed AR (OA/OA_{max}) is compared with the theoretical one, underlining the goodness of the results.

We also give to the user the possibility of setting a minimal orifice area



(a) opening process. (b) closing process.

Figure 5.3: The computed AR versus the theoretical one in both opening and closing procedure, compared also with the valve angle.

(OA_{min}) such that if $AO < OA_{min}$ we force $\mathcal{K} = 0$. Indeed, we noticed that it is useless (and sometimes harmful) to start the opening process, if the consequent orifice area does not allow the flow passage. Hence, it is suggested to set a minimal orifice area equal to the area of a few finite elements in order to be sure that there is enough space in the discrete domain to make the blood pass. In fig. 5.3 it is also shown that for the first time steps the AR computed remains null, since we set $OA_{min} = 25\ mm^2$ with our mesh characterized by $h = 1.85\ mm$ in the valvular zone.

5.3 Boundary Condition and Stabilization Problems

We solved our final simulation with the moving valve using boundary condition (BC) in pressure, because they are more suitable to the reduced model to compute the angular position. Indeed, imposing a flow-rate in the inlet boundary seems not to work with the reduced model because at the beginning of the opening process the imposition of a positive flow-rate with a closed valve generates a non realistic pick in pressure difference causing a too quick opening. More in detail, to impose this kind of condition we use

eq. 3.1.4b in the normal direction as follows

$$\boldsymbol{\tau} \cdot \boldsymbol{n} \cdot \boldsymbol{n} \; = \; p(t), \text{ on } \partial\Omega_{IN} \tag{5.3.1}$$

setting the other two components of the stress vector null.

Concerning the profiles of our BC we proceed as follows:

1. We impose at the outlet a minimum pressure of $80 mmHg$ and a maximum of $120 mmHg$ because they are the physiological values in an healthy ascending aorta. This two values are interpolated in order to obtain a pressure profile qualitatively similar to the one described in [33]. Indeed, we put the minimum at the beginning of the systole, when the left ventricle pressure overcomes the aortic pressure and the maximum at the late systole and we interpolate between the two using a parabolic profile.

2. Following once again [33], the left ventricle diastolic pressure is set equal to $10 mmHg$ and interpolated with the systolic one with two steep polynomial.

3. In systole we impose firstly a flow-rate in the inlet with a fully open valve [8]. Hence, we compute the pressure obtained from this simulation to find the correct profile to impose with the moving valve strategy. Since in [33] the figure with the aortic flow rate was not really clear we take the flow-rate profile from [8] rescaling it with values described in [33]. We make this choice because the paper of Astorino *et al.* [8] is one of the most similar work in literature.

In fig. 5.4 the three BC profiles described above are shown.

The main problem we noticed in the implementation of this work is linked to the stabilization of the convective term in case of high Reynolds numbers. As already said in the first section of this chapter, we had also a problem in using *inf-sup* unstable finite element spaces because we need a stabilization method more coherent with our formulation. At the current state of development of our work, it is not clear if we have the same problem regarding the stabilization of the convective term. Indeed, the IP stabilization

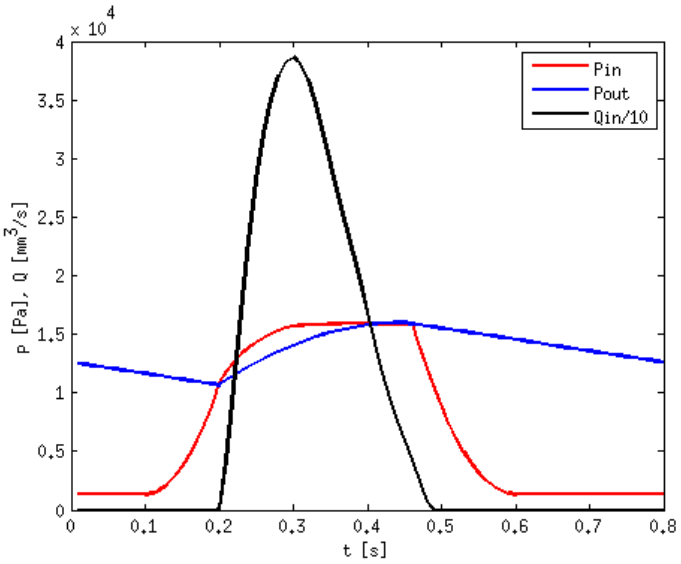


Figure 5.4: The used BC during an heart bit.

works for our problem, but not until the realistic velocity. We find good results only rescaling the BC in order to have velocity about an order of magnitude smaller then the realistic one. However, actually it is not clear if the problem is only linked to the not refined enough mesh, to the choice of the stabilization parameters or to the necessity of adding some stabilization terms specific for our problem. Concerning the mesh, we surely noticed that a coarse mesh in the region near the valve (where blood velocities are higher and the space is less) causes instability problems also with BC rescaled of an order of magnitude. Hence, it is possible that with a more refined mesh (in respect of the one described in section 4.5 and used in our simulation) we can obtain stable results also with realistic Reynolds numbers. To investigate this, it is necessary to perform different simulations using finer meshes and varying the IP parameters. Since our first aim was to implement and check our moving valve method, we decide to run the code rescaling the BC. Hence, we postpone the solution of the stabilization problem to a successive work.

More in detail, we rescale BC of fig. 5.4 as follows:

- We maintain the same realistic outlet pressure for the whole heart bit and the same inlet pressure during the diastole, since we don't have problems while the valve is closed.
- During systole, between the two instants in which inlet and outlet pressures are equal, we impose an inlet pressure simply rescaling its difference with the outlet pressure in order to obtain the desired rescaled velocity.
- To obtain a realistic movement of the leaflets, we also properly rescale the parameters in table 3.1 in order to maintain unmodified the opening and closing dynamics.

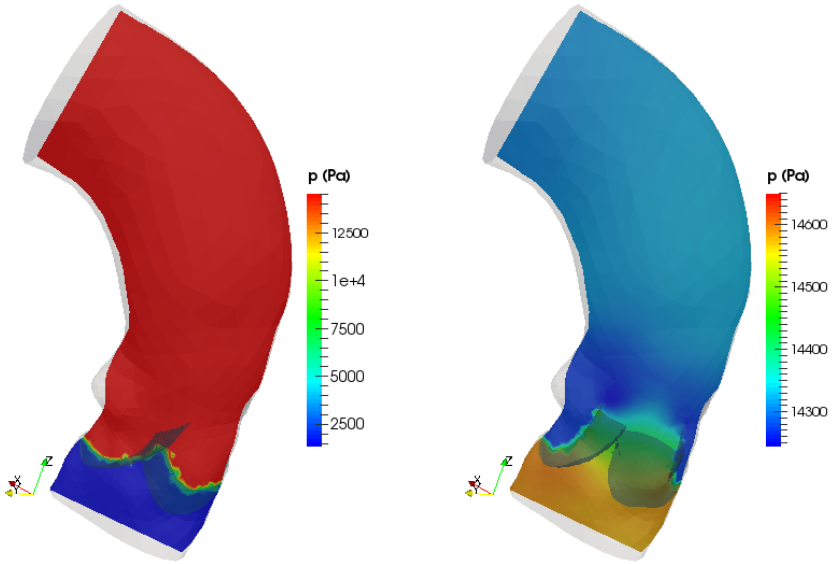
Hence, results described in next section are characterized by a flow rate six time less than the realistic one. However, we will show that they are sufficient to understand the advantages of the inclusion of a moving valve inside the aortic domain. Concerning the IP parameters, we set $\gamma_\beta = 2 \cdot 10^{-6}$ and $\gamma_{div} = 10^{-2}$. We choose these parameters because they have been already used in other thesis dealing with CFD in aorta with LifeV [12].

5.4 Results Analysis

We finally show the result of our simulation. Despite we underline again that they should be considered only preliminaries since we are solving with velocities six time smaller than the realistic ones, we think that the potentialities of this method will appear equally clear.

Pressure Jump. We first analyse the results concerning the pressure gradient. Looking at fig. 5.5a, we can recognize a realistic situation. Indeed, as we have already written, during diastole we solve our simulation with physiological pressure BC. In this phase, since the valve is totally closed, the pressure jump is totally concentrated across the valve leaflets. We want also to underline that the thickness of the region in which the pressure is not constant, depends on the setting of the ε parameter. Since in our case,

thanks to the evaluation of the smooth Dirac function at the quadrature nodes, we can set ε four time less than the diameter of a finite element, the pressure jump is totally concentrated across a single finite element. Hence, the thickness of this region depends only on the dimension of the finite element in the valvular zone and can be further reduced using a finer mesh.
Looking instead at fig. 5.5b, we see that also in the systolic phase most of the pressure difference is located near the valvular zone. This is caused by the shrinkage that the leaflets produce in the aortic diameter. However, the total pressure difference in this phase is obviously much smaller, since the valve is open and represents an obstacle easier to overcome in respect to the closed one.
We finally noticed that obviously this kind of result can be obtained only solving the problem with pressure BC and considering a valve model. An extension of our work to an FSI model can be the natural evolution of this thesis and can help to model better the wall stresses linked to this pressure jump near the valvular zone.

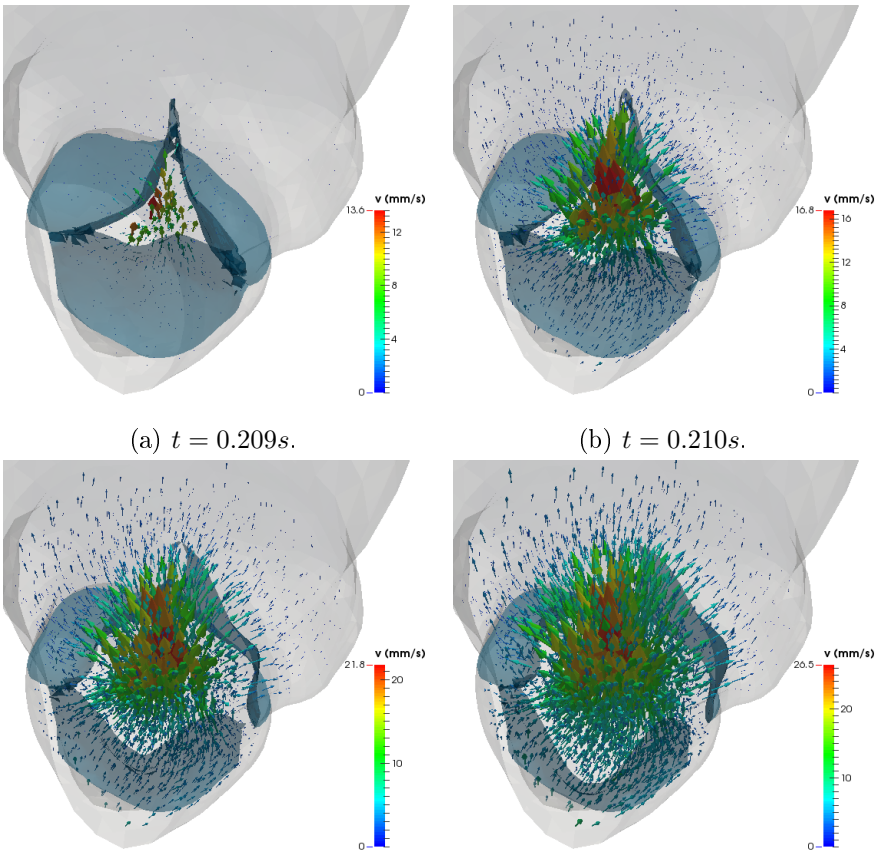


(a) $t = 0.6s$, diastole. (b) $t = 0.3s$, systole.

Figure 5.5: Difference in the pressure jump across the valve leaflets between diastole and systole.

Opening Process and Systole. In fig. 5.6 we first show the opening phase of the leaflets with the velocity vectors crossing the increasing orifice. This phase has a really little characteristic time (see also fig. 5.3a). Furthermore, the blood velocities are very small everywhere and the flux can be considered totally laminar, without the presence of vortexes anywhere. This is more evident looking at fig. 5.7a, 5.7b and 5.7c where the velocity streamlines are shown. Indeed, they are almost parallel each others and compressed in the valvular zone especially in the first instant of the opening process.
In the other three figures of 5.7 we show the streamlines in different instants of the systolic flow with a fully open valve.
In fig. 5.7d, we are still in the early systole. Despite the velocities are greater than the ones of the opening process, the flow is still laminar.
Instead, once the systolic pick is reached (fig. 5.7e), we start to see the generation of vortexes in the two coronary sinus. As we discussed in section 4.4 commenting fig. 4.4, the space between the lumen surface of the aorta and the leaflets is bigger in these two sinus in respect to the non-coronary one. We can suppose that this difference is exactly caused by the generation of these two vortexes that allow blood to flow towards the coronaries, avoiding a leaflet too close to the wall.
The intensity of these vortexes becomes greater during the late systole (fig. 5.7f) while the blood flow is decreasing its intensity. In this phase the vortexes extend their region occupying also part of the aortic root. These phenomena becomes more evident while the closure process begins.
Before analysing it, we show again in fig. 5.8 the vortex in a coronary sinus, zooming behind the leaflet while the closure process is already started. Indeed, the coronary flow rate profile in time has got its maximum pick during diastole. The presence of this vortex behind the leaflet can be considered as a sort of source for the blood flow inside the coronary, despite we don't include coronaries in our geometry (because of the extra computational cost necessary). This kind of vortex is not visible in a simulation without the valve surface. Hence, the inclusion of the valve leaflet is necessary if the goal of a simulation is the study of the flux also inside the coronaries (for example to study the performance of an artificial valve which should be able

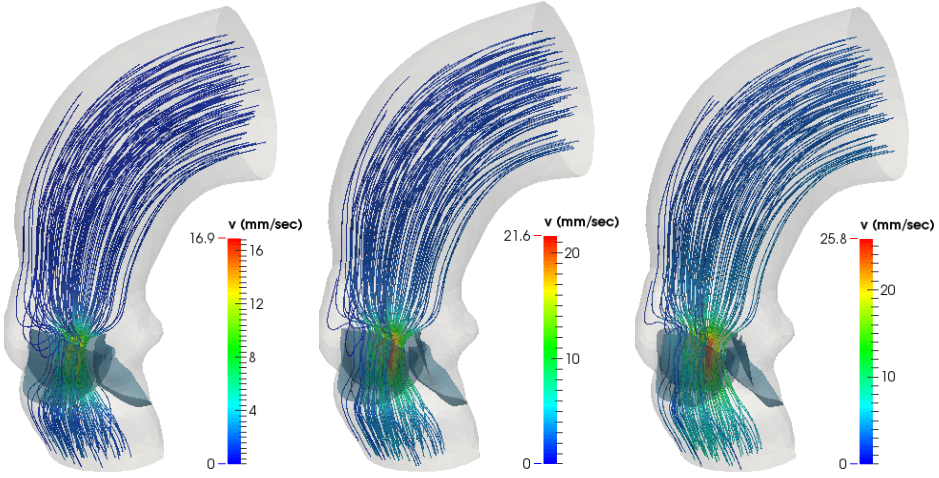
to maintain these vortexes).



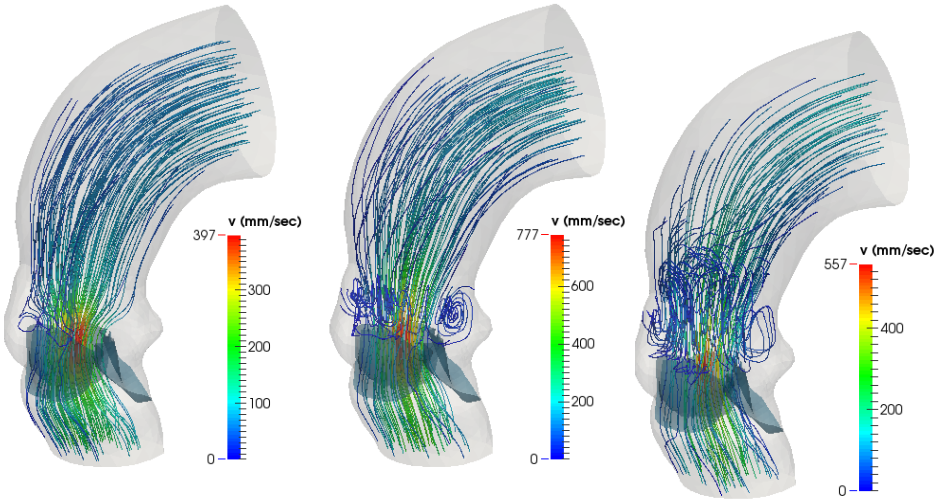
(a) $t = 0.209s$. (b) $t = 0.210s$.

(c) $t = 0.211s$. (d) $t = 0.212s$.

Figure 5.6: Velocity vectors near the valve leaflets during the opening process.

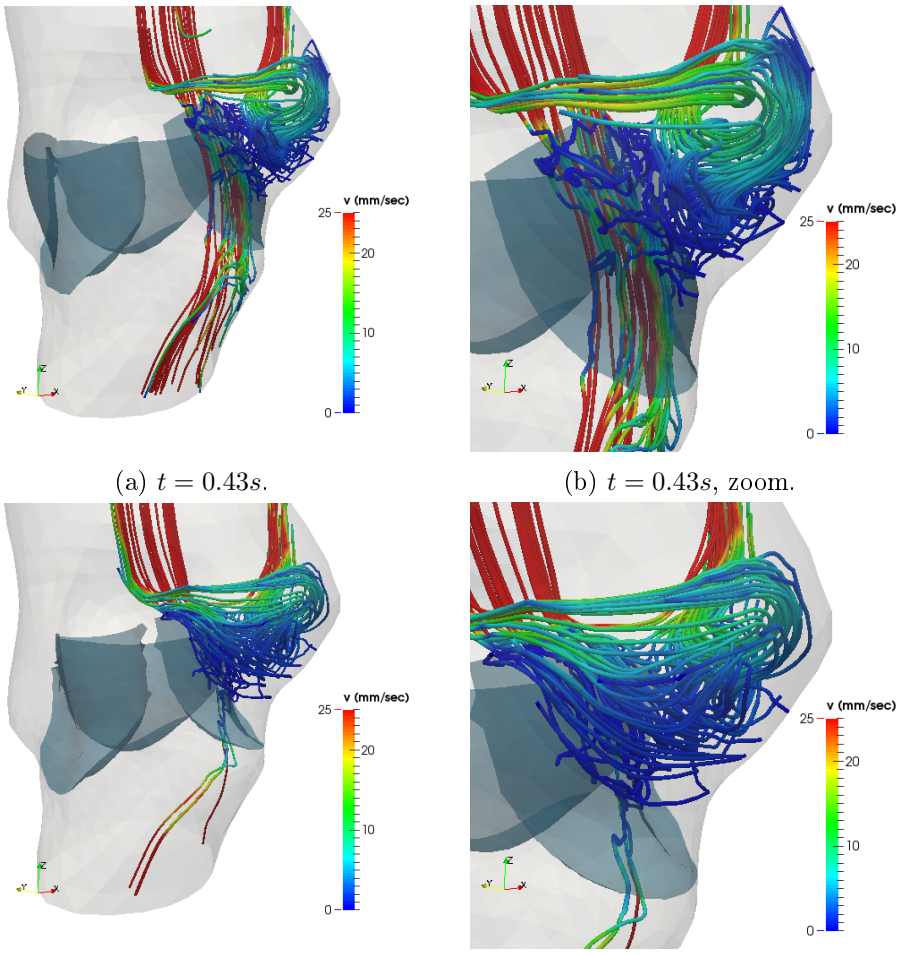


(a) $t = 0.210s$, opening. (b) $t = 0.211s$, opening. (c) $t = 0.212s$, opening.



(d) $t = 0.25s$, early systole. (e) $t = 0.31s$, systolic pick. (f) $t = 0.4s$, late systole.

Figure 5.7: Streamlines during opening process and systolic flow.

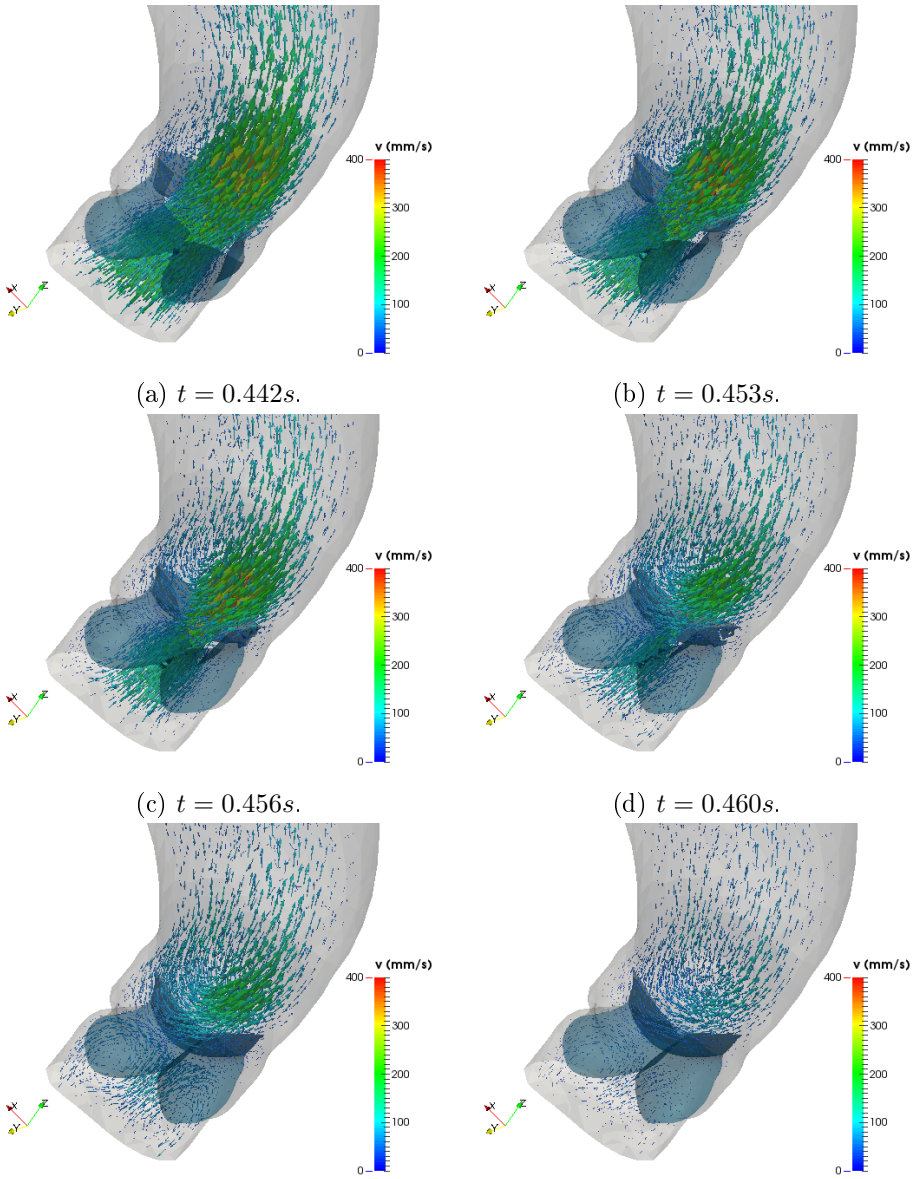


(a) $t = 0.43s$. (b) $t = 0.43s$, zoom.

(c) $t = 0.45s$. (d) $t = 0.45s$, zoom.

Figure 5.8: Vortex genesis in a coronary sinus during late systole.

Closing Process and Diastole. In fig. 5.9 and 5.10 we show the blood flow during the closing process and the beginning of diastole both with vectors and streamlines.
The two vortexes already visible in fig. 5.7f in this phase join each others generating a big recirculation vortex. This recirculation starts when the valve is open and there is still a positive flow rate (fig. 5.10a, 5.10b, 5.10c).
After the ending of the closing precess, the total flow rate becomes null, but the vortex is still evident and it occupies the whole width of the aorta. Looking at fig. 5.10d and 5.10e we notice it clearly.
Its intensity decreases in time but remains not negligible also when the valve is already closed from a little (fig. 5.10f). We highlight that this type of vortex during diastole was reported in in-vivo studies on healthy patients [38].
Once again, we underline that this kind of phenomena are in general not so evident in numerical simulation without valve leaflets. Hence, we can suppose that the valve leaflets represent a sort of obstacle for the fluid necessary to model better this kind of recirculation, especially if we want to locate them in the correct place.

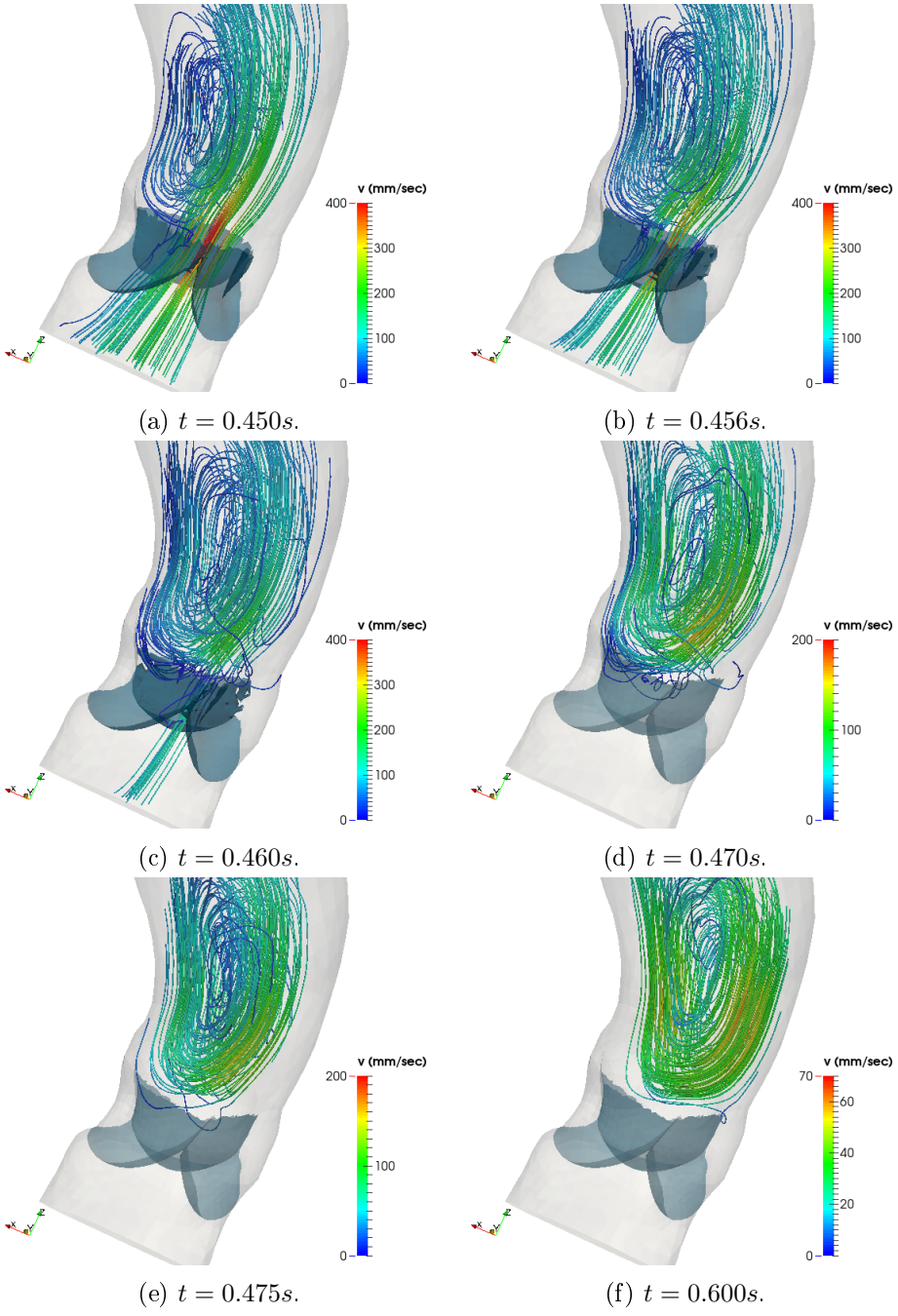


(a) $t = 0.442s$. (b) $t = 0.453s$.

(c) $t = 0.456s$. (d) $t = 0.460s$.

(e) $t = 0.462s$. (f) $t = 0.500s$.

Figure 5.9: Velocity vectors during the closing process.



(a) $t = 0.450s$. (b) $t = 0.456s$.

(c) $t = 0.460s$. (d) $t = 0.470s$.

(e) $t = 0.475s$. (f) $t = 0.600s$.

Figure 5.10: Streamlines during the closing process and the diastole.

Conclusions and Perspectives

In this thesis we propose a strategy to include in numerical simulation of aortic fluid dynamc a moving aortic valve model. We started from the reduced model for heart valves proposed by Astorino *et al.* [8] called resistive immersed surface (RIS) in which the mechanics of the leaflets is neglected and the valve is replaced by two immersed surfaces fixed in space: one describing the 3D shape of the valve during the opened phase, the other during the closed one. The two valve surfaces are inserted in the classic Navier-Stokes equations adding a dissipative term with a resistance that can be interpreted as a penalization parameter enforcing the condition of null velocity on it. In our work we proposed to use the same RIS method of Astorino *et al.* [8] but we extended their model representing the open and closed valve surfaces through a level set formulation. In this way, the immersed surfaces are described analytically and we avoid the need of inserting this surfaces in the finite element mesh domain. Moreover we proposed to compute the movement of the valve between its closed and open position using a reduced zero-dimensional model [33] which computes the valvular angle governed by pressure and flow-rate values in the left ventricle and in the aorta. For each time step of the cardiac cycle, the valvular level set surface is then interpolated starting from the open and closed representations without the need of generating a moving mesh.

The thesis deals with a patient-specific geometry: for this reason we develop a full framework which starts from medical images and finishes with the numerical simulations. This is also an important goal toward the development of patient-specific models to help clinicians in studying and understanding diseases. Moreover our valve reconstruction procedure can also be used in other fields of applications, for example to construct surface meshes representing the valve leaflets to perform simulations with leaflets included as fixed rigid walls in the domain [12, 41].

Concerning the obtained results, we solved our final simulation with the moving valve using boundary condition in pressure, because they work better with the reduce model to compute the angular position. Simulations are run with the finite element library LifeV [1]. The main problem we noticed with our proposed method is linked to the stabilization of the convective term in case of high Reynolds numbers. In this work, we decided to focus on the results of our simulations with a moving valve, but rescaling all the boundary conditions of about an order of magnitude to avoid the problems of instabilities. Obviously, the first future development of this thesis is to deal with this problem. In particular further invastigation is needed using finer meshes and varying the IP parameters to understand if it is necessary to implement a specific stabilization method for our problem. Once the stability problem will be solved, future developments will include a deeper study of the method including:

- the use of different physiological conditions;
- the comparison between our results and the results obtained without the inclusion of the leaflets in the domain and/or the results obtained modelling leaflets as fixed boundaries inside the domain;
- the use of different geometries;
- the analysis of results as a function of the mesh size;

As future perspectives, this method can be applied to study the fluid dynamic in the aorta in presence of a bicuspid aortic valve, allowing a patient-specific inclusion of the bicuspid leaflets. Moreover it can be extended in order to be included in a fluid-structure interaction model of the aortic arch to study for example different types of valve prosthesis. Finally, it can be easily extended to the other cardiac valves, helping in the building of a complete heart-integration model.

Acknowledgments

First of all, I would like to thank Professor Alfio Quarteroni for giving me the opportunity to do this thesis on a topic so passionate and for allowing me to spend a long time in Lausanne, at a so stimulating university such as the École Polytechnique Fédérale de Lausanne.
A thanks goes to Ph.D. Aymen Laadhari for the readiness with which he followed me since our first meeting in Lausanne.
I would like to give a really special thanks to Ph.D. Elena Faggiano for introducing me to medical applications of Computational Science & Engineering (from imaging to CFD). It is definitely thanks to her if now I have a better idea about my future.
A special thanks goes to all the research group in Lausanne, in particular to Professor Simone Deparis for following me even when it was not required and for suggesting me what to focus on during periods in which the work was not producing the desired results.
I finally acknowledge Dr. Roberto Scrofani and Dr. Sonia Ippolito for providing us with the medical data and for helping in the understanding of medical images.

Ringrazio innanzi tutto i miei genitori per il sostegno che mi hanno dato soprattutto nei momenti più difficili e le mie sorelle Lalli e Frin che vedo sempre meno per colpa della lontananza, ma con le quali è sempre più bello passare del tempo quando capita l'occasione. Un grazie va anche a tutto il resto della mia famiglia, in particolare: a nonna Iole (e a tutto quello che mi ha cucinato), a Maria e ad Angela; a nonna Rosa per il suo sostegno; ai miei zii per avermi 'adottato' a Favignana a tempo indeterminato; a nonno Mimì e nonno Tatà.
Ringrazio Luca per aver reso più leggero tutto il tempo passato insieme a studiare e per avermi trasmesso la passione per la nerdaggine (e per la birra

artigianale) e Fra per le mega-mosse che si è strasentito soprattutto nel periodo di Losanna.
Un ringraziamento particolarissimo va a Novi per tutto il tempo passato insieme (soprattutto quello totalmente dedicato a fare gli stupidi) e per avermi ascoltato nei momenti importanti.
Da buon precisino elencherò in un bell'elenco puntato e in rigoroso ordine casuale tutti i vari gruppi di amici sparsi per l'Italia che voglio ringraziare:

- I Boosteri e in particolare il suo leader indiscusso Paola che nonostante la lontananza ci rende sempre partecipi della sua straordinaria follia con le sue mitiche note vocali che (a sua insaputa) l'hanno ormai resa celebre in gran parte del nord Italia.

- La Famiglia di Palermo, in particolare il Pipps, Andrea, Giuvà e Billo (ma ancor di più tutti quelli che non nomino e che si offenderanno per questo). A mio cugino va un secondo grazie per tutto il tempo passato insieme.

- Il gruppo-Campus il cui nome attuale non si può certo scrivere in una tesi.

- I devoti al Divin Porcello che hanno reso il mio periodo di Losanna molto più divertente del previsto.

- Tutti i compagni di tutti i vari sport di questi anni, in particolare lo Spektor No Limits team che ormai si fa vedere sempre meno spesso sulle nevi, ma che ha avuto un ruolo fondamentale per evadere dalla studio.

Vorrei infine dedicare questa tesi ad un Koala, un Koala meraviglioso che mi ha reso felice.

Bibliography

[1] Lifev project. http://www.lifev.org/.

[2] Trilinos project. http://www.trilinos.org/.

[3] The vascular modeling toolkit (vmtk). http://www.vmtk.org/.

[4] Visualization toolkit (vtk). http://www.vtk.org/.

[5] L. Antiga, B. Ene-Iordache, and A. Remuzzi. Computational geometry for patient-specific reconstruction and meshing of blood vessels from mr and ct angiography. *Medical Imaging, IEEE Transactions on*, 22(5):674–684, 2003.

[6] L. Antiga, M. Piccinelli, L. Botti, B. Ene-Iordache, A. Remuzzi, and D. A. Steinman. An image-based modeling framework for patient-specific computational hemodynamics. *Medical & biological engineering & computing*, 46(11):1097–1112, 2008.

[7] M. Astorino, J.-F. Gerbeau, O. Pantz, and K.-F. Traore. Fluid–structure interaction and multi-body contact: application to aortic valves. *Computer Methods in Applied Mechanics and Engineering*, 198(45):3603–3612, 2009.

[8] M. Astorino, J. Hamers, S. C. Shadden, and J.-F. Gerbeau. A robust and efficient valve model based on resistive immersed surfaces. *International journal for numerical methods in biomedical engineering*, 28(9):937–959, 2012.

[9] O. K. Baskurt. *Handbook of hemorheology and hemodynamics*, volume 69. IOS press, 2007.

[10] B. Bellhouse. Fluid mechanics of a model mitral valve and left ventricle. *Cardiovascular research*, 6(2):199–210, 1972.

[11] J. Bonnemain, E. Faggiano, A. Quarteroni, and S. Deparis. A patient-specific framework for the analysis of the haemodynamics in patients with ventricular assist device. Technical report, MOX, Politecnico di Milano, 2012.

[12] D. Bonomi. Numerical study of the influence of aortic valve leaflets in the fluid-dynamics in ascending aorta - master thesis, AY 2012-2013.

[13] E. Burman, M. A. Fernández, and P. Hansbo. Continuous interior penalty finite element method for oseen's equations. *SIAM journal on numerical analysis*, 44(3):1248–1274, 2006.

[14] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997.

[15] T. Chan, S. Esedoglu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.

[16] T. Chan and L. Vese. Active contours without edges. *Image Processing, IEEE Transactions on*, 10(2):266–277, 2001.

[17] H.-D. Cheng, X. Jiang, Y. Sun, and J. Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[18] L. D. Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[19] C. A. Conti, A. Della Corte, E. Votta, L. Del Viscovo, C. Bancone, L. S. De Santo, and A. Redaelli. Biomechanical implications of the congenital bicuspid aortic valve: a finite element study of aortic root function from in vivo data. *The Journal of thoracic and cardiovascular surgery*, 140(4):890–896, 2010.

[20] J. De Hart, F. Baaijens, G. Peters, and P. Schreurs. A computational fluid-structure interaction analysis of a fiber-reinforced stentless aortic valve. *Journal of biomechanics*, 36(5):699–712, 2003.

[21] J. De Hart, G. Peters, P. Schreurs, and F. Baaijens. A three-dimensional computational analysis of fluid–structure interaction in the aortic valve. *Journal of biomechanics*, 36(1):103–112, 2003.

[22] A. Della Corte, C. Bancone, C. A. Conti, E. Votta, A. Redaelli, L. Del Viscovo, and M. Cotrufo. Restricted cusp motion in right-left type of bicuspid aortic valves: a new risk marker for aortopathy. *The Journal of thoracic and cardiovascular surgery*, 144(2):360–369, 2012.

[23] M. Discacciati, A. Quarteroni, and S. Quinodoz. Numerical approximation of internal discontinuity interface problems. *SIAM Journal on Scientific Computing*, 35(5):A2341–A2369, 2013.

[24] E. Faggiano, L. Antiga, G. Puppini, A. Quarteroni, G. B. Luciani, and C. Vergara. Helical flows and asymmetry of blood jet in dilated ascending aorta with normally functioning bicuspid valve. *Biomechanics and modeling in mechanobiology*, pages 1–13, 2012.

[25] A. A. Farag. Edge-based image segmentation. *Remote Sensing Reviews*, 6(1):95–121, 1992.

[26] M. Fedele, L. Barbarotta, and F. Cremonesi. Medical image segmentation using the rsfe split-bregman algorithm, 2013.

[27] M. A. Fernández, J.-F. Gerbeau, and V. Martin. Numerical simulation of blood flows through a porous interface. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(06):961–990, 2008.

[28] V. Girault and P.-A. Raviart. Finite element methods for navier-stokes equations. 1986.

[29] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[30] B. E. Griffith, X. Luo, D. M. McQueen, and C. S. Peskin. Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method. *International Journal of Applied Mechanics*, 1(01):137–177, 2009.

[31] V. Hristidis, H. Xiaolei, and G. Tsechpenakis. *Information discovery on electronic health records*, chapter 10. CRC Press, 2010.

[32] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[33] T. Korakianitis and Y. Shi. Numerical simulation of cardiovascular dynamics with healthy and diseased heart valves. *Journal of biomechanics*, 39(11):1964–1982, 2006.

[34] C. Li, C.-Y. Kao, J. Gore, and Z. Ding. Minimization of region-scalable fitting energy for image segmentation. *Image Processing, IEEE Transactions on*, 17(10):1940–1949, 2008.

[35] C. Li, C. Xu, C. Gui, and M. Fox. Level set evolution without reinitialization: a new variational formulation. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 430–436 vol. 1, 2005.

[36] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Siggraph Computer Graphics*, volume 21, pages 163–169. ACM, 1987.

[37] G. Marom, R. Haj-Ali, E. Raanani, H.-J. Schäfers, and M. Rosenfeld. A fluid–structure interaction model of the aortic valve with coaptation and compliant aortic root. *Medical & biological engineering & computing*, 50(2):173–182, 2012.

[38] U. Morbiducci, R. Ponzini, G. Rizzo, M. Cadioli, A. Esposito, F. De Cobelli, A. Del Maschio, F. M. Montevecchi, and A. Redaelli. In vivo quantification of helical blood flow in human aorta by time-resolved three-dimensional cine phase contrast magnetic resonance imaging. *Annals of biomedical engineering*, 37(3):516–531, 2009.

[39] E. Orso. Analisi computazionale dell'interazione fluido-struttura in aorta ascendente con valvola aortica stentless - master thesis, AY 2012-2013.

[40] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005.

[41] S. Pasta, A. Rinaudo, A. Luca, M. Pilato, C. Scardulla, T. G. Gleason, and D. A. Vorp. Difference in hemodynamic and wall stress of ascending thoracic aortic aneurysms with bicuspid and tricuspid aortic valve. *Journal of biomechanics*, 46(10):1729–1738, 2013.

[42] C. S. Peskin. The immersed boundary method. *Acta numerica*, 11, 2002.

[43] A. Quarteroni. *Numerical models for differential problems*, volume 2. Springer, 2010.

[44] S. Salsa. *Partial differential equations in action.* Springer, 2009.

[45] S. Salsa. *Equazioni a derivate parziali: Metodi, modelli e applicazioni.* Springer Italia Srl, 2010.

[46] J. Serra. Introduction to mathematical morphology. *Computer vision, graphics, and image processing*, 35(3):283–305, 1986.

[47] Y. Shi, Y. Zhao, T. Yeo, and N. Hwang. Numerical simulation of opening process in a bileaflet mechanical heart valve under pulsatile flow condition. *The Journal of heart valve disease*, 12(2):245–255, 2003.

[48] M. A. Sid-Ahmed. *Image Processing.* McGraw-Hill, 1994.

[49] J. W. Thomas. *Numerical partial differential equations: finite difference methods*, volume 1. Springer, 1995.

[50] C. Vergara, F. Viscardi, L. Antiga, and G. B. Luciani. Influence of bicuspid valve geometry on ascending aortic fluid dynamics: a parametric study. *Artificial organs*, 36(4):368–378, 2012.

[51] M. H. Yacoub, P. J. Kilner, E. J. Birks, and M. Misfeld. The aortic outflow and root: a tale of dynamism and crosstalk. *The Annals of thoracic surgery*, 68(3):S37–S43, 1999.

[52] Y. Yang, C. Li, C.-Y. Kao, and S. Osher. Split bregman method for minimization of region-scalable fitting energy for image segmentation. In *Advances in Visual Computing*, volume 6454 of *Lecture Notes in Computer Science*, pages 117–128. Springer Berlin Heidelberg, 2010.

[53] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.